Assignment Solution on

Transport Layer Security

Network and System Security (SIL765) |Assignment-5(Part-1) |2024–2025

Submitted by

Arindam Sal

Entry Number: 2024JCS2041

Submitted on: 20th April, 2025

Indian Institute of Technology Delhi

M.Tech in Cyber Security

Contents

Overview

This report presents the implementation and analysis of the Transport Layer Security (TLS) protocol as part of Assignment-5(Part-1). The goal is to understand the inner workings of the TLS handshake, certificate verification, hostname validation, and secure communication.

The implementation was done in Python using the built-in `ssl` and `socket` modules, tested against real-world HTTPS servers like `google.com`. Screenshots and observations from Wireshark and terminal outputs are included.

1 Task 1: TLS Handshake

Execution Instructions

1. Open terminal and run the TLS client:

   ```
   python3 tls_client.py google.com
   ```

2. When the prompt `TCP connection established.  Press Enter to continue...` appears, open Wireshark.

3. Start a packet capture with a filter like:

   ```
   ip.addr == <your_IP> && tcp.port == 443
   ```

4. Return to the terminal and press Enter to initiate the TLS handshake.

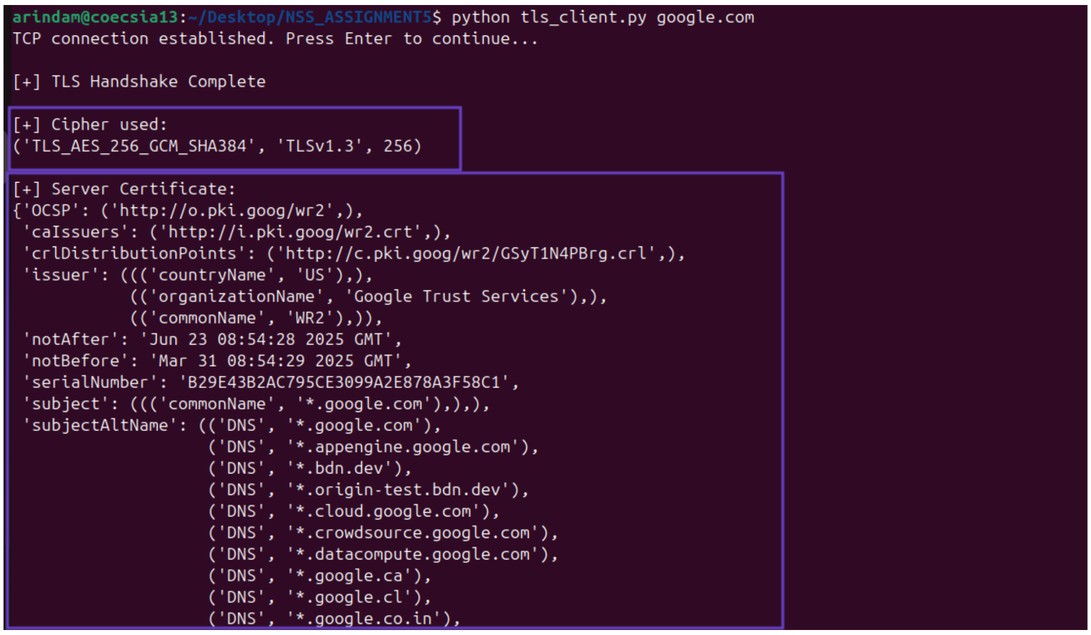
5. Observe the terminal output. It will show the cipher suite negotiated and the server certificate details.

6. Stop the Wireshark capture. Locate:
   - The 3-way TCP handshake (SYN, SYN-ACK, ACK)
   - The TLS handshake (Client Hello, Server Hello, Certificate, etc.)

7. Save a screenshot of the terminal output and a Wireshark capture segment showing both handshakes.

Observations and Analysis

- **Cipher Suite**: The terminal displays a tuple like `('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)` indicating the encryption algorithm, protocol version, and key length used in the session. This confirms successful negotiation.
- **Server Certificate**: The certificate chain of the server is printed. It shows issuer details, subject (e.g., `*.google.com`), validity period, and SANs (Subject Alternative Names).
- **Wireshark View**: The capture shows:
  - Initial TCP handshake: SYN $\rightarrow$, SYN-ACK $\leftarrow$, ACK $\rightarrow$
  - TLS handshake messages: `Client Hello`, `Server Hello`, `Certificate`, `Server Finished`, `Client Finished`
- These layers show that TLS relies on an underlying TCP connection. TLS handshake begins only after a successful TCP 3-way handshake.
- This validates the secure channel is built only after establishing a reliable transport path.



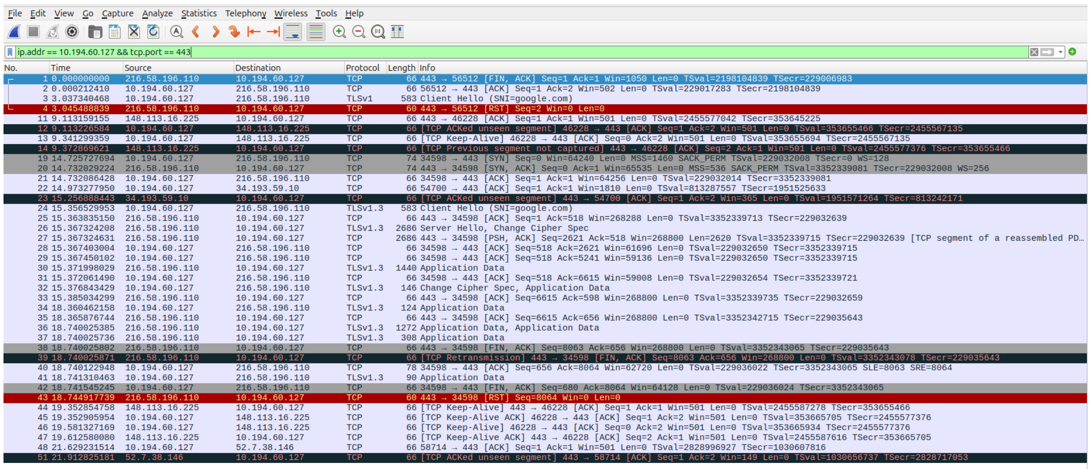
```
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ python tls_client.py google.com
TCP connection established. Press Enter to continue...

[+] TLS Handshake Complete

[+] Cipher used:
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)

[+] Server Certificate:
{'OCSP': ('http://o.pki.goog/wr2',),
 'caIssuers': ('http://i.pki.goog/wr2.crt',),
 'crlDistributionPoints': ('http://c.pki.goog/wr2/GSyT1N4PBrg.crl',),
 'issuer': ((('countryName', 'US'),),
           (('organizationName', 'Google Trust Services'),),
           (('commonName', 'WR2'),)),
 'notAfter': 'Jun 23 08:54:28 2025 GMT',
 'notBefore': 'Mar 31 08:54:29 2025 GMT',
 'serialNumber': 'B29E43B2AC795CE3099A2E878A3F58C1',
 'subject': ((('commonName', '*.google.com'),),),
 'subjectAltName': (('DNS', '*.google.com'),
                   ('DNS', '*.appengine.google.com'),
                   ('DNS', '*.bdn.dev'),
                   ('DNS', '*.origin-test.bdn.dev'),
                   ('DNS', '*.cloud.google.com'),
                   ('DNS', '*.crowdsource.google.com'),
                   ('DNS', '*.datacompute.google.com'),
                   ('DNS', '*.google.ca'),
                   ('DNS', '*.google.cl'),
                   ('DNS', '*.google.co.in'),
```

