Transport Layer Security (TLS)

Table of Contents

Review Questions . . . 1
- Network Traffic Interception . . . 1
- TLS Protection . . . 2
- Importance of Certificates in TLS . . . 2

Tinkering with TLS . . . 3
- Missing `client_random` Value . . . 3
- Missing `server_random` Value . . . 4
- Missing Hash Value . . . 5

Tricking Your Browser . . . 6
- Certificate Authority . . . 6
- Target Domain . . . 9
- Creating a Certificate for the Target Domain . . . 10
- Configuring Host Operating System . . . 12
- Accessing the Target Domain . . . 15

Review Questions

NOTE Assignment questions and instructions

Network Traffic Interception

(Q 1.1)

Ways in which an attacker can intercept network traffic

- ARP Spoofing
- DNS Cache Poisoning
- Attacks on BGP
  - One-hop Attack
  - Sub-prefix Hijacking

All the attacks listed above cause packets to be routed to a machine controlled by the attacker (interception), so that they can read and/or manipulate traffic, and monitor activity and/or carry out attacks.

Attackers can usually cause most damage when they are in the same local network as their victims, which is why everyone cautions against using public WiFi Access Points for insecure HTTP browsing and sensitive internet usage, such as internet banking.

TLS Protection

(Q 1.2)

Protections TLS provides to reduce problems caused by interceptions

- TLS makes use of Digital Certificates to prove identity, so an interception attempt will fail, because the attacker will not be able to provide a valid certificate for the actual server.
- TLS includes random values in its messages and uses those values in its integrity checking, so even if an attacker is recording all messages, they cannot replay them, as the server will generate a different random value for every connection, which will lead to the handshake failing.
- TLS uses the Diffie-Hellman (DH) Key Exchange Algorithm to exchange key generation material and the algorithm has been created such that the master secret cannot be created by someone intercepting the connection.
    - Perfect Forward Secrecy is also provided by TLS connections using the Ephemeral DH (DHE) Key Exchange Algorithm, which implies that even if the attacker is able to compromise the key material of the current TLS session, they cannot compromise the previous or future connections using those values.
- Once new keys are generated on both ends at the end of a TLS handshake, TLS encrypts all the handshake messages on both ends and sends it across in a Finished message. The client and the server compare these exchanged encrypted messages and only if they match, does the handshake succeed. An attacker who has intercepted a message and modified it will be caught here, as they cannot change the encrypted 'Finished' message.
- After the handshake is complete, TLS encrypts and MACs all application data, so an attacker cannot understand the data being shared and cannot even mess with the encrypted data due to the MAC signature.

Importance of Certificates in TLS

(Q 1.3)

Digital Certificates are a key component of TLS, as they provide authentication, i.e., they prove that the server the client is communicating with, is actually the server that they requested and not someone impersonating the server.

> **NOTE** If an impersonating server sends another server’s certificate to the client, the impersonating server won’t be able to produce a valid signature for the Diffie-Hellman `g^b` value that it sends to the client, as it does not have access to the private key of that certificate to create a valid signature.

So, a LOT of the security of TLS is dependent on the client verifying the server’s certificate chain. In the absence of this, anyone can impersonate the real server. An attacker can intercept messages and can carry out a Monkey in the Middle (MITM) Attack.

In a certificate chain, each certificate is signed by the certificate having higher authority that it and this finally ends at the root certificate by a Certificate Authority (CA), which has the highest authority and is self-signed. This self-signed root certificate is a trusted certificate that ships with Operating Systems and browsers.

Generally when attackers want to play mischief and impersonate a site, they cannot get a CA to sign their fake impersonated certificate (because a real certificate for that site exists and the attacker is not the owner of the actual domain), so attackers self-sign their fake certificate. Web browsers and protocols are smart-enough to realise that a self-signed certificate that is not signed by any trusted CAs signals security issues and so does not let the user access the site.

So if the signature verification is skipped, then an attacker presenting a fake self-signed certificate will not be caught and a successful MITM Attack can be carried out.

The MITM Attack:

- The client starts communication with the server by first establishing a TCP connection with it and then sending a Client Hello message.
- The server responds with a Server Hello message and its certificates.
- The attacker will replace the legitimate certificate chain with its own fake self-signed certificate, change and sign the Diffie-Hellman parameter sent by the server (`g^b` to `g^b'`) using the public key in the fake certificate and then send it to the client.
- As the client is not verifying the signature on the certificate it is being presented, it accepts the fake certificate, fake signature and malicious `g^b'` pre-master secret, and successfully verifies the pre-master secret (`g^b'`) using the public key in the fake certificate.
- The attacker then compromises the Client Key Exchange message, where instead of the server receiving `g^a` from the client, it receives `g^a'` from the attacker, as the attacker has intercepted the actual value (`g^a`).
- Thus, the attacker was able to generate a different set of keys for the client (`g^ab'`) and the server (`g^a'b`), and every time messages are exchanged between the two, it will decrypt the intercepted messages, make modifications as necessary, encrypt them again and send them across.
- The attacker then encrypts and MACs the client and server Finished messages, thus acting as a transparent proxy, unbeknownst to the two communicating parties.

Tinkering with TLS

NOTE Assignment questions and instructions

Missing `client_random` Value

(Q 2.1)

The protocol is secure against Replay Attacks even with the `client_random` value missing, due to the presence of the `server_random` value.

If an attacker decides to replay client messages, the server will send a different server random value each time, so the generated master secret will be different each time, leading to the generation of different messages, message hashes and cipher texts, which will cause the TLS handshake to fail in the final Finished message stage.

Missing `server_random` Value

(Q 2.2)

TLS 1.2 is not secure against Replay Attacks if Key Encapsulation or Static Diffie-Hellman (DH) Key Exchange Algorithm are used to exchange keys, but TLS 1.2 is secure against Replay Attacks if the Ephemeral Diffie-Hellman (DHE) Key Exchange Algorithm is used.

Key Encapsulation or Static DH

When a TLS handshake is using Key Encapsulation or Static DH, an attacker can replay client messages to have the server generate the same responses every time messages are replayed, thus allowing Replay Attacks.

Replay attacks allow the attacker to repeat the encrypted application data communication with the server, which can cause some unwanted repeated actions on user data, depending on the contents of the request. The attacker will not be able to read and understand the encrypted data, but they don’t need to do that to cause harm. They can keep doing these attacks for a lot of connections and they will eventually replay a message that modifies the state of a user’s data, thus causing problems.

To carry out this attack, the attacker has to

- Capture and store all messages of a TLS communication.
- Establish a TCP connection with the server.
- Blindly replay all the stored messages (the ones sent from the client to the server) without changing their contents, one-by-one in order of timestamp as the server keeps responding.

The attacker has to send the following intercepted messages unchanged one-by-one in order, after creating a TCP connection with the server:

- Client Hello
- Client Key Exchange
- Finished
- Application data

DHE

With the Ephemeral Diffie-Hellman (DHE) Key Exchange Algorithm, new key generation material is generated for every connection, to maintain Perfect Forward Secrecy.

NOTE It is being assumed that the server does not cache the key generation material (`b`, `g`,

> `p`, `g^b`, etc.).

This means that if no caching is assumed, the absence of the `server_random` value does not make TLS 1.2 vulnerable to Replay Attacks. The newly generated Key Exchange parameters provide the same protection as the `server_random` value does.

Missing Hash Value

(Q 2.3)

TLS 1.2 is vulnerable to Downgrade attacks whether hashes are present or not, but missing hash values in the handshake makes it easier to carry out these attacks, as it is an important checking mechanism against tampering.

An attacker intercepting communication can modify a TLS Client Hello message and strip out all the strong cipher suites and just use weak ones. The server then choses the best one out of the weak cipher suites it receives. The attacker records all the parameters that it receives and can trivially figure out the keys due to the weak ciphers used. Thus, the attacker can now selectively encrypt, decrypt and modify messages and transparently read and/or modify messages without anyone noticing.

The presence of hashes in the Finished message just slows down the attacker as it has to compute a new hash (which is computer from a version of all messages exchanged in the handshake that includes the modifications it made to all the previous messages). This can cause timeout issues leading to handshake failure, but the absence of hashes eases some of that burden on the attacker, speeding up the attack, thus leading to better chances of the attack succeeding.

