Thomas Bakaysa

PKI Lab

Before I could start the lab. Some things to take care of was: Changed the hosts file in /etc/hosts so that the right IP address gets mapped.

Copying and modifying the openssl.cnf file was easy, small changes were made with no problem. Creating the CA's public and private key was not a problem.

Task 1: Becoming a CA

This section of the CA's certificate indicates that this is a CA. Specifically the 'CA:TRUE' line.

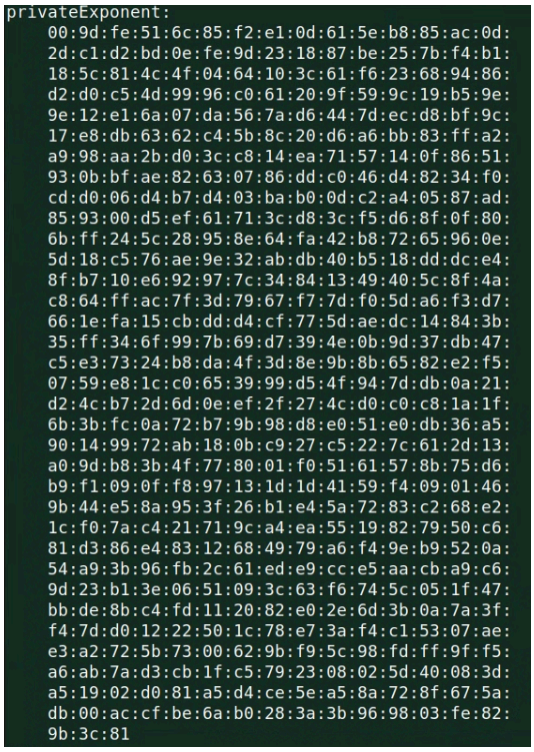
```
        X509v3 Basic Constraints: critical
            CA:TRUE
 Signature Algorithm: sha256WithRSAEncryption
      36:71:a5:42:3b:8f:97:54:ea:b2:65:0a:2e:c8:fe:23:ce:a7:
```

```
Modulus:
    00:de:43:b8:45:25:f7:f8:b4:2b:1a:e1:ca:1b:29:
    7a:10:af:aa:32:3f:15:f5:10:0c:f6:20:58:6b:40:
    17:49:59:25:d7:7e:3d:f2:0b:6b:8e:1f:ee:5e:f9:
    da:4c:f4:a3:4f:c8:0c:2d:3d:fa:46:77:b4:38:2f:
    52:b6:10:88:dc:18:a7:42:3e:4f:e1:25:6f:e5:62:
    c4:b0:9c:bf:2d:e7:d5:72:9f:0b:f7:59:f8:bd:dc:
    fd:c3:e7:2f:ad:c0:3c:3c:a3:16:9d:9c:43:e3:d3:
    9b:2e:5c:92:e3:2e:39:aa:8b:a3:bf:e2:db:ed:28:
    2f:65:31:40:8f:b6:ff:c8:b7:8a:e6:9c:65:0b:59:
    0f:f3:32:ca:bb:08:1a:74:9d:d5:1a:d3:32:6c:a6:
    04:c4:77:81:8c:a1:c6:01:a6:f4:33:3e:56:7f:44:
    d4:11:06:f9:d5:b6:b3:70:e4:ab:7f:1c:e8:fc:16:
    f9:2c:43:07:53:ec:57:ae:7f:1d:3b:17:f5:45:7d:
    25:71:74:5d:26:1d:59:15:d7:c9:17:4e:f6:64:3e:
    e0:42:ff:af:7e:af:41:ce:8b:46:e9:56:20:e8:34:
    48:5f:ce:51:13:6f:c7:a7:d4:81:c2:0d:cd:12:b6:
    f3:fb:b2:8a:58:f3:81:20:33:7c:36:bc:f5:bf:9e:
    1c:28:ad:c9:27:ab:93:f0:5d:dd:03:9c:02:9b:cb:
    06:12:21:ae:ec:48:3c:ce:fe:25:a2:e2:bd:cf:72:
    ba:d1:27:b9:31:01:da:25:0b:92:ea:02:b2:d5:0a:
    40:48:5b:f5:08:50:7f:e3:45:b0:43:e0:87:ad:de:
    26:7b:25:d6:e4:de:3a:a7:54:8b:00:70:f0:48:31:
    82:54:28:fb:2c:9d:ee:ee:ea:1e:74:68:0f:63:ec:
    3a:55:45:a0:f9:88:e3:10:51:70:d0:e2:a1:fe:b3:
    a0:ef:83:0b:21:b1:81:c7:8a:cf:3d:a9:61:e3:52:
    56:ca:04:2f:df:f5:a4:33:a1:1c:18:3d:09:08:5a:
    38:1f:51:ab:ae:37:67:fb:fc:db:96:11:07:dd:2b:
    ae:09:b9:c6:61:20:79:d8:7a:62:3b:f0:5d:8d:ae:
    46:ba:cc:9c:ff:38:60:a5:8e:14:8e:46:0e:8d:e4:
    c1:e9:0c:1d:f6:ba:fd:e0:7b:91:43:7f:1e:81:10:
    8d:22:f4:cc:e6:81:bd:80:c5:5a:3d:7a:cf:b1:ca:
    ac:55:89:62:dd:41:ca:4d:99:d9:e0:f9:58:87:d0:
    83:8c:96:7d:9e:f1:c4:30:89:af:10:b9:5d:78:c0:
    74:b4:b7:a2:87:58:79:fd:c1:b8:72:51:77:bf:31:
    8d:68:5b
```