Figure 1: Terminal output showing cipher used and server certificate received from google.com



ip.addr == 10.194.60.127 && tcp.port == 443

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	216.58.196.110	10.194.60.127	TCP	66	443 → 56512 [FIN, ACK] Seq=1 Ack=1 Win=1050 Len=0 TSval=2198104839 TSecr=229006983
2	0.000212410	10.194.60.127	216.58.196.110	TCP	66	56512 → 443 [ACK] Seq=1 Ack=2 Win=502 Len=0 TSval=229017283 TSecr=2198104839
3	3.037340468	10.194.60.127	216.58.196.110	TLSv1	583	Client Hello (SNI=google.com)
4	3.045488839	216.58.196.110	10.194.60.127	TCP	60	443 → 56512 [RST] Seq=2 Win=0 Len=0
11	9.113159155	148.113.16.225	10.194.60.127	TCP	66	443 → 46228 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2455577042 TSecr=353645225
12	9.113226584	10.194.60.127	148.113.16.225	TCP	66	[TCP ACKed unseen segment] 46228 → 443 [ACK] Seq=1 Ack=2 Win=501 Len=0 TSval=353655466 TSecr=2455567135
13	9.341299359	10.194.60.127	148.113.16.225	TCP	66	[TCP Keep-Alive] 46228 → 443 [ACK] Seq=0 Ack=2 Win=501 Len=0 TSval=353655694 TSecr=2455567135
14	9.372869621	148.113.16.225	10.194.60.127	TCP	66	[TCP Previous segment not captured] 443 → 46228 [ACK] Seq=2 Ack=1 Win=501 Len=0 TSval=2455577376 TSecr=353655466
19	14.725727694	10.194.60.127	216.58.196.110	TCP	74	34598 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=229032008 TSecr=0 WS=128
20	14.732029224	216.58.196.110	10.194.60.127	TCP	74	443 → 34598 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=536 SACK_PERM TSval=3352339081 TSecr=229032008 WS=256
21	14.732086428	10.194.60.127	216.58.196.110	TCP	66	34598 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=229032014 TSecr=3352339081
22	14.973277950	10.194.60.127	34.193.59.10	TCP	66	54700 → 443 [ACK] Seq=1 Ack=1 Win=1810 Len=0 TSval=813287557 TSecr=1951525633
23	15.256888443	34.193.59.10	10.194.60.127	TCP	66	[TCP ACKed unseen segment] 443 → 54700 [ACK] Seq=1 Ack=2 Win=365 Len=0 TSval=1951571264 TSecr=813242171
24	15.356529953	10.194.60.127	216.58.196.110	TLSv1.3	583	Client Hello (SNI=google.com)
25	15.363835150	216.58.196.110	10.194.60.127	TCP	66	443 → 34598 [ACK] Seq=1 Ack=518 Win=268288 Len=0 TSval=3352339713 TSecr=229032639
26	15.367324208	216.58.196.110	10.194.60.127	TLSv1.3	2686	Server Hello, Change Cipher Spec
27	15.367324631	216.58.196.110	10.194.60.127	TCP	2686	443 → 34598 [PSH, ACK] Seq=2621 Ack=518 Win=268800 Len=2620 TSval=3352339715 TSecr=229032639 [TCP segment of a reassembled PD…
28	15.367403004	10.194.60.127	216.58.196.110	TCP	66	34598 → 443 [ACK] Seq=518 Ack=2621 Win=61696 Len=0 TSval=229032650 TSecr=3352339715
29	15.367450102	10.194.60.127	216.58.196.110	TCP	66	34598 → 443 [ACK] Seq=518 Ack=5241 Win=59136 Len=0 TSval=229032650 TSecr=3352339715
30	15.371998029	216.58.196.110	10.194.60.127	TLSv1.3	1440	Application Data
31	15.372061490	10.194.60.127	216.58.196.110	TCP	66	34598 → 443 [ACK] Seq=518 Ack=6615 Win=59008 Len=0 TSval=229032654 TSecr=3352339721
32	15.376843429	10.194.60.127	216.58.196.110	TLSv1.3	146	Change Cipher Spec, Application Data
33	15.385034299	216.58.196.110	10.194.60.127	TCP	66	443 → 34598 [ACK] Seq=6615 Ack=598 Win=268800 Len=0 TSval=3352339735 TSecr=229032659
34	18.360462158	10.194.60.127	216.58.196.110	TLSv1.3	124	Application Data
35	18.365876744	216.58.196.110	10.194.60.127	TCP	66	443 → 34598 [ACK] Seq=6615 Ack=656 Win=268800 Len=0 TSval=3352342715 TSecr=229035643
36	18.740025385	216.58.196.110	10.194.60.127	TLSv1.3	1272	Application Data, Application Data
37	18.740025736	216.58.196.110	10.194.60.127	TLSv1.3	308	Application Data
38	18.740025802	216.58.196.110	10.194.60.127	TCP	66	443 → 34598 [FIN, ACK] Seq=8063 Ack=656 Win=268800 Len=0 TSval=3352343065 TSecr=229035643
39	18.740025871	216.58.196.110	10.194.60.127	TCP	66	[TCP Retransmission] 443 → 34598 [FIN, ACK] Seq=8063 Ack=656 Win=268800 Len=0 TSval=3352343078 TSecr=229035643
40	18.740122948	10.194.60.127	216.58.196.110	TCP	78	34598 → 443 [ACK] Seq=656 Ack=8064 Win=62720 Len=0 TSval=229036022 TSecr=3352343065 SLE=8063 SRE=8064
41	18.741310463	10.194.60.127	216.58.196.110	TLSv1.3	90	Application Data
42	18.741545245	10.194.60.127	216.58.196.110	TCP	66	34598 → 443 [FIN, ACK] Seq=680 Ack=8064 Win=64128 Len=0 TSval=229036024 TSecr=3352343065
43	18.744917739	216.58.196.110	10.194.60.127	TCP	60	443 → 34598 [RST] Seq=8064 Win=0 Len=0
44	19.352854758	148.113.16.225	10.194.60.127	TCP	66	[TCP Keep-Alive] 443 → 46228 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2455587278 TSecr=353655466
45	19.352905954	10.194.60.127	148.113.16.225	TCP	66	[TCP Keep-Alive ACK] 46228 → 443 [ACK] Seq=1 Ack=2 Win=501 Len=0 TSval=353665705 TSecr=2455577376
46	19.581327169	10.194.60.127	148.113.16.225	TCP	66	[TCP Keep-Alive] 46228 → 443 [ACK] Seq=0 Ack=2 Win=501 Len=0 TSval=353665934 TSecr=2455577376
47	19.612580080	148.113.16.225	10.194.60.127	TCP	66	[TCP Keep-Alive ACK] 443 → 46228 [ACK] Seq=2 Ack=1 Win=501 Len=0 TSval=2455587616 TSecr=353665705
48	21.629231514	10.194.60.127	52.7.38.146	TCP	66	58714 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2828996927 TSecr=1030607816
51	21.912825181	52.7.38.146	10.194.60.127	TCP	66	[TCP ACKed unseen segment] 443 → 58714 [ACK] Seq=1 Ack=2 Win=149 Len=0 TSval=1030656737 TSecr=2828717053