The attack steps:

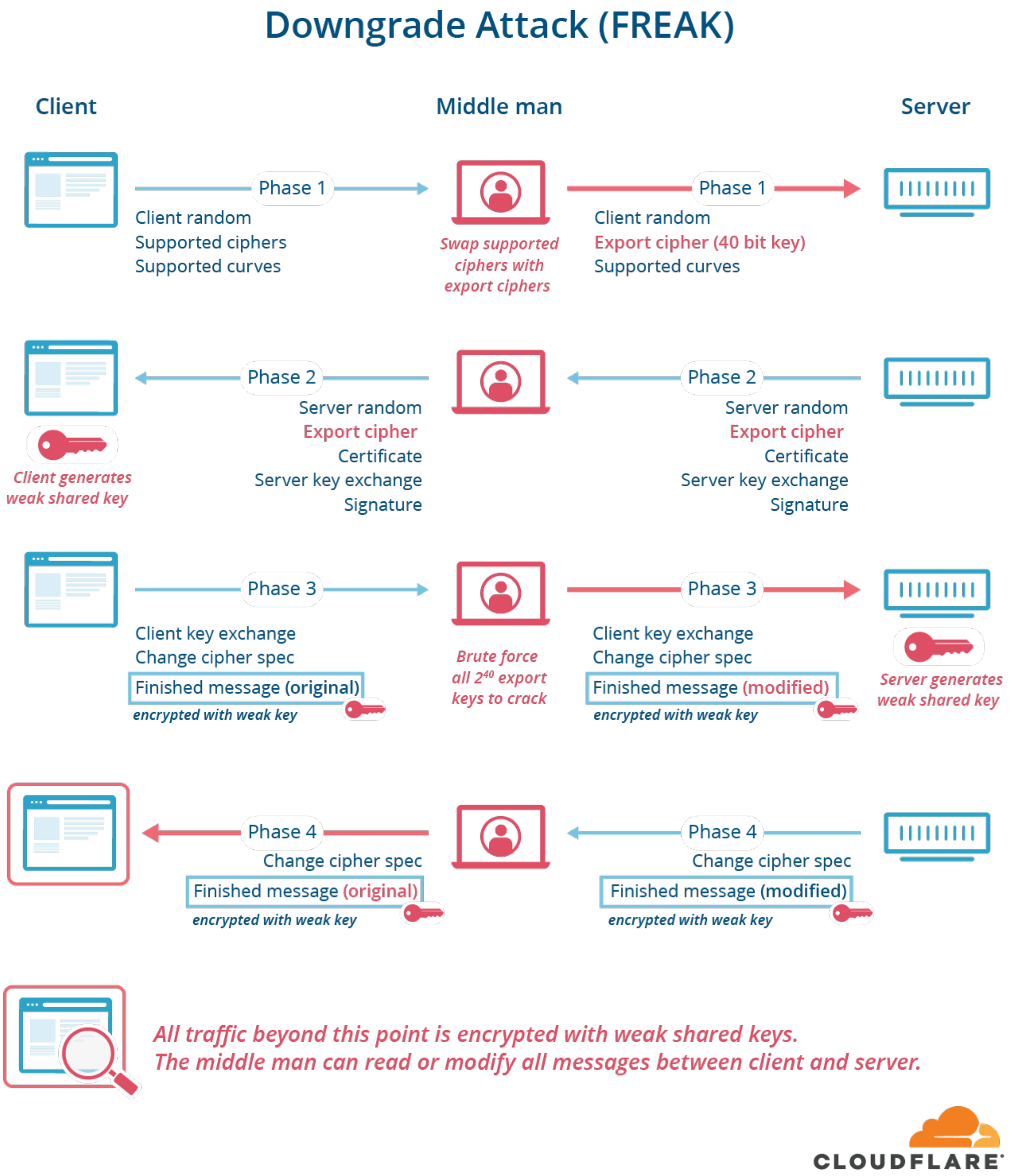


Image source

Tricking Your Browser

NOTE

- Assignment questions and instructions
- More on Digital Certificates

Hijacking a domain on a local by issuing fake certificates.

Certificate Authority

- Create a directory for all Certificate Authority (CA) information. (`bucs558-ca` here.)

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>mkdir bucs558-ca
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>cd bucs558-ca
```

- The CA's certificate and private key have been provided in the assignment. Paste the contents in two separate files `bucs558-ca.crt` and `bucs558-ca.key` respectively.
- It is easy to extract the public key of the CA from its private key

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>openssl rsa -in bucs558-ca.key -pubout
-out bucs558-ca.pubkey
writing RSA key

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat bucs558-ca.pubkey
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA3ak32ELiqRv/7SQ1r1Hn
DKVfnepIMuQ4n1OKjhx0IMJ9MjZtnf2UA4P/M8LS9cfAoKCbl3GNVjfkzajCDIay
m4u+8FLRgau5UCDaEPcwvMasQsOzuBBvrSkra5bQzMZDTiBxrrSuUVJ1G7lmghP4
4Eu4VD1D6U9VtZbyrDLmGGQT5YJphlF5hphYK3MQndJZxmdLK66BdtuOAg9gNg6n
Sm+balUdRFBZy4TIVp2GR2Bc/XxVukekb6rIMPHIfVZv1Q/U3ZcTV7myk4hW3ewJ
/Sm793ZcxbLW+uIogSd2mjUTfNGK1t24Xw8bXyL8mz3H5Lb9KTFhZ1pNJ0XscgnZ
GrEeg+OcqX6n5i35VJHNzuDaN7jgVSFDDzKOxWGVkd2GTPdBXWJArna92MEHBIkR
3MQu5gOODpKKTnvylO62s0FLMBwWofNVahC6s1F0dHxMP/vEIGP/+FUIfHh8XAml
kvZ3hMZBwJF1+zp5Pi9n+A3uGQ0sXztch7ARTRzvZ/XcPR78vkp4kVfwBcIL2t6q
TA1SbP2IsoYnb9hHOoHwhkBt8DnsksDmgoZOSLeBALHVbgpyhich8rSlUN5nX1XP
NUYZl42MugOE8hrfZ0187/fDxM3VSFdwzyy+mYRcZWKctlmnvRtpFopYxHsxj3/a
DQT48GdOOOItzLsL49ZlEUMCAwEAAQ==
-----END PUBLIC KEY-----
```

- Details of the provided CA certificate can be found as shown below.
    - The CA e-mail is `unsavorycaffine@bu.edu`.

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>openssl x509 -in bucs558-ca.crt -noout
-text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            bc:c7:dc:99:cc:50:0f:30
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = MA, L = Boston, O = BostonUniversityCS558, OU = CS558, CN
= cs558.bu.edu, emailAddress = unsavorycaffine@bu.edu
        Validity
            Not Before: Apr  3 21:49:43 2023 GMT
            Not After : May  3 21:49:43 2023 GMT
        Subject: C = US, ST = MA, L = Boston, O = BostonUniversityCS558, OU = CS558,
CN = cs558.bu.edu, emailAddress = unsavorycaffine@bu.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

```
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:dd:a9:37:d8:42:e2:a9:1b:ff:ed:24:35:af:51:
                    e7:0c:a5:5f:9d:ea:48:32:e4:38:9f:53:8a:8e:1c:
                    ...
                    da:0d:04:f8:f0:67:4e:38:e2:2d:cc:bb:0b:e3:d6:
                    65:11:43
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                17:48:AD:B7:D1:C5:FA:7E:74:EC:40:0E:3A:A8:6D:D9:CE:B8:7E:EF
            X509v3 Authority Key Identifier:
                keyid:17:48:AD:B7:D1:C5:FA:7E:74:EC:40:0E:3A:A8:6D:D9:CE:B8:7E:EF

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         1b:e7:f5:90:06:67:7c:bc:d0:c5:72:30:2d:8b:f4:14:95:3c:
         ea:a5:23:b9:e8:97:99:30:a5:b2:74:62:ff:e9:58:88:5d:3e:
         ...
         52:bd:ef:85:c7:ce:59:03:35:d4:23:26:9e:38:40:cf:34:82:
         7c:f2:db:ae:62:11:03:e8
```

- Create a configuration file `bucs558-ca.cnf` for future certificate generation and add the following content to it

```
# we use 'ca' as the default section because we're using the ca command
[ ca ]
default_ca = bucs558-ca

[ bucs558-ca ]
# a text file containing the next serial number to use in hex. Mandatory.
# This file must be present and contain a valid serial number.
serial = ./serial

# the text database file to use. Mandatory. This file must be present though
# initially it will be empty.
database = ./index.txt

# specifies the directory where new certificates will be placed. Mandatory.
new_certs_dir = ./newcerts

# the file containing the CA certificate. Mandatory
certificate = ./bucs558-ca.crt

# the file contaning the CA private key. Mandatory
private_key = ./bucs558-ca.key

# the message digest algorithm. Remember to not use MD5
default_md = sha256
```

```
# for how many days will the signed certificate be valid
default_days = 30

# a section with a set of variables corresponding to DN fields
policy = bucs558-ca-policy

[ bucs558-ca-policy ]
# if the value is "match" then the field value must match the same field in the
# CA certificate. If the value is "supplied" then it must be present.
# Optional means it may be present. Any fields not mentioned are silently
# deleted.
countryName = match
stateOrProvinceName = supplied
organizationName = supplied
commonName = supplied
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

- Run the following commands in the CA directory (`bucs558-ca`), in preparation of generating certificates.

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>mkdir newcerts
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>touch index.txt
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>echo 01 > serial
```

Target Domain

- Pick a domain to target (`harshkapadia.me` here), create a directory for the domain (`harshkapadia.me` here).

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>mkdir harshkapadia.me
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>cd harshkapadia.me
```

- Generate a 4096 bit RSA private key and send the output to a file named `harshkapadia.me.key`.

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\harshkapadia.me>openssl genrsa -out
harshkapadia.me.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
................................++++
...................................++++
e is 65537 (0x010001)
```

- Create a Certificate Signing Request (CSR) that the CA will use to generate a certificate for this domain.