The modulus and public exponent, also known as 'e', were present in the certificate. Which is expected since this would be required to verify the signature that is attached to the bottom of the certificate.

```
publicExponent: 65537 (0x10001)
```

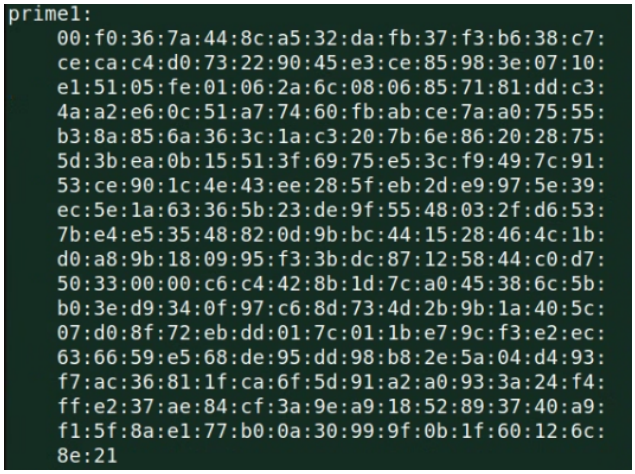
The certificate understandably does not contain the private exponent nor the primes that were used to create the RSA key pair. However, using the "openssl rsa …" command, extracting such information was a piece of cake after remembering what the password was supposed to be.

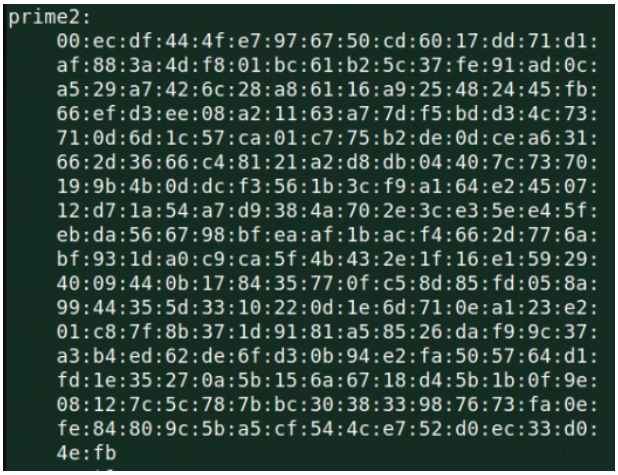


The information provided contains everything required for the RSA key, both the private and public values. First was the modulus and public exponent, as it was in the certificate.