Figure 2: Wireshark capture showing TCP 3-way handshake followed by TLS handshake

2 Task 2: CA's Certificate Verification

Execution Instructions

1. Create an empty folder named `certs` inside the project directory:

   ```
   mkdir certs
   ```

2. Run the client with the 'custom' argument to point the certificate directory to `./certs`:

   ```
   python3 tls_client.py google.com custom
   ```

3. Observe the output: TLS handshake fails with an error indicating that the client is unable to verify the server's certificate.

4. Copy the default CA certificates from the system directory into the local `certs/` folder:

   ```
   sudo cp -r /etc/ssl/certs/* ./certs/
   ```

5. Rerun the client with the same command:

   ```
   python3 tls_client.py google.com custom
   ```

6. This time, observe that the TLS handshake completes successfully, and the server certificate is printed.

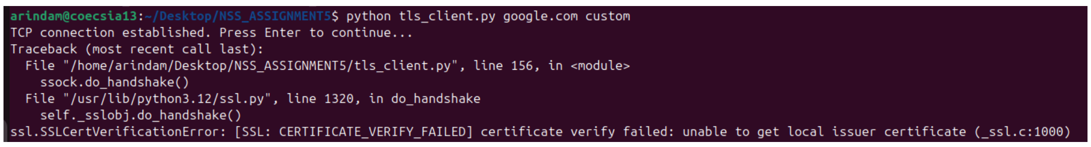
Observations and Analysis

- **Initial Error:** When the `certs/` directory is empty, the client fails to validate the server's certificate because it lacks the root/intermediate CA certificates needed for trust verification. This results in an error:

  ```
  ssl.SSLCertVerificationError: certificate verify failed: unable to
  get local issuer certificate
  ```

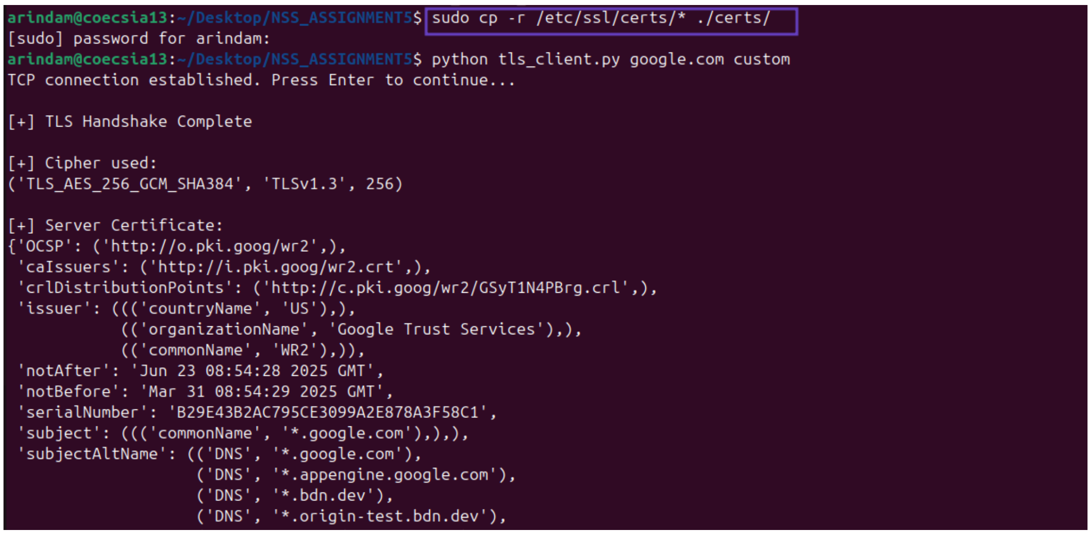
- **After Copying Certificates:** Once valid certificates are added, the server certificate is successfully validated. The TLS handshake completes and data can be securely exchanged.