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\harshkapadia.me>openssl req -new -key
harshkapadia.me.key -out harshkapadia.me.csr
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 Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Massachusetts
Locality Name (eg, city) []:Boston
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Harsh Kapadia
Organizational Unit Name (eg, section) []:hk
Common Name (e.g. server FQDN or YOUR name) []:harshkapadia.me
Email Address []:contact@harshkapadia.me

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

- Create an extensions configuration file `harshkapadia.me.extensions.cnf` and add the following content to it. This is mainly to add all the subdomains that can be authenticated by the certificate.
    - The `*.harshkapadia.me` entry implies that all the sub-domains of `harshkapadia.me` are covered by the certificate.

```
basicConstraints=CA:FALSE
subjectAltName=@hk_subject_alt_names
subjectKeyIdentifier = hash

[ hk_subject_alt_names ]
DNS.1 = harshkapadia.me
DNS.2 = *.harshkapadia.me
```

Creating a Certificate for the Target Domain

- All the configuration files required for generating a certificate have already been created above.
- Generate a certificate for the target domain

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>openssl ca -config ./bucs558-ca.cnf
-out ../harshkapadia.me/harshkapadia.me.crt -extfile
../harshkapadia.me/harshkapadia.me.extensions.cnf -in
../harshkapadia.me/harshkapadia.me.csr
Using configuration from ./bucs558-ca.cnf
```

```
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :ASN.1 12:'Massachusetts'
localityName          :ASN.1 12:'Boston'
organizationName      :ASN.1 12:'Harsh Kapadia'
organizationalUnitName:ASN.1 12:'hk'
commonName            :ASN.1 12:'harshkapadia.me'
emailAddress          :IA5STRING:'contact@harshkapadia.me'
Certificate is to be certified until May 12 21:53:40 2023 GMT (30 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

- The `harshkapadia.me.crt` target domain certificate file should have been generated in the target domain's directory (`harshkapadia.me` here). The details of the certificate can be printed and the certificate can be verified as well.

```

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\harshkapadia.me>ls -a
.  ..  harshkapadia.me.crt  harshkapadia.me.csr  harshkapadia.me.extensions.cnf
harshkapadia.me.key  harshkapadia.me.pubkey

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\harshkapadia.me>openssl x509 -in
harshkapadia.me.crt -noout -text # Certificate details
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = MA, L = Boston, O = BostonUniversityCS558, OU = CS558, CN
= cs558.bu.edu, emailAddress = unsavorycaffine@bu.edu
        Validity
            Not Before: Apr 12 21:53:40 2023 GMT
            Not After : May 12 21:53:40 2023 GMT
        Subject: C = US, ST = Massachusetts, O = Harsh Kapadia, OU = hk, CN =
harshkapadia.me, emailAddress = contact@harshkapadia.me
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:f2:56:53:c6:05:0e:a4:41:40:0d:3f:8b:bc:c6:
                    04:b5:6c:89:e3:8e:47:29:df:a6:b1:8c:18:3a:ed:
                    ...
                    22:bf:43:e1:a1:48:4d:8d:6d:ea:62:df:9f:61:b5:
                    38:79:35
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Alternative Name:
                DNS:harshkapadia.me, DNS:*.harshkapadia.me
            X509v3 Subject Key Identifier:
                7F:6C:46:27:5D:09:3C:F1:79:8E:3D:64:67:4F:F4:D1:C1:0E:88:6B
    Signature Algorithm: sha256WithRSAEncryption
         ab:fc:45:29:87:68:72:6a:4c:24:d2:e1:81:6b:11:40:78:a4:
         cc:c5:82:67:51:88:ad:35:5a:fc:e1:0f:69:71:ea:34:39:c0:
         ...
         18:eb:bd:76:76:b4:b7:c9:51:30:ab:4e:80:46:ce:14:cf:bc:
         64:0e:fd:3f:66:69:5e:87

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\harshkapadia.me>openssl verify -CAfile
../bucs558-ca/bucs558-ca.crt harshkapadia.me.crt # Verify certificate
harshkapadia.me.crt: OK
```

- In the CA directory (`bucs558-ca`) some files have been generated

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>ls -a
.   bucs558-ca.cnf  bucs558-ca.key     index.txt       index.txt.old  serial
..  bucs558-ca.crt  bucs558-ca.pubkey  index.txt.attr  newcerts       serial.old

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat index.txt
V       230512215340Z           01      unknown /C=US/ST=Massachusetts/O=Harsh
Kapadia/OU=hk/CN=harshkapadia.me/emailAddress=contact@harshkapadia.me

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat index.txt.old
# empty

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat index.txt.attr
unique_subject = yes

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat serial
02 # The next serial number was automatically generated

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>cat serial.old
01 # The last serial number that was manually configured

X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw\bucs558-ca>ls -a newcerts
.  ..  01.pem # The CA has a copy of the certificate with the serial number as the
title
```

Configuring Host Operating System

- In this case, the host Operating System (OS) is Microsoft Windows 11.
- To be able to redirect the target domain to another web page, it has to be redirected on the OS.

Add the following lines to the `C:\Windows\System32\drivers\etc\hosts` file.

```
\# Added by Harsh Kapadia for BU CAS CS 558
127.0.0.1 harshkapadia.me
127.0.0.1 networking.harshkapadia.me
```

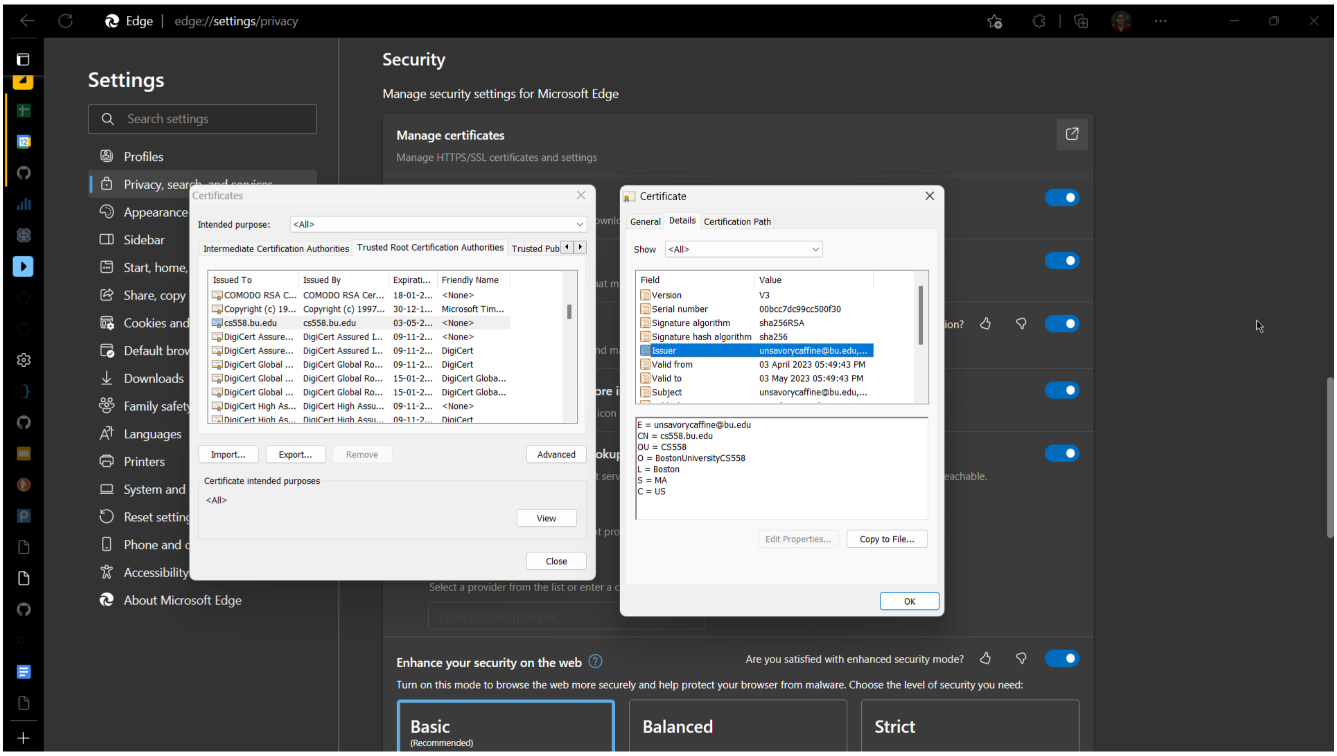
NOTE The sub-domains of `harshkapadia.me` (eg: `networking.harshkapadia.me`) have to be added individually, as the `hosts` file does not support wildcards (`*.harshkapadia.me`).

- To serve an alternate page once the domain is hijacked, create an `index.html` file outside the CA and target domain directories.