The private key was right below the public exponent. It's called the private exponent but we also know it as 'd'.

The two primes used to create the modulus are also listed in the command results. Prime1 is 'p' while prime 2 is 'q'. By providing us with all this information, it should allow us to confirm the values stored by the file.





```
--
-- Representation of RSA private key with information for the CRT
-- algorithm.
--
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

This excerpt from the Openssl documentation is what I used to determine which values are which. Looking at this, shows me that the system saves all components required for the RSA algorithm.

All of these are really big numbers, which makes sense. We aren't working with shortened keys anymore since we are simulating an actual web server with encrypted traffic. Unlike the other lab, we are now working with numbers that are the proper length to adequately protect data traffic to the server.

Task 2: Generating a certificate request

Generating the certificate request was no problem. The command went through just fine. However, because I ran the command before I read the entirety of the task, I had to re-run it with the hostname extensions to add the additional names.

```
[02/17/25]seed@VM:~/.../image_www$ openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -subj "/CN=www.bakaysa20
25.com/O=Thomas B. /C=US" -passout pass:dees
Generating a RSA private key
..............................+++++
.............................+++++
writing new private key to 'server.key'
-----
```

```
[02/17/25]seed@VM:~/.../image_www$ openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -subj "/CN=www.bakaysa20
25.com/O=Thomas B. /C=US" -addext "subjectAltName = DNS:www.bakaysa2025.com, DNS:www.bakaysa2025a.com, DNS:www.bakaysa2025b.com, DN
S:www.bakaysa2025thomas.com" -passout pass:dees
Generating a RSA private key
..+++++
...................................................+++++
writing new private key to 'server.key'
-----
```

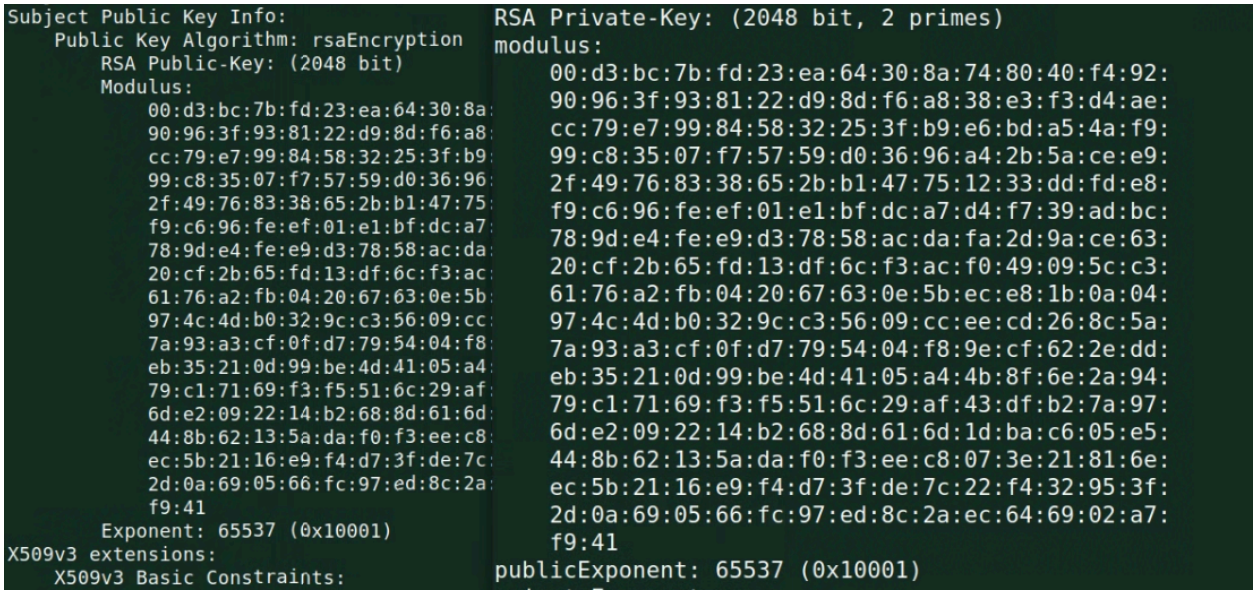
Running the command twice, the latter with the added extension. Each time it ran, I checked the values for the RSA key and the modulus was different each time.