- This experiment highlights the importance of trusted Certificate Authorities (CAs) in verifying the identity of the server.



```
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ python tls_client.py google.com custom
TCP connection established. Press Enter to continue...
Traceback (most recent call last):
  File "/home/arindam/Desktop/NSS_ASSIGNMENT5/tls_client.py", line 156, in <module>
    ssock.do_handshake()
  File "/usr/lib/python3.12/ssl.py", line 1320, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1000)
```

Figure 3: TLS handshake failure due to empty CA certs folder



```
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ sudo cp -r /etc/ssl/certs/* ./certs/
[sudo] password for arindam:
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ python tls_client.py google.com custom
TCP connection established. Press Enter to continue...

[+] TLS Handshake Complete

[+] Cipher used:
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)

[+] Server Certificate:
{'OCSP': ('http://o.pki.goog/wr2',),
 'caIssuers': ('http://i.pki.goog/wr2.crt',),
 'crlDistributionPoints': ('http://c.pki.goog/wr2/GSyT1N4PBrg.crl',),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', 'Google Trust Services'),),
            (('commonName', 'WR2'),)),
 'notAfter': 'Jun 23 08:54:28 2025 GMT',
 'notBefore': 'Mar 31 08:54:29 2025 GMT',
 'serialNumber': 'B29E43B2AC795CE3099A2E878A3F58C1',
 'subject': ((('commonName', '*.google.com'),),),
 'subjectAltName': (('DNS', '*.google.com'),
                    ('DNS', '*.appengine.google.com'),
                    ('DNS', '*.bdn.dev'),
                    ('DNS', '*.origin-test.bdn.dev'),
```

Figure 4: TLS handshake success after copying CA certs into custom folder

3 Task 3: Hostname Verification

Execution Instructions

1. Use the following command to get the IP address of the server (e.g., google.com):

   ```
   dig google.com +short
   ```

2. Open the system hosts file with root privileges:

   ```
   sudo nano /etc/hosts
   ```

3. Append a line at the end of the file mapping the IP to a fake hostname:

   ```
   216.58.196.110 anything.com
   ```

4. Set the following in your script:

   ```
   context.check_hostname = True
   ```

   Then run the client:

   ```
   python3 tls_client.py anything.com
   ```

   Expected: TLS handshake should fail with hostname mismatch.

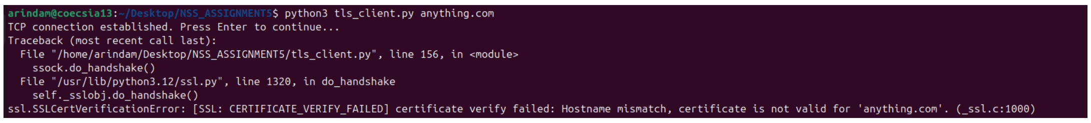
5. Now switch to:

   ```
   context.check_hostname = False
   ```

   and rerun the same command. This time the TLS handshake should succeed.

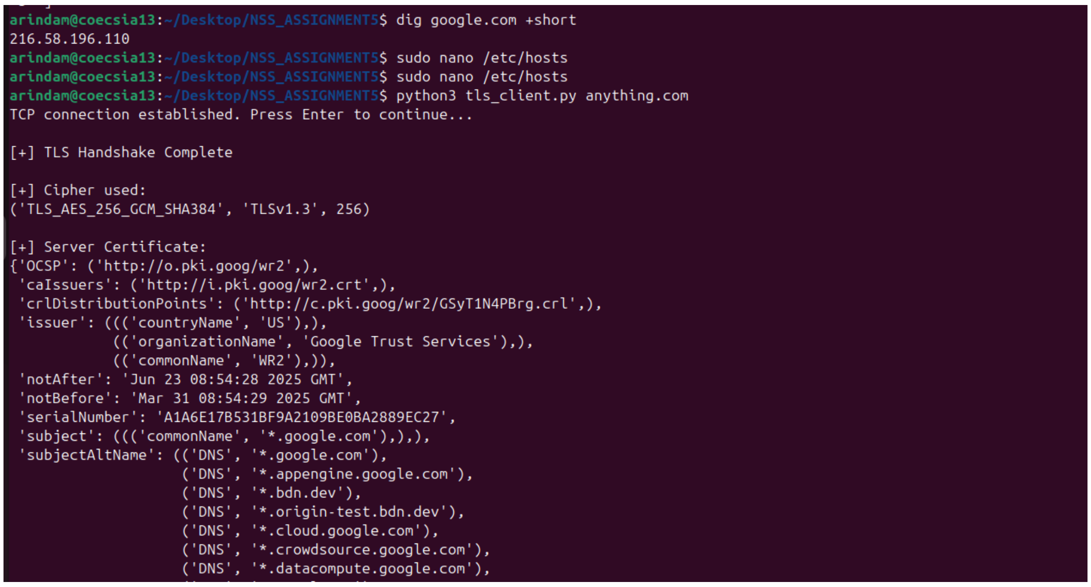
Observations and Analysis

- **Hostname Validation (True):** The TLS handshake fails because the server certificate does not list ‘anything.com‘ in its Common Name (CN) or Subject Alternative Names (SAN). This protects against DNS spoofing and man-in-the-middle (MITM) attacks.
- **Hostname Validation (False):** The handshake succeeds because the client no longer validates if the certificate subject matches the hostname. This is insecure in practice and only used for testing.
- **Conclusion:** Hostname verification ensures that the server you are connecting to is actually the one it claims to be. Disabling it opens up severe security vulnerabilities.



```
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ python3 tls_client.py anything.com
TCP connection established. Press Enter to continue...
Traceback (most recent call last):
  File "/home/arindam/Desktop/NSS_ASSIGNMENT5/tls_client.py", line 156, in <module>
    ssock.do_handshake()
  File "/usr/lib/python3.12/ssl.py", line 1320, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for 'anything.com'. (_ssl.c:1000)
```