```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>vim index.html # Add content shown below
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>cat index.html
<html>
        <head>
                <title>Harsh Kapadia</title>
        </head>

        <body style="text-align: center;">
                <h1>Harsh Kapadia</h1>
        </body>
</html>
```

- The CA certificate (`bucs558-ca.crt`) is a root certificate that is providing trust to the certificate (`harshkapadia.me.crt`) that we created for the target domain (`harshkapadia.me`). The host OS and browser doesn't trust the CA root certificate (`bucs558-ca.crt`) though, so the domain certificate (`harshkapadia.me.crt`) won't be accepted. So, the root certificate (`bucs558-ca.crt`) needs to be added to the OS's certificate store manually. Adding the root certificate separately to the browser's key store might also be required (like in Mozilla Firefox, as shown below).


Edge | edge://settings/privacy
Settings
Search settings
Profiles
Privacy, search, and services
Appearance
Sidebar
Start, home,
Share, copy
Cookies and
Default brov
Downloads
Family safety
Languages
Printers
System and
Reset setting
Phone and
Accessibility
About Microsoft Edge
Security
Manage security settings for Microsoft Edge
Manage certificates
Manage HTTPS/SSL certificates and settings
Enhance your security on the web
Are you satisfied with enhanced security mode?
Turn on this mode to browse the web more securely and help protect your browser from malware. Choose the level of security you need:
Basic
(Recommended)
Balanced
Strict

Certificates

Intended purpose: <All>

Intermediate Certification Authorities | Trusted Root Certification Authorities | Trusted Pub

Issued To	Issued By	Expirati...	Friendly Name
COMODO RSA C...	COMODO RSA Cer...	18-01-2...	<None>
Copyright (c) 19...	Copyright (c) 1997...	30-12-1...	Microsoft Tim...
cs558.bu.edu	cs558.bu.edu	03-05-2...	<None>
DigiCert Assure...	DigiCert Assured I...	09-11-2...	<None>
DigiCert Assure...	DigiCert Assured I...	09-11-2...	DigiCert
DigiCert Global ...	DigiCert Global Ro...	09-11-2...	DigiCert
DigiCert Global ...	DigiCert Global Ro...	15-01-2...	DigiCert Globa...
DigiCert Global ...	DigiCert Global Ro...	15-01-2...	DigiCert Globa...
DigiCert High As...	DigiCert High Assu...	09-11-2...	<None>
DigiCert High As...	DigiCert High Assu...	09-11-2...	DigiCert

Import... Export... Remove Advanced
Certificate intended purposes
<All>
View
Close

Certificate

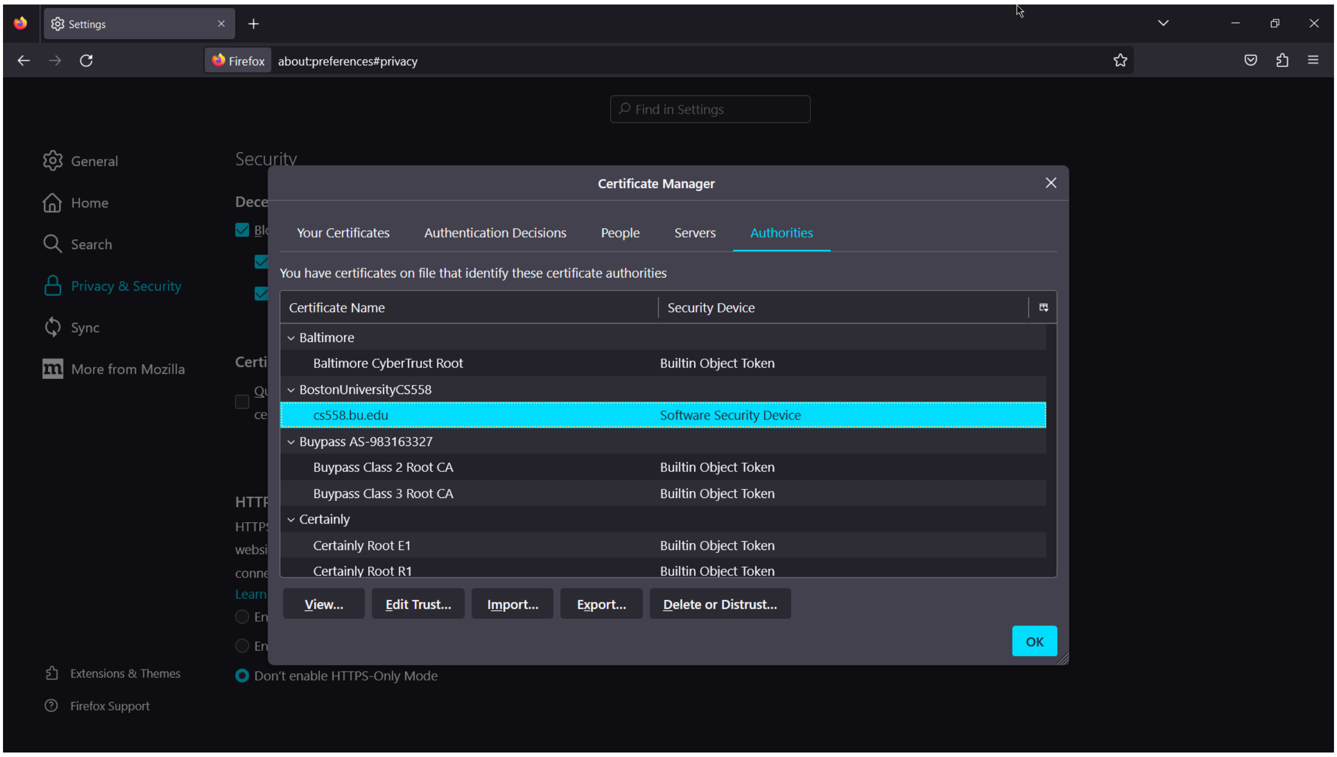
General | Details | Certification Path

Show <All>

Field	Value
Version	V3
Serial number	00bcc7dc99cc500f30
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	unsavorycaffine@bu.edu,...
Valid from	03 April 2023 05:49:43 PM
Valid to	03 May 2023 05:49:43 PM
Subject	unsavorycaffine@bu.edu,...