```
Requested Extensions:
    X509v3 Subject Alternative Name:
        DNS:www.bakaysa2025.com, DNS:www.bakaysa2025a.com, DNS:www.bakaysa2025b.com, DNS:www.bakaysa2025thomas.com
```

Task 3: Generating the server certificate

The only problem I ran into is that I somehow skipped the directory creation step, I had to make sure the required directories were created and the “index.txt” file existed. After the setup was finished, the command ran with no problems and the server’s certificate was created.

```
[02/17/25]seed@VM:~/.../image_www$ openssl ca -config openssl.cnf -policy policy_anything -md sha256 -days 3650 -in server.csr -out
 server.crt -batch -cert ca.crt -keyfile ca.key
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Feb 18 04:16:27 2025 GMT
            Not After : Feb 16 04:16:27 2035 GMT
        Subject:
            countryName               = US
            organizationName          = Thomas B.
            commonName                = www.bakaysa2025.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                82:BB:74:32:51:1E:74:F4:96:22:F2:56:CE:59:73:B9:D8:8D:D6:37
            X509v3 Authority Key Identifier:
                keyid:15:BA:6C:9F:C1:63:80:DD:6C:17:10:D2:D0:59:45:22:C7:64:F9:11

            X509v3 Subject Alternative Name:
                DNS:www.bakaysa2025.com, DNS:www.bakaysa2025a.com, DNS:www.bakaysa2025b.com, DNS:www.bakaysa2025thomas.com
Certificate is to be certified until Feb 16 04:16:27 2035 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```



Double checking the modulus of the server and the server's key shows that the certificate was generated correctly.

Task 4: Deploying Certificate in an Apache-Based HTTPS Website

Created a new ssl config file for the new server. Naming everything bakaysa2025 and making sure all the directories and files were placed and named correctly.

```
root@9664b10d2625:/etc/apache2/sites-available# cat bakaysa2025_apahce_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bakaysa2025
    ServerName www.bakaysa2025.com
    ServerAlias www.bakaysa2025a.com
    ServerAlias www.bakaysa2025b.com
    ServerAlias www.bakaysa2025thomas.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/bakaysa2025.crt
    SSLCertificateKeyFile /certs/bakaysa2025.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/bakaysa2025
    ServerName www.bakaysa2025.com
    DirectoryIndex index_red.html
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning message
ServerName localhost
```

I renamed the certificate and key files to reflect what they actually are instead of just calling them server.crt and server.key.

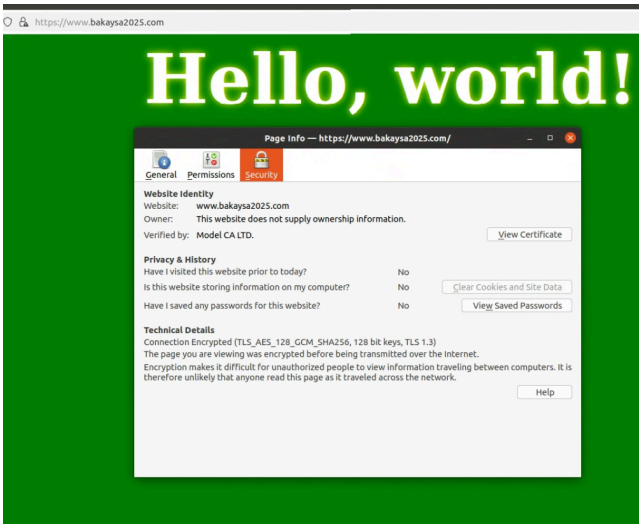
```
[02/18/25]seed@VM:~/.../image_www$ cd certs
[02/18/25]seed@VM:~/.../certs$ ls -l
total 32
-rw-rw-r-- 1 seed seed 5919 Feb 18 00:20 bakaysa2025.crt
-rw------- 1 seed seed 1854 Feb 18 00:20 bakaysa2025.key
-rw-rw-r-- 1 seed seed 5808 Dec  5  2020 bank32.crt
-rw-rw-r-- 1 seed seed 1854 Dec  5  2020 bank32.key
-rw-rw-r-- 1 seed seed 1923 Dec  5  2020 modelCA.crt
-rw-rw-r-- 1 seed seed  231 Dec  5  2020 README.txt
```