Figure 5: Handshake fails due to hostname mismatch when check_hostname = True



```
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ dig google.com +short
216.58.196.110
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ sudo nano /etc/hosts
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ sudo nano /etc/hosts
arindam@coecsia13:~/Desktop/NSS_ASSIGNMENT5$ python3 tls_client.py anything.com
TCP connection established. Press Enter to continue...

[+] TLS Handshake Complete

[+] Cipher used:
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)

[+] Server Certificate:
{'OCSP': ('http://o.pki.goog/wr2',),
 'caIssuers': ('http://i.pki.goog/wr2.crt',),
 'crlDistributionPoints': ('http://c.pki.goog/wr2/GSyT1N4PBrg.crl',),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', 'Google Trust Services'),),
            (('commonName', 'WR2'),)),
 'notAfter': 'Jun 23 08:54:28 2025 GMT',
 'notBefore': 'Mar 31 08:54:29 2025 GMT',
 'serialNumber': 'A1A6E17B531BF9A2109BE0BA2889EC27',
 'subject': ((('commonName', '*.google.com'),),),
 'subjectAltName': (('DNS', '*.google.com'),
                    ('DNS', '*.appengine.google.com'),
                    ('DNS', '*.bdn.dev'),
                    ('DNS', '*.origin-test.bdn.dev'),
                    ('DNS', '*.cloud.google.com'),
                    ('DNS', '*.crowdsource.google.com'),
                    ('DNS', '*.datacompute.google.com'),
```

Figure 6: Handshake succeeds when check_hostname = False, despite spoofed hostname

4 Task 4: Communicating Data Securely

4.1 Part A: Sending an HTTP Request

Execution Instructions

1. Run the client script with any valid HTTPS hostname, e.g.,

   ```
   python3 tls_client.py google.com
   ```

2. After TLS handshake completes, the client sends an HTTP GET request to '/':

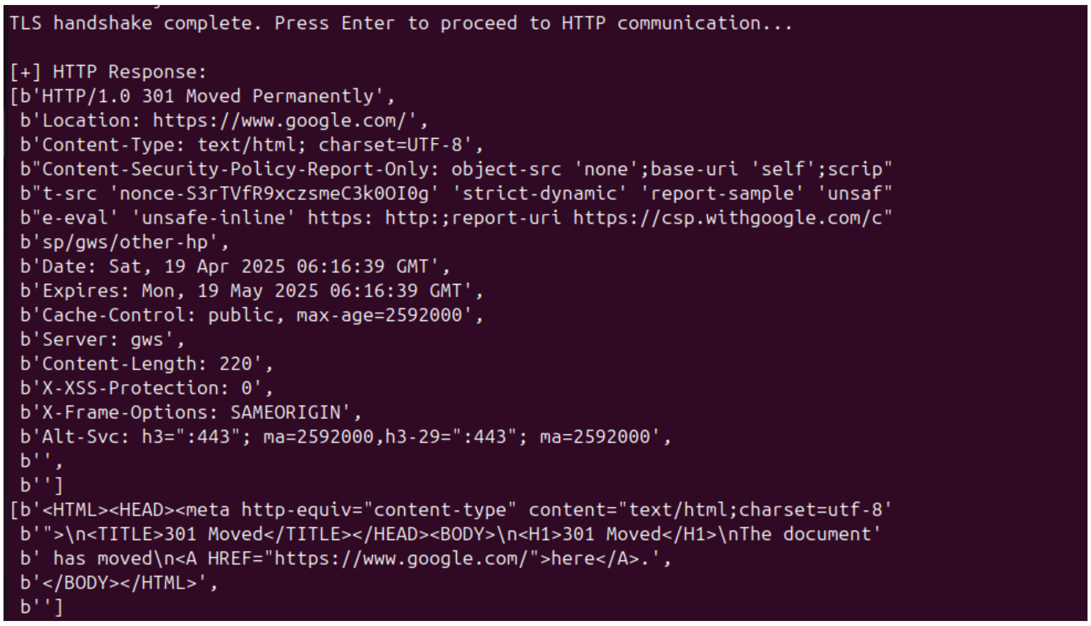
   ```
   GET / HTTP/1.0\r\nHost:  google.com\r\n\r\n
   ```

3. The response is received and printed in the terminal in chunks. The server usually responds with an HTTP 301 redirect or a small HTML page.

4. Observe and log the response headers and body.

Observations and Analysis

- The response confirms that the TLS channel is capable of securely transmitting and receiving application-layer data.
- Response headers like 'Location', 'Content-Type', and 'Date' validate that the server is processing the HTTP request correctly.
- This interaction also simulates what browsers internally perform over HTTPS sessions.



```
TLS handshake complete. Press Enter to proceed to HTTP communication...

[+] HTTP Response:
[b'HTTP/1.0 301 Moved Permanently',
 b'Location: https://www.google.com/',
 b'Content-Type: text/html; charset=UTF-8',
 b"Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';scrip"
 b"t-src 'nonce-S3rTVfR9xczsmeC3k0OI0g' 'strict-dynamic' 'report-sample' 'unsaf"
 b"e-eval' 'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/c"
 b'sp/gws/other-hp',
 b'Date: Sat, 19 Apr 2025 06:16:39 GMT',
 b'Expires: Mon, 19 May 2025 06:16:39 GMT',
 b'Cache-Control: public, max-age=2592000',
 b'Server: gws',
 b'Content-Length: 220',
 b'X-XSS-Protection: 0',
 b'X-Frame-Options: SAMEORIGIN',
 b'Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000',
 b'',
 b'']
[b'<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8'
 b'">\n<TITLE>301 Moved</TITLE></HEAD><BODY>\n<H1>301 Moved</H1>\nThe document'
 b' has moved\n<A HREF="https://www.google.com/">here</A>.',
 b'</BODY></HTML>',
 b'']
```

Figure 7: Terminal output showing response to basic HTTP GET request over TLS

4.2 Part B: Downloading an Image File

Execution Instructions

1. Run the client with additional argument ‘img‘ to trigger image download:

```
python3 tls_client.py www.google.com default img
```

2. The client sends a GET request to a valid image URL, e.g.,

```
GET /images/branding/googlelogo/2x/googlelogo_color_92x30dp.png
HTTP/1.0\r\n
```

3. The binary response is saved as ‘google_logo.png‘ in the working directory.

4. Open the file in an image viewer to confirm correctness.

Observations and Analysis

- TLS supports secure transmission of binary files like images.
- By parsing the HTTP response header and skipping it, we isolate image data before saving it.
- This test mimics a secure download behavior as used in HTTPS-based websites and applications.