```
E = unsavorycaffine@bu.edu
CN = cs558.bu.edu
OU = CS558
O = BostonUniversityCS558
L = Boston
S = MA
C = US
```

Microsoft Windows' Certificate Store



Certificate Manager

Your Certificates | Authentication Decisions | People | Servers | Authorities

You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
Baltimore	
Baltimore CyberTrust Root	Builtin Object Token
BostonUniversityCS558	
cs558.bu.edu	Software Security Device
Buypass AS-983163327	
Buypass Class 2 Root CA	Builtin Object Token
Buypass Class 3 Root CA	Builtin Object Token
Certainly	
Certainly Root E1	Builtin Object Token
Certainly Root R1	Builtin Object Token

View... Edit Trust... Import... Export... Delete or Distrust...
OK

Mozilla Firefox’s Certificate Store

- To be able to carry out the attack, there has to be a TLS terminating server (`openssl s_server`) running that captures the request to the target domain (`harshkapadia.me`) that was redirected to `127.0.0.1` (`localhost`) by the OS due to the `host` file configuration above.
    - The server is listening at port 443, as that is the default port for HTTPS requests.

NOTE The command below has to be run from the directory where the `index.html` file was created.

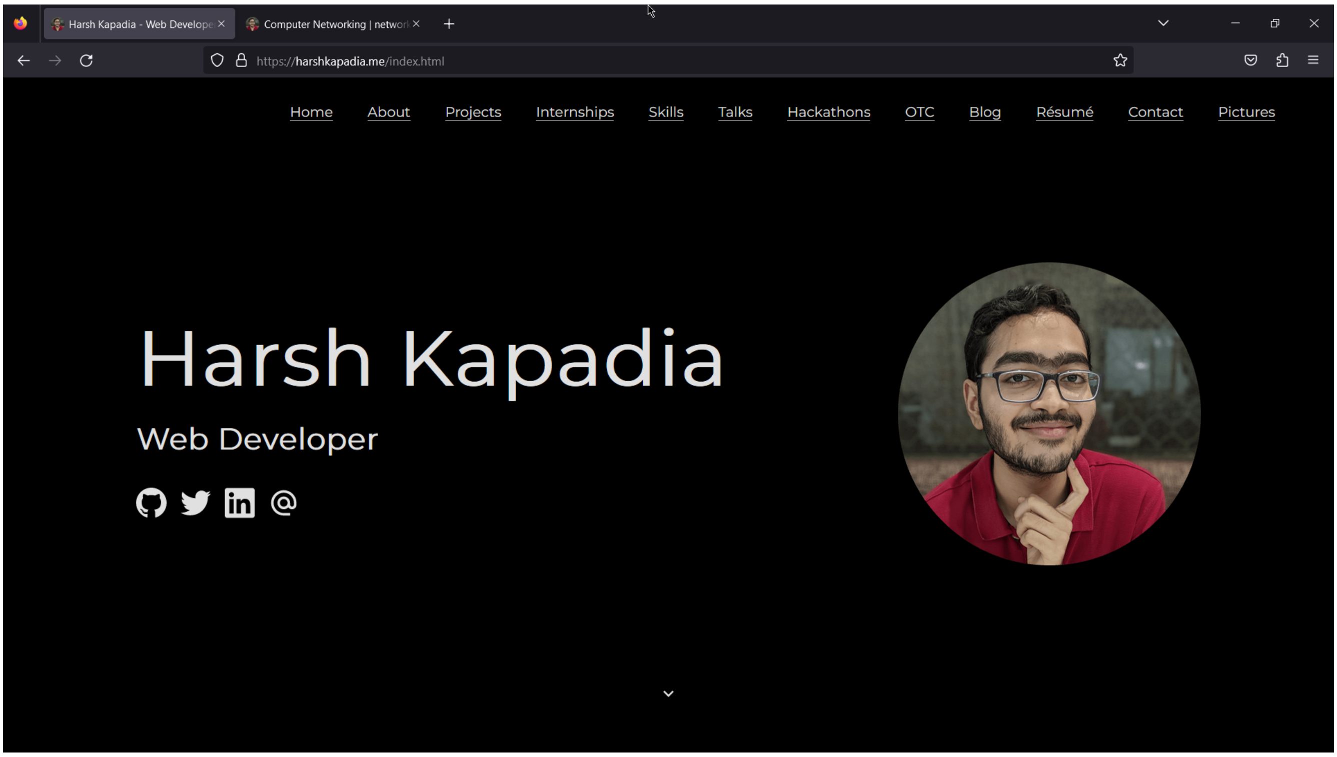
```
X:\ms-bu\sem-2\bu-cas-cs-558\cert-hw>openssl s_server -accept 127.0.0.1:443 -cert
./harshkapadia.me/harshkapadia.me.crt -cert_chain ./bucs558-ca/bucs558-ca.crt -key
./harshkapadia.me/harshkapadia.me.key -WWW
Using default temp DH parameters
ACCEPT
FILE:index.html
FILE:index.html
FILE:index.html
FILE:index.html
```

Accessing the Target Domain

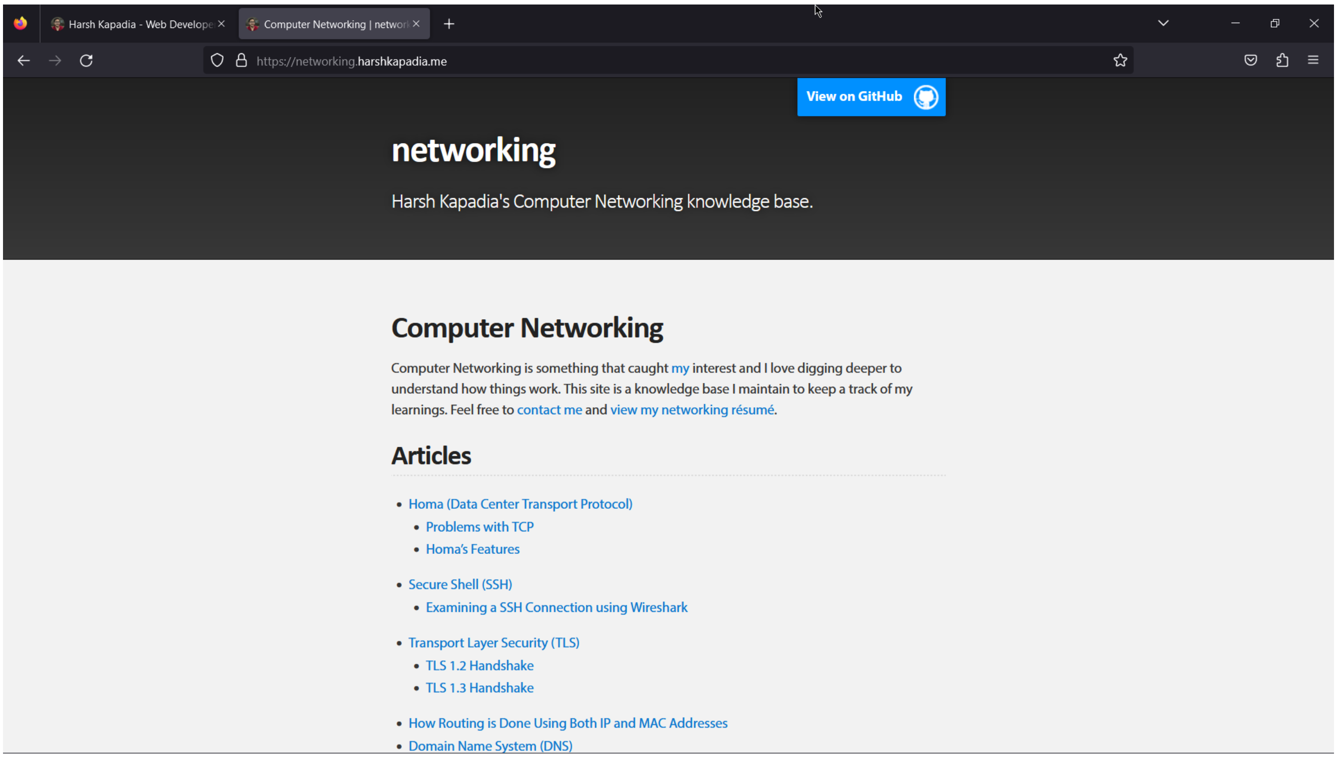
- Now on going to `https://harshkapadia.me/index.html` or `https://networking.harshkapadia.me/index.html`, instead of seeing the original web site, the highjacked `index.html` file is visible with a lock beside the URL, which means that the browser accepted the phony certificates that we presented it. Thus, the browser has been tricked.

Original Web Pages

This is how the two web pages looked before hijacking:



Base domain: `https://harshkapadia.me/index.html`


Harsh Kapadia - Web Develope
Computer Networking | networ
https://networking.harshkapadia.me
View on GitHub

networking

Harsh Kapadia's Computer Networking knowledge base.

Computer Networking

Computer Networking is something that caught my interest and I love digging deeper to understand how things work. This site is a knowledge base I maintain to keep a track of my learnings. Feel free to contact me and view my networking résumé.

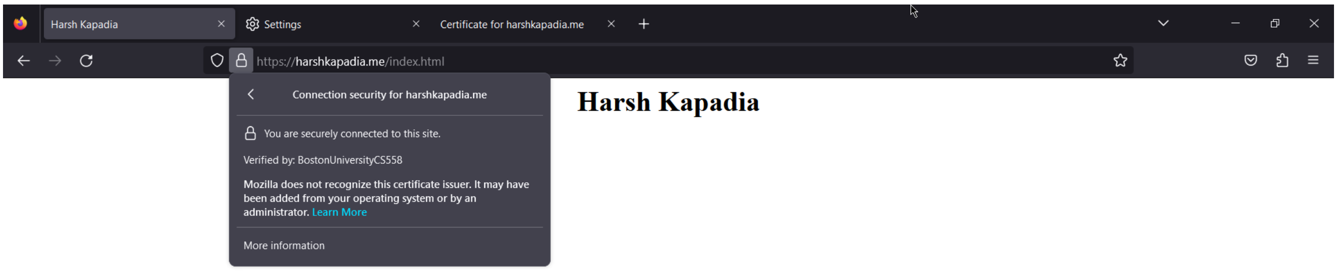
Articles

- Homa (Data Center Transport Protocol)
  - Problems with TCP
  - Homa's Features
- Secure Shell (SSH)
  - Examining a SSH Connection using Wireshark
- Transport Layer Security (TLS)
  - TLS 1.2 Handshake
  - TLS 1.3 Handshake
- How Routing is Done Using Both IP and MAC Addresses