I also modified the Dockerfile so that it can run correctly with the renamed files.

```
[02/17/25]seed@VM:~/.../image_www$ cat Dockerfile
FROM handsonsecurity/seed-server:apache-php

ARG WWWDIR=/var/www/bakaysa2025

COPY ./index.html ./index_red.html $WWWDIR/
COPY ./bakaysa2025_apache_ssl.conf /etc/apache2/sites-available
COPY ./certs/bakaysa2025.crt ./certs/bakaysa2025.key  /certs/

RUN  chmod 400 /certs/bakaysa2025.key \
     && chmod 644 $WWWDIR/index.html \
     && chmod 644 $WWWDIR/index_red.html \
     && a2ensite bakaysa2025_apache_ssl

CMD  tail -f /dev/null
```

This also made me realize that I need to move the newly created certificates and keys to the ‘certs’ folder. I also renamed them, as shown in the picture at the very top of this page, to better reflect what they’re for.

```
[02/18/25]seed@VM:~/.../image_www$ dcbuild
Building web-server
Step 1/7 : FROM handsonsecurity/seed-server:apache-php
 ---> 2365d0ed3ad9
Step 2/7 : ARG WWWDIR=/var/www/bakaysa2025
 ---> Using cache
 ---> 439c8119f74d
Step 3/7 : COPY ./index.html ./index_red.html $WWWDIR/
 ---> Using cache
 ---> d135a331a5a1
Step 4/7 : COPY ./bakaysa2025_apache_ssl.conf /etc/apache2/sites-available
 ---> 2f09aea25e22
Step 5/7 : COPY ./certs/bakaysa2025.crt ./certs/bakaysa2025.key  /certs/
 ---> 03832583874f
Step 6/7 : RUN  chmod 400 /certs/bakaysa2025.key      && chmod 644 $WWWDIR/index.html      && chmod 644 $WWWDIR/index_red.html
 && a2ensite bakaysa2025_apache_ssl
 ---> Running in a5ef67935c41
Enabling site bakaysa2025_apache_ssl.
To activate the new configuration, you need to run:
  service apache2 reload
Removing intermediate container a5ef67935c41
 ---> c061e7f0581c
Step 7/7 : CMD  tail -f /dev/null
 ---> Running in ea55e6a3409b
Removing intermediate container ea55e6a3409b
 ---> c110745f1d2c

Successfully built c110745f1d2c
Successfully tagged seed-image-www-pki:latest
```



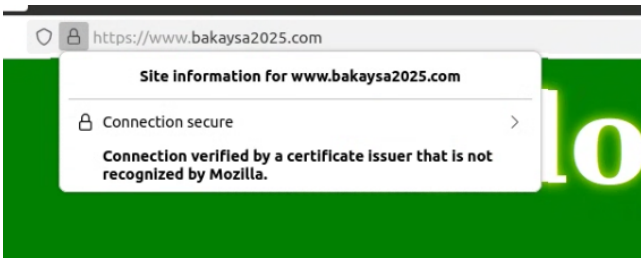
I then ran ‘dcbuild’ to rebuild the container with the correct files and certificates.

The website came up no problem, the first time I opened it the background was red, which I assume is because the connection was not encrypted. This was simply because the browser defaulted to http instead of https.

As shown in the screenshot, the website traffic is encrypted. Even though firefox tried to dissuade me from opening the site, stating that this site is suspicious since it doesn’t recognize the CA being used.

After adding our in house CA certificate to the list of trusted certificates, firefox finally trusts the website enough to not give a warning when trying to connect to it.