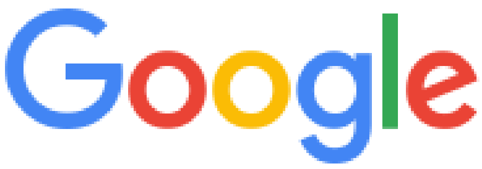


Figure 8: Image fetched over TLS from Google and saved locally

Assignment Solution on

Website Security Analysis

Network and System Security (SIL765) |Assignment-5(Problem-2) |2024–2025

Submitted by

Arindam Sal

Entry Number: 2024JCS2041

Submitted on: 20th April, 2025

Indian Institute of Technology Delhi

M.Tech in Cyber Security

Contents

Part 1: Vulnerability Testing Using Two Tools

Tool 1: Burp Suite (Community Edition)

Burp Suite is a proxy-based tool that enables real-time interception, modification, and analysis of HTTP and HTTPS traffic between a web browser and a web server. It is primarily used for web application security testing.

Vulnerability 1: Cross-Site Scripting (XSS)

Objective: To check if the application sanitizes user input passed via query parameters.
Step:

- Navigated to `library.iitd.ac.in`.
- Injected the payload `<script>alert(1)</script>` in the search bar.
- Observed the response in Burp Suite.

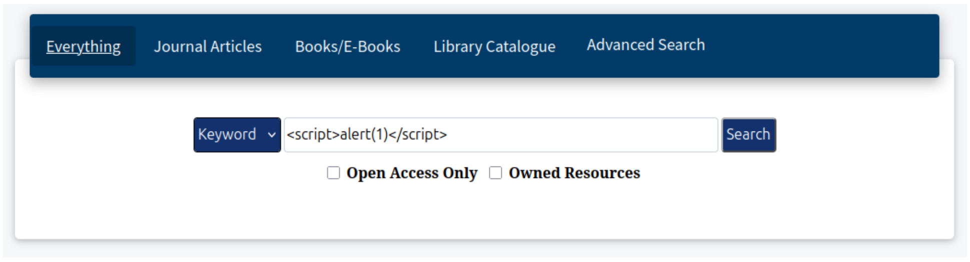
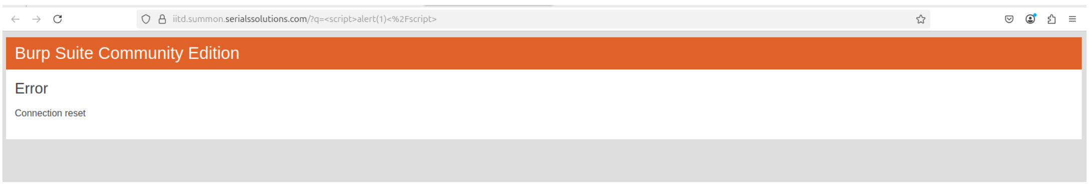


Figure 1: Attempting XSS injection in search field

Result: The site returned a "Connection Reset" error indicating a backend filter or Web Application Firewall (WAF) blocked the request.


iitd.summon.serialssolutions.com/?q=<script>alert(1)<%2Fscript>

Burp Suite Community Edition

Error

Connection reset

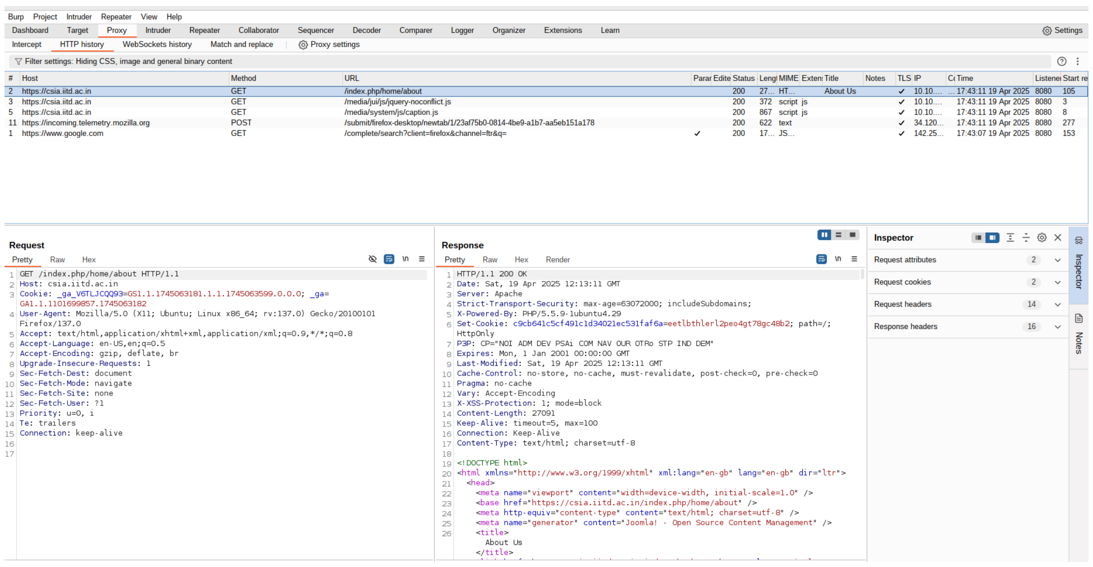
Figure 2: Error response after XSS payload submission

Rationale: XSS attacks allow attackers to execute malicious scripts in users' browsers, leading to session hijacking, phishing, or defacement.

Vulnerability 2: Security Misconfiguration

Step:

- Accessed the About page of `csia.iitd.ac.in`.
- Used Burp to inspect HTTP response headers.


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS, image and general binary content
Inspector
Request attributes 2
Request cookies 2
Request headers 14
Response headers 16

#	Host	Method	URL	Params	Edited	Status	Length	MIME	Extension	Title	Notes	TLS	IP	Time	Listener	Start response
2	https://csia.iitd.ac.in	GET	/index.php/home/about			200	27…	HT…		About Us		✓	10.10.…	17:43:11 19 Apr 2025	8080	105
3	https://csia.iitd.ac.in	GET	/media/jui/js/jquery-noconflict.js			200	372	script	js			✓	10.10.…	17:43:11 19 Apr 2025	8080	3
5	https://csia.iitd.ac.in	GET	/media/system/js/caption.js			200	867	script	js			✓	10.10.…	17:43:11 19 Apr 2025	8080	8
11	https://incoming.telemetry.mozilla.org	POST	/submit/firefox-desktop/newtab/1/23af75b0-0814-4be9-a1b7-aa5eb151a178			200	622	text				✓	34.120…	17:43:11 19 Apr 2025	8080	277
1	https://www.google.com	GET	/complete/search?client=firefox&channel=ftr&q=	✓		200	17…	JS…				✓	142.25…	17:43:07 19 Apr 2025	8080	153