- Domain Name System (DNS)

Sub domain: `https://networking.harshkapadia.me/index.html`

Mozilla Firefox

After the hijacking:


Harsh Kapadia
Settings
Certificate for harshkapadia.me
https://harshkapadia.me/index.html
More information

Connection security for harshkapadia.me

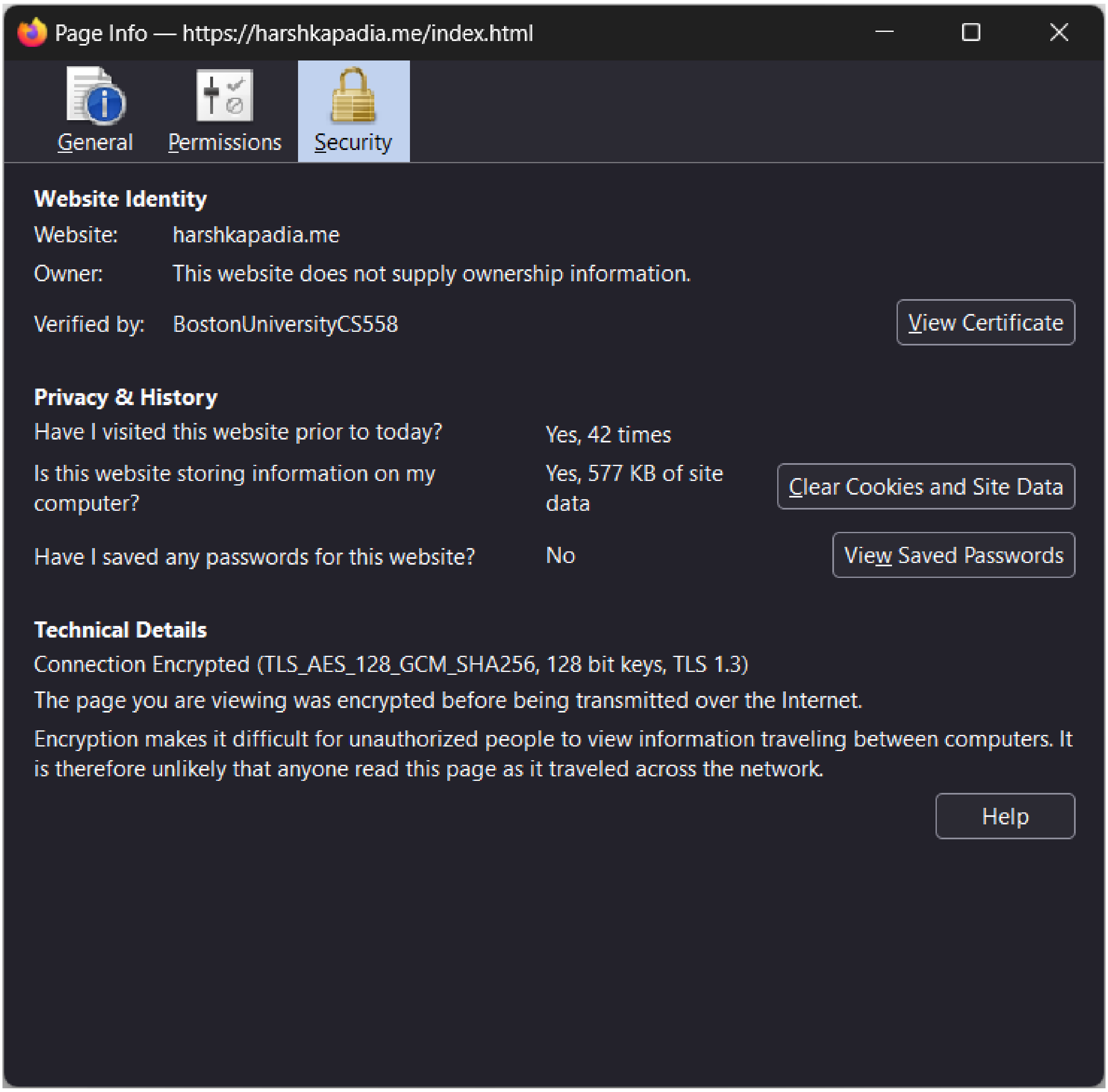
You are securely connected to this site.

Verified by: BostonUniversityCS558

Mozilla does not recognize this certificate issuer. It may have been added from your operating system or by an administrator. Learn More

Harsh Kapadia

Base domain: `https://harshkapadia.me/index.html`



Page Info — https://harshkapadia.me/index.html
General
Permissions
Security

Website Identity

Website: harshkapadia.me

Owner: This website does not supply ownership information.

Verified by: BostonUniversityCS558

Privacy & History

Have I visited this website prior to today? Yes, 42 times

Is this website storing information on my computer? Yes, 577 KB of site data

Clear Cookies and Site Data

Have I saved any passwords for this website? No

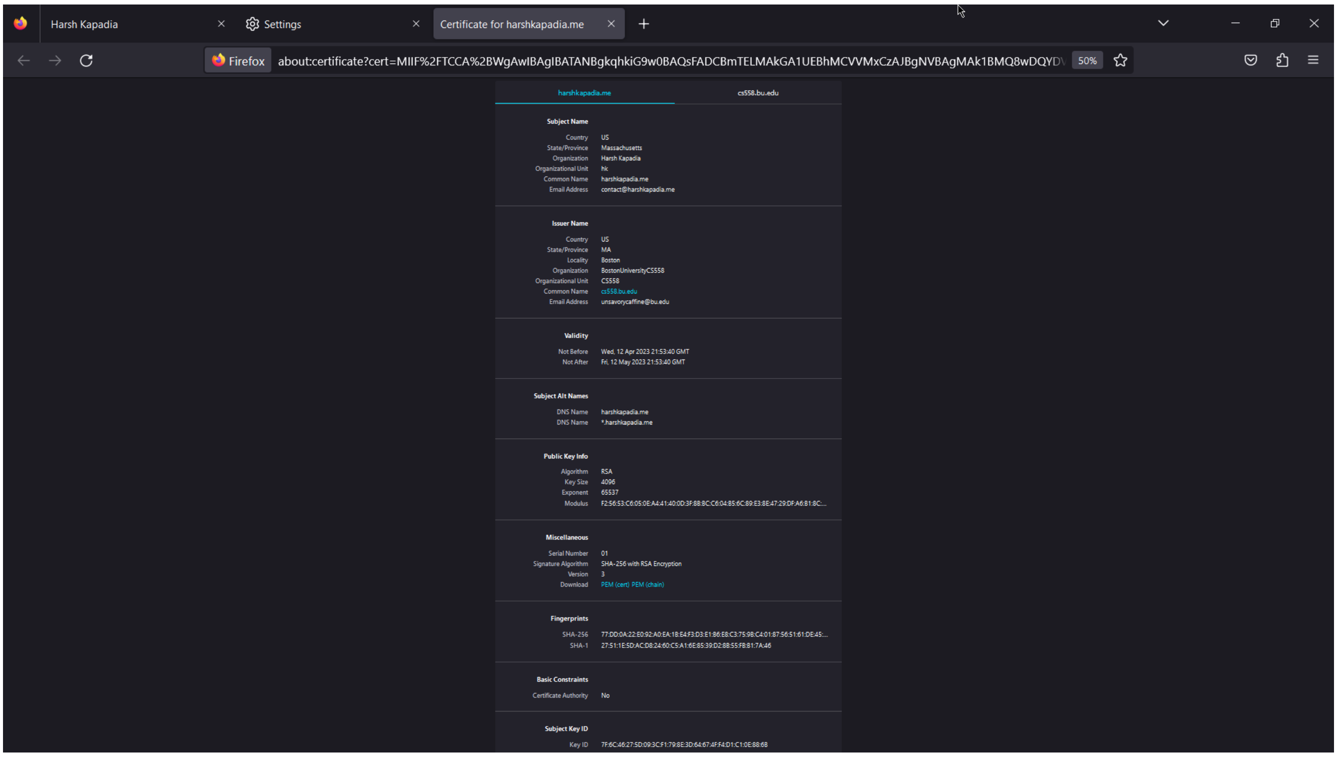
Technical Details

Connection Encrypted (TLS_AES_128_GCM_SHA256, 128 bit keys, TLS 1.3)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Firefox's site security dialog box


Harsh Kapadia
Settings
Certificate for harshkapadia.me
Firefox
about:certificate?cert=MIIF%2FTCCA%2BWgAwIBAgIBATANBgkqhkiG9w0BAQsFADCBmTELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAk1BMQ8wDQYD
50%
harshkapadia.me
cs558.bu.edu

Subject Name

Country	US
State/Province	Massachusetts
Organization	Harsh Kapadia
Organizational Unit	hk
Common Name	harshkapadia.me
Email Address	contact@harshkapadia.me

Issuer Name

Country	US
State/Province	MA
Locality	Boston
Organization	BostonUniversityCS558
Organizational Unit	CS558
Common Name	cs558.bu.edu
Email Address	unsavorycaffine@bu.edu

Validity

Not Before	Wed, 12 Apr 2023 21:53:40 GMT
Not After	Fri, 12 May 2023 21:53:40 GMT

Subject Alt Names

DNS Name	harshkapadia.me
DNS Name	*.harshkapadia.me

Public Key Info

Algorithm	RSA
Key Size	4096
Exponent	65537
Modulus	F2:56:53:C6:05:0E:A4:41:40:0D:3F:8B:BC:C6:04:B5:6C:89:E3:8E:47:29:DF:A6:B1:8C:...

Miscellaneous

Serial Number	01
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

Fingerprints

SHA-256	77:DD:0A:22:E0:92:A0:EA:1B:E4:F3:D3:E1:B6:E8:C3:75:9B:C4:01:87:56:51:61:DE:45:...
SHA-1	27:51:1E:5D:AC:D8:24:60:C5:A1:6E:85:39:D2:8B:55:FB:B1:7A:46

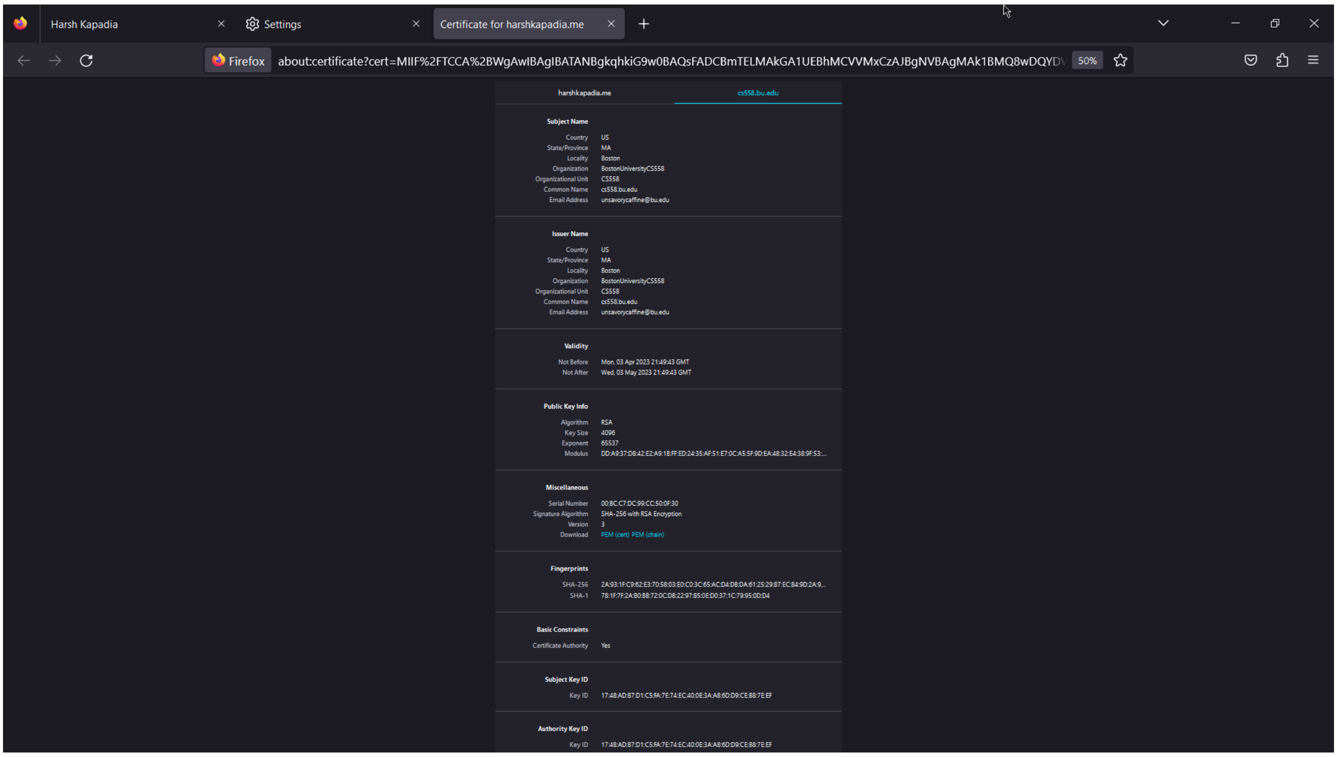
Basic Constraints

Certificate Authority	No

Subject Key ID

Key ID	7F:6C:46:27:5D:09:3C:F1:79:8E:3D:64:67:4F:F4:D1:C1:0E:88:6B

The leaf (domain) certificate as seen in Firefox


Harsh Kapadia
Settings
Certificate for harshkapadia.me
Firefox
about:certificate?cert=MIIF%2FTCCA%2BWgAwIBAgIBATANBgkqhkiG9w0BAQsFADCBmTELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAk1BMQ8wDQYD