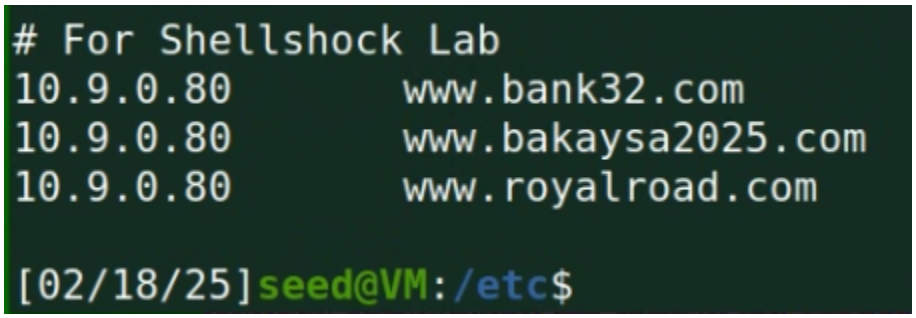
It does make it known that Mozilla itself doesn’t recognize this CA.



Task 5: Man in the middle attack

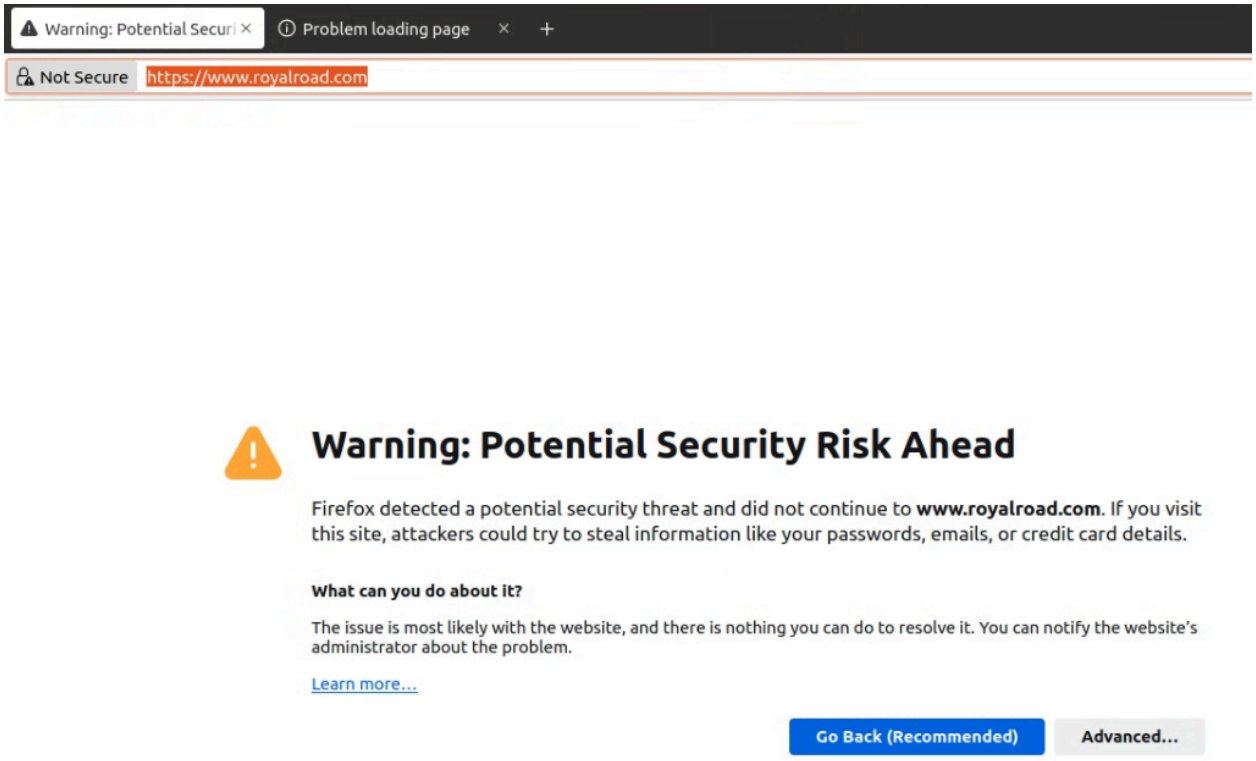
Set up the fake website and poison my own DNS cache.

```
root@be1756434cda:/etc/apache2/sites-available# cat royalroad_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/bakaysa2025
    ServerName www.royalroad.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/bakaysa2025.crt
    SSLCertificateKeyFile /certs/bakaysa2025.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/bakaysa2025
    ServerName www.royalroad.com
    DirectoryIndex index_red.html
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning message
ServerName localhost
```