Request

```
GET /index.php/home/about HTTP/1.1
Host: csia.iitd.ac.in
Cookie: _ga_V6TLJCQQ93=GS1.1.1745063181.1.1.1745063599.0.0.0; _ga=
GA1.1.1101699857.1745063182
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:137.0) Gecko/20100101
Firefox/137.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 19 Apr 2025 12:13:11 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains;
X-Powered-By: PHP/5.5.9-1ubuntu4.29
Set-Cookie: c9cb641c5cf491c1d34021ec531faf6a=eetlbthlerl2peo4gt78gc48b2; path=/;
HttpOnly
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Last-Modified: Sat, 19 Apr 2025 12:13:11 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
X-XSS-Protection: 1; mode=block
Content-Length: 27091
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb" lang="en-gb" dir="ltr">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <base href="https://csia.iitd.ac.in/index.php/home/about" />
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta name="generator" content="Joomla! - Open Source Content Management" />
    <title>
      About Us
    </title>
```

Figure 3: Security headers examined from server response

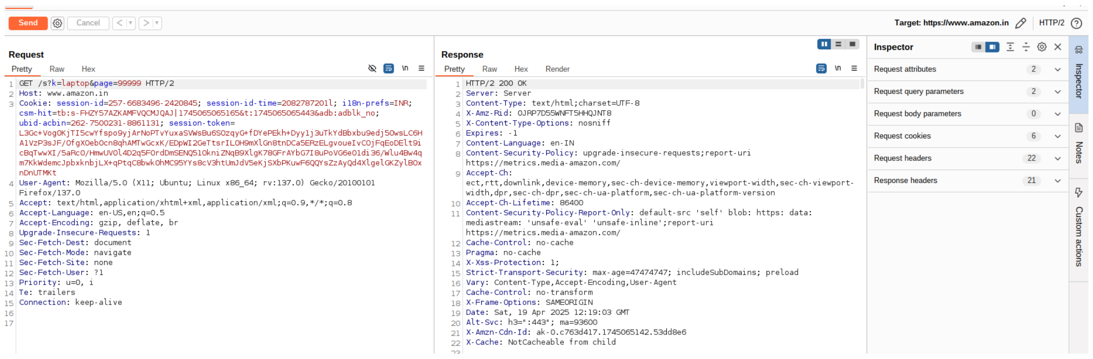
Findings:

- `X-XSS-Protection` was present.
- `Strict-Transport-Security` (HSTS) was set.
- **However**, `Content-Security-Policy` was missing.

Rationale: Without proper CSP headers, modern browsers cannot enforce client-side script restrictions, making XSS harder to prevent.

Vulnerability 3: Parameter Tampering
Step:

- Intercepted URL `https://www.amazon.in/s?k=laptop&page=3`.
- Modified parameter to `page=99999` and forwarded.



Request

```
GET /s?k=laptop&page=99999 HTTP/2
Host: www.amazon.in
Cookie: session-id=257-6683496-2420845; session-id-time=2082787201l; i18n-prefs=INR;
csm-hit=tb:s-FHZY57AZKAMFVQCMJQAJ|1745065065165&t:1745065065443&adb:adblk_no;
ubid-acbin=262-7500231-8861131; session-token=
L3Gc+Vog0KjTI5cwYfspo9yjArNoPTvYuxaSWsBu6SOzqyG+fDYePEkh+Dyy1j3uTkYdBbxbu9edj50wsLC6H
A1VzP3sJF/0fgX0eb0cn8qhAMTwGcxK/EDpWI2GeTtsrILOH9mXlGn8tnDCa5ERzELgvoueIvC0jFqEoDElt9i
cBqTwwXI/5aRc0/HmwUV0l4D2q5F0rdDmSENQ510kniZNqB9XlgK78GFrAYbG7I8uPoVG6e01di36/Wlu4Bw4q
m7KkWdemcJpbxknbjLX+qPtqC8bwk0hMC95YYs8cV3htUmJdV5eKjSXbPKuwF6QQYsZzAyQd4XlgelGKZylBOx
nDnUTMKt
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:137.0) Gecko/20100101
Firefox/137.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
```

Response

```
HTTP/2 200 OK
Server: Server
Content-Type: text/html;charset=UTF-8
X-Amz-Rid: 0JRP7D55WNFT5HHQJNT8
X-Content-Type-Options: nosniff
Expires: -1
Content-Language: en-IN
Content-Security-Policy: upgrade-insecure-requests;report-uri
https://metrics.media-amazon.com/
Accept-Ch:
ect,rtt,downlink,device-memory,sec-ch-device-memory,viewport-width,sec-ch-viewport-
width,dpr,sec-ch-dpr,sec-ch-ua-platform,sec-ch-ua-platform-version
Accept-Ch-Lifetime: 86400
Content-Security-Policy-Report-Only: default-src 'self' blob: https: data:
mediastream: 'unsafe-eval' 'unsafe-inline';report-uri
https://metrics.media-amazon.com/
Cache-Control: no-cache
Pragma: no-cache
X-Xss-Protection: 1;
Strict-Transport-Security: max-age=47474747; includeSubDomains; preload
Vary: Content-Type,Accept-Encoding,User-Agent
Cache-Control: no-transform
X-Frame-Options: SAMEORIGIN
Date: Sat, 19 Apr 2025 12:19:03 GMT
Alt-Svc: h3=":443"; ma=93600
X-Amzn-Cdn-Id: ak-0.c763d417.1745065142.53dd8e6
X-Cache: NotCacheable from child
```