50%
harshkapadia.me
cs558.bu.edu

Subject Name

Country	US
State/Province	MA
Locality	Boston
Organization	BostonUniversityCS558
Organizational Unit	CS558
Common Name	cs558.bu.edu
Email Address	unsavorycaffine@bu.edu

Issuer Name

Country	US
State/Province	MA
Locality	Boston
Organization	BostonUniversityCS558
Organizational Unit	CS558
Common Name	cs558.bu.edu
Email Address	unsavorycaffine@bu.edu

Validity

Not Before	Mon, 03 Apr 2023 21:49:43 GMT
Not After	Wed, 03 May 2023 21:49:43 GMT

Public Key Info

Algorithm	RSA
Key Size	4096
Exponent	65537
Modulus	DD:A9:37:D8:42:E2:A9:1B:FF:ED:24:35:AF:51:E7:0C:A5:5F:9D:EA:48:32:E4:38:9F:53:...

Miscellaneous

Serial Number	00:BC:C7:DC:99:CC:50:0F:30
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

Fingerprints

SHA-256	2A:93:1F:C9:62:E3:70:58:03:E0:C0:3C:65:AC:D4:DB:DA:61:25:29:87:EC:84:9D:2A:9...
SHA-1	78:1F:7F:2A:B0:BB:72:0C:D8:22:97:85:0E:D0:37:1C:79:95:0D:D4

Basic Constraints

Certificate Authority	Yes

Subject Key ID

Key ID	17:48:AD:B7:D1:C5:FA:7E:74:EC:40:0E:3A:A8:6D:D9:CE:B8:7E:EF

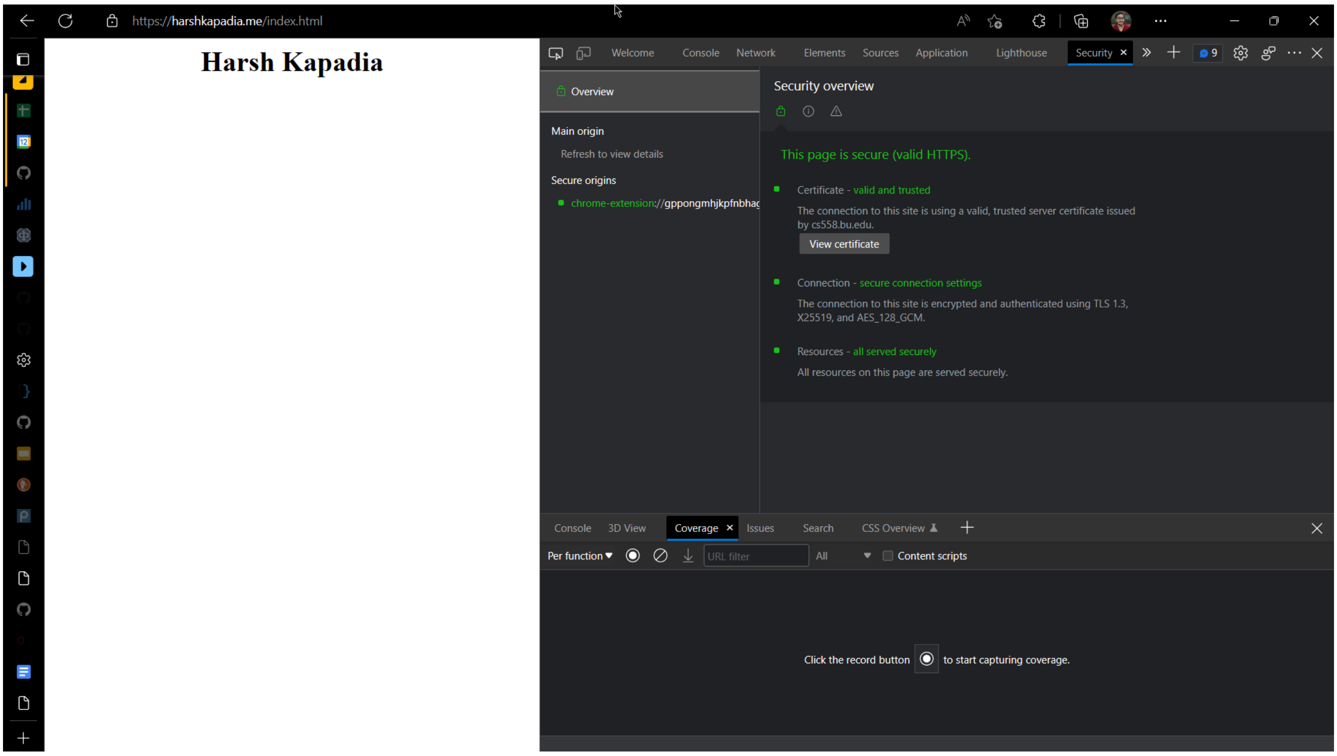
Authority Key ID

Key ID	17:48:AD:B7:D1:C5:FA:7E:74:EC:40:0E:3A:A8:6D:D9:CE:B8:7E:EF

The root (CA) certificate as seen in Firefox

Microsoft Edge

After the hijacking:



Security overview

This page is secure (valid HTTPS).

Certificate - valid and trusted

The connection to this site is using a valid, trusted server certificate issued by cs558.bu.edu.

Connection - secure connection settings

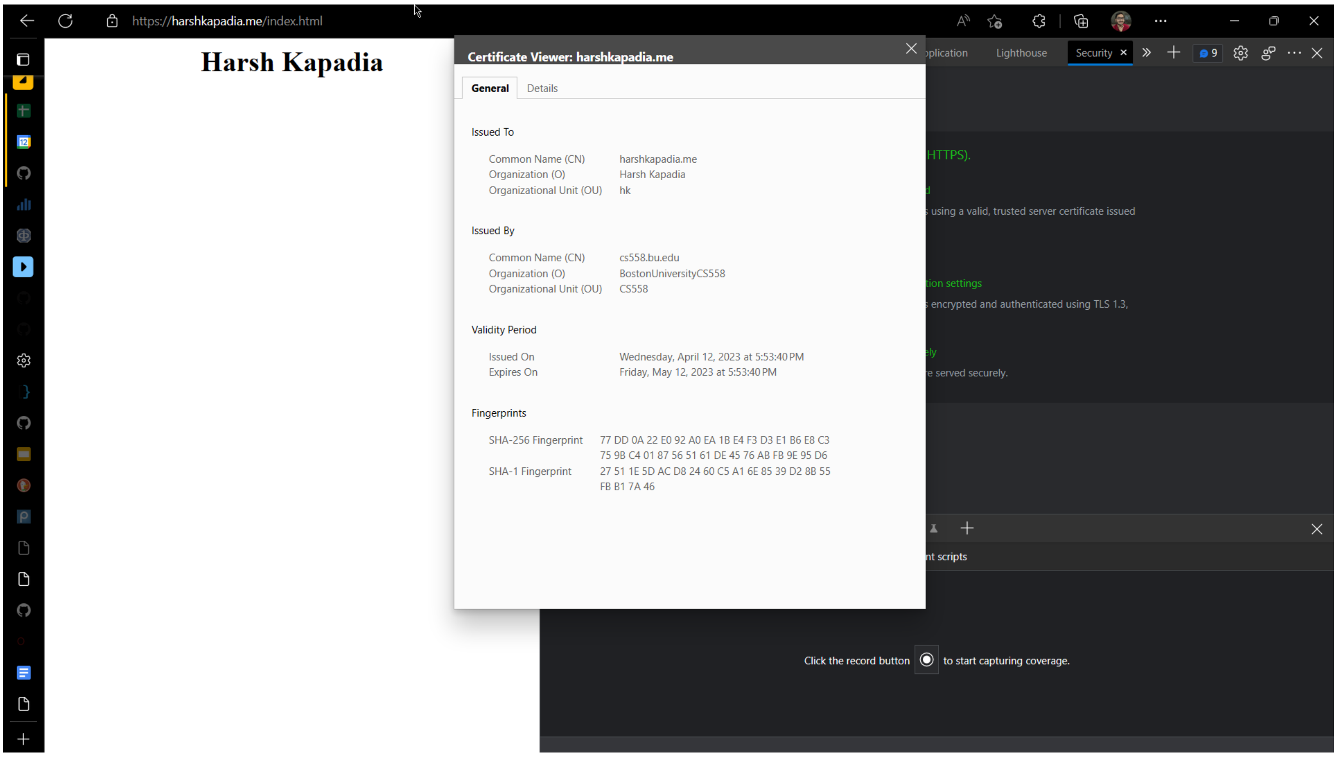
The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.

Resources - all served securely

All resources on this page are served securely.

Click the record button ◉ to start capturing coverage.

Base domain: `https://harshkapadia.me/index.html`



Certificate Viewer: harshkapadia.me

Issued To

Common Name (CN)	harshkapadia.me
Organization (O)	Harsh Kapadia
Organizational Unit (OU)	hk

Issued By

Common Name (CN)	cs558.bu.edu
Organization (O)	BostonUniversityCS558
Organizational Unit (OU)	CS558

Validity Period

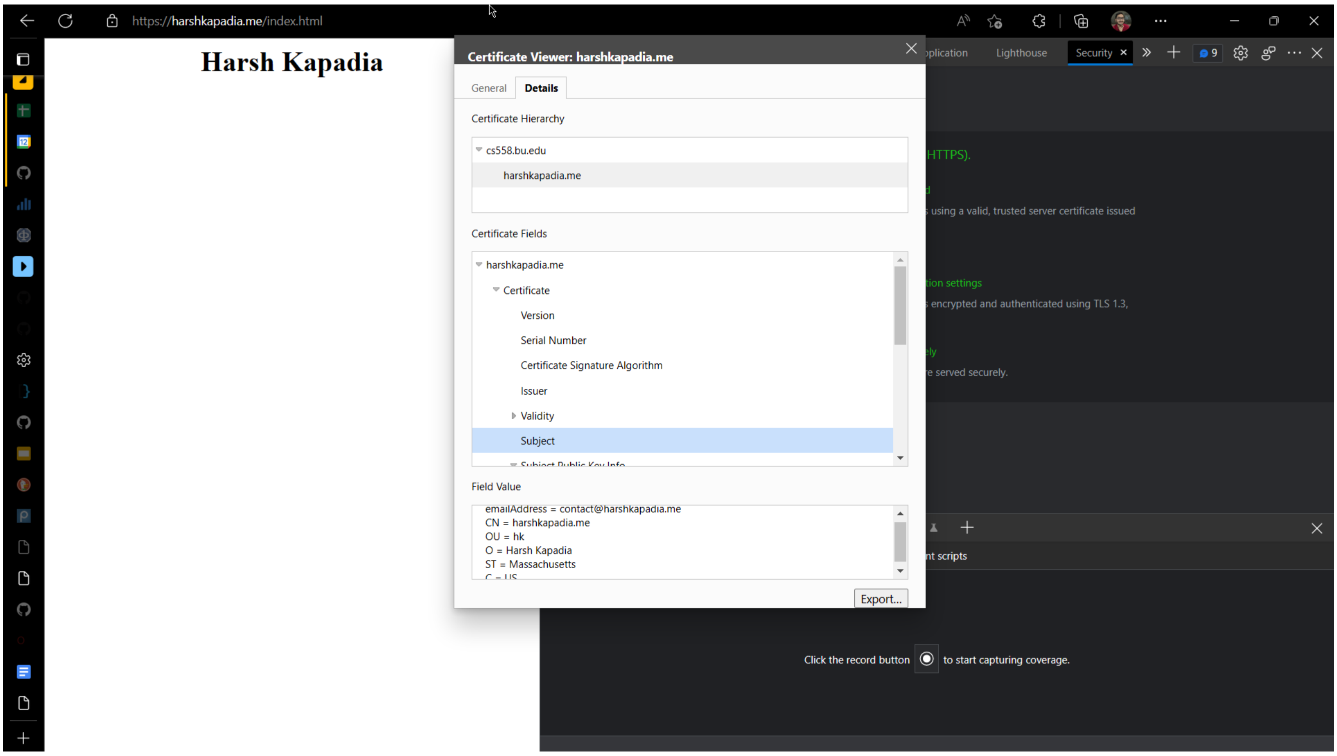
Issued On	Wednesday, April 12, 2023 at 5:53:40 PM
Expires On	Friday, May 12, 2023 at 5:53:40 PM

Fingerprints

SHA-256 Fingerprint	77 DD 0A 22 E0 92 A0 EA 1B E4 F3 D3 E1 B6 E8 C3 75 9B C4 01 87 56 51 61 DE 45 76 AB FB 9E 95 D6
SHA-1 Fingerprint	27 51 1E 5D AC D8 24 60 C5 A1 6E 85 39 D2 8B 55 FB B1 7A 46

Click the record button ◉ to start capturing coverage.

General view of the leaf (domain) certificate as seen in Edge


https://harshkapadia.me/index.html
Harsh Kapadia
Certificate Viewer: harshkapadia.me
General
Details
plication
Lighthouse
Security
Export...

Certificate Hierarchy

- cs558.bu.edu
  - harshkapadia.me

Certificate Fields

- harshkapadia.me
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
    - Subject
    - Subject Public Key Info

Field Value