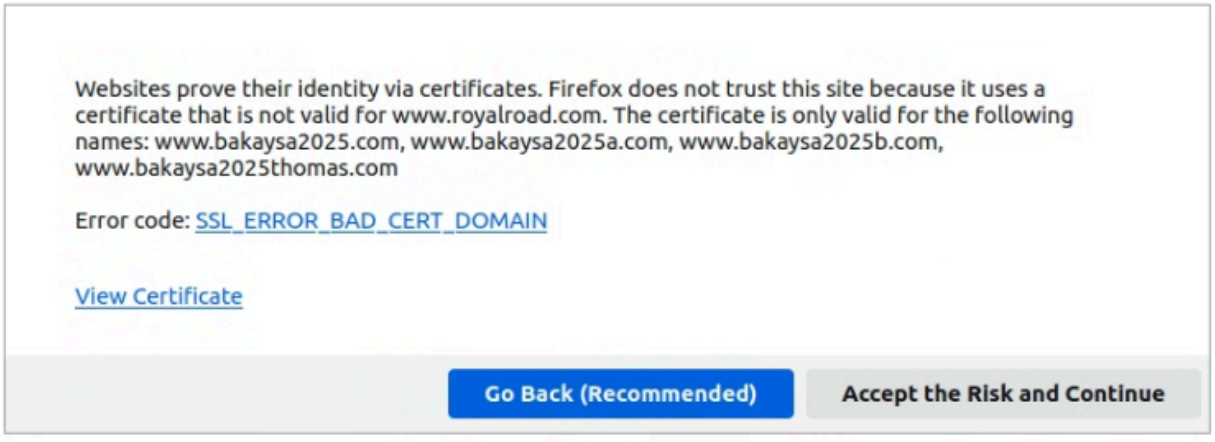
I chose royalroad.com. After creating the fake website and poisoning my own DNS cache to re-direct to the website we just set up.

In a real life scenario, this would most likely redirect to a clone of the website that is poised to extract as much information from the user as it can.



As it should, firefox detect that the certificates do not match the domain name of the website. It rightly smells something fishy and warns the user.

I like the firefox gives detailed information on why it suspects the website. Here it shows that the certificate is wrong, it's presenting my bakaysa2025 certificate for a website called royalroad.



Task 6: MitM Attack with Compromised CA

Using the fact that we have access to a CA that my browser already trusts, we can just create a certificate for royalroad.com that will trick our browser into thinking that nothing is wrong. Unless you look at the warning that firefox gives, stating that this is a certificate that Mozilla doesn't recognize, the connection looks legitimate.

Trying to create the spoofed certificate made me want to tear my hair out. I was doing everything right except for adding the extension for different DNS names. I'm not completely sure why it needed it, but it would not work otherwise. The certificate would not work with just the commonName field.

```
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4101 (0x1005)
        Validity
            Not Before: Feb 18 08:19:54 2025 GMT
            Not After : Feb 16 08:19:54 2035 GMT
        Subject:
            countryName               = US
            organizationName          = Royal Inc.
            commonName                = www.royalroad.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                45:FD:B4:BA:83:1D:29:A8:F6:55:6B:31:0A:C3:37:C5:4A:1C:8B:B1
            X509v3 Authority Key Identifier:
                keyid:15:BA:6C:9F:C1:63:80:DD:6C:17:10:D2:D0:59:45:22:C7:64:F9:11

            X509v3 Subject Alternative Name:
                DNS:www.royalroad.com, DNS:www.royalroad2.com
Certificate is to be certified until Feb 16 08:19:54 2035 GMT (3650 days)
```