```
emailAddress = contact@harshkapadia.me
CN = harshkapadia.me
OU = hk
O = Harsh Kapadia
ST = Massachusetts
C = US
```

HTTPS).

using a valid, trusted server certificate issued

tion settings

encrypted and authenticated using TLS 1.3,

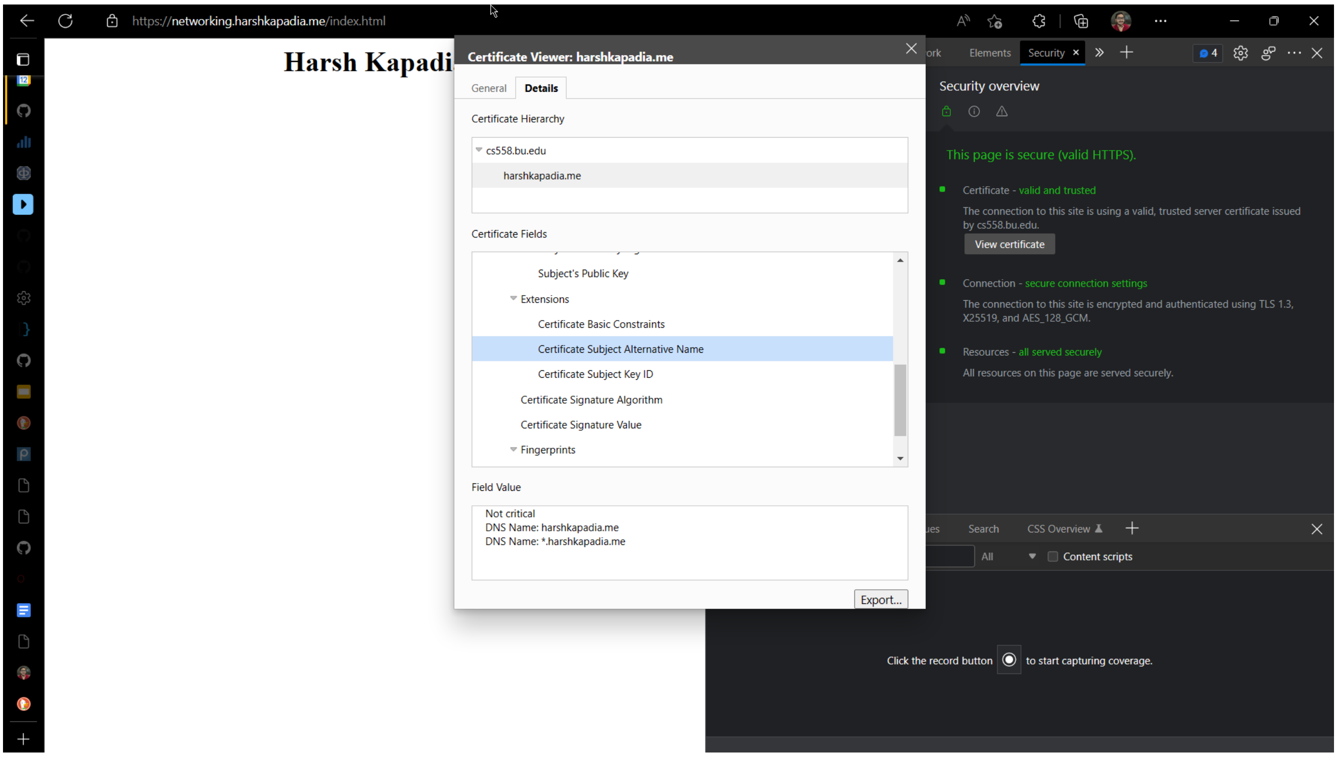
ly

e served securely.

nt scripts

Click the record button ● to start capturing coverage.

Hierarchical view of certificates as seen in Edge



Certificate Hierarchy

- cs558.bu.edu
  - harshkapadia.me

Certificate Fields

- Subject's Public Key
- Extensions
  - Certificate Basic Constraints
  - Certificate Subject Alternative Name
  - Certificate Subject Key ID
- Certificate Signature Algorithm
- Certificate Signature Value
- Fingerprints

Field Value

```
Not critical
DNS Name: harshkapadia.me
DNS Name: *.harshkapadia.me
```

Security overview

This page is secure (valid HTTPS).

- Certificate - valid and trusted

  The connection to this site is using a valid, trusted server certificate issued by cs558.bu.edu.

- Connection - secure connection settings

  The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.

- Resources - all served securely

  All resources on this page are served securely.

□ Content scripts

Click the record button ● to start capturing coverage.

Sub domain: `https://networking.harshkapadia.me/index.html`