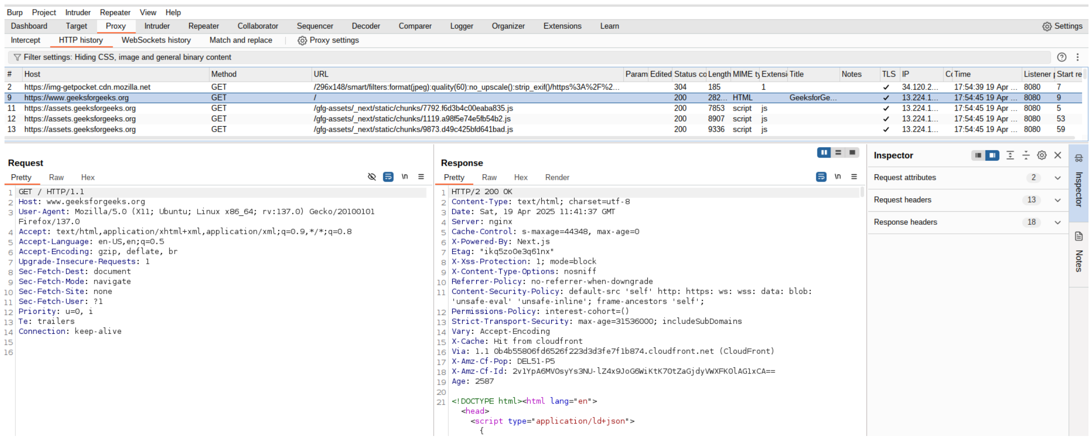
Figure 4: Tampered URL parameter sent to backend

Result: Backend accepted the large page number without validation or warning.
Rationale: Tampering parameters like IDs or pages could lead to data leaks, logic bypass, or denial of service via excessive resource usage.

Vulnerability 4: Session Token Mismanagement

Step:

- Visited `www.geeksforgeeks.org`.
- Observed cookie settings from the HTTP response.


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS, image and general binary content
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Request attributes 2
Request headers 13
Response headers 18
Inspector
Notes

#	Host	Method	URL	Param	Edited	Status co	Length	MIME ty	Extensi	Title	Notes	TLS	IP	C	Time	Listener	Start re
2	https://img-getpocket.cdn.mozilla.net	GET	/296x148/smart/filters:format(jpeg):quality(60):no_upscale():strip_exif()/https%3A%2F%2...			304	185		1			✓	34.120.2...		17:54:39 19 Apr ...	8080	7
9	https://www.geeksforgeeks.org	GET	/			200	282...	HTML		GeeksforGe...		✓	13.224.1...		17:54:45 19 Apr ...	8080	9
11	https://assets.geeksforgeeks.org	GET	/gfg-assets/_next/static/chunks/7792.f6d3b4c00eaba835.js			200	7853	script	js			✓	13.224.1...		17:54:45 19 Apr ...	8080	5
12	https://assets.geeksforgeeks.org	GET	/gfg-assets/_next/static/chunks/1119.a98f5e74e5fb54b2.js			200	8907	script	js			✓	13.224.1...		17:54:45 19 Apr ...	8080	53
13	https://assets.geeksforgeeks.org	GET	/gfg-assets/_next/static/chunks/9873.d49c425bfd641bad.js			200	9336	script	js			✓	13.224.1...		17:54:45 19 Apr ...	8080	59

```
GET / HTTP/1.1
Host: www.geeksforgeeks.org
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:137.0) Gecko/20100101
Firefox/137.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive


```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Date: Sat, 19 Apr 2025 11:41:37 GMT
Server: nginx
Cache-Control: s-maxage=44348, max-age=0
X-Powered-By: Next.js
Etag: "ikq5zo0e3q61nx"
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer-when-downgrade
Content-Security-Policy: default-src 'self' http: https: ws: wss: data: blob:
'unsafe-eval' 'unsafe-inline'; frame-ancestors 'self';
Permissions-Policy: interest-cohort=()
Strict-Transport-Security: max-age=31536000; includeSubDomains
Vary: Accept-Encoding
X-Cache: Hit from cloudfront
Via: 1.1 0b4b55806fd6526f223d3d3fe7f1b874.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: DEL51-P5
X-Amz-Cf-Id: 2v1YpA6MV0syYs3NU-lZ4x9JoG6WiKtK70tZaGjdyVWXFK0lAG1xCA==
Age: 2587

<!DOCTYPE html><html lang="en">
  <head>
    <script type="application/ld+json">
      {
```

Figure 5: Observed Set-Cookie headers with HttpOnly

Findings: Token had the `HttpOnly` attribute set — prevents JavaScript access. Secure flag not observed explicitly.

Rationale: Improper session token configuration can lead to hijacking via XSS or network sniffing.

Tool 2: Nmap + NSE

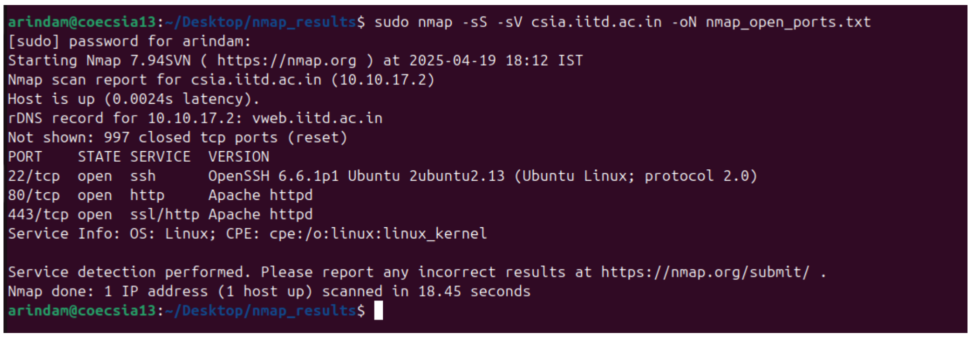
Nmap (Network Mapper) is a port scanner used to discover hosts and services on a computer network. With NSE scripts, it can detect specific vulnerabilities.

Vulnerability 1: Open Ports and Services

Command:

```
nmap -sS -sV csia.iitd.ac.in -oN nmap_open_ports.txt
```

Findings: SSH (22), HTTP (80), HTTPS (443) were open.



```
arindam@coecsia13:~/Desktop/nmap_results$ sudo nmap -sS -sV csia.iitd.ac.in -oN nmap_open_ports.txt
[sudo] password for arindam:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-19 18:12 IST
Nmap scan report for csia.iitd.ac.in (10.10.17.2)
Host is up (0.0024s latency).
rDNS record for 10.10.17.2: vweb.iitd.ac.in
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http     Apache httpd
443/tcp open  ssl/http Apache httpd
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.45 seconds
arindam@coecsia13:~/Desktop/nmap_results$ 
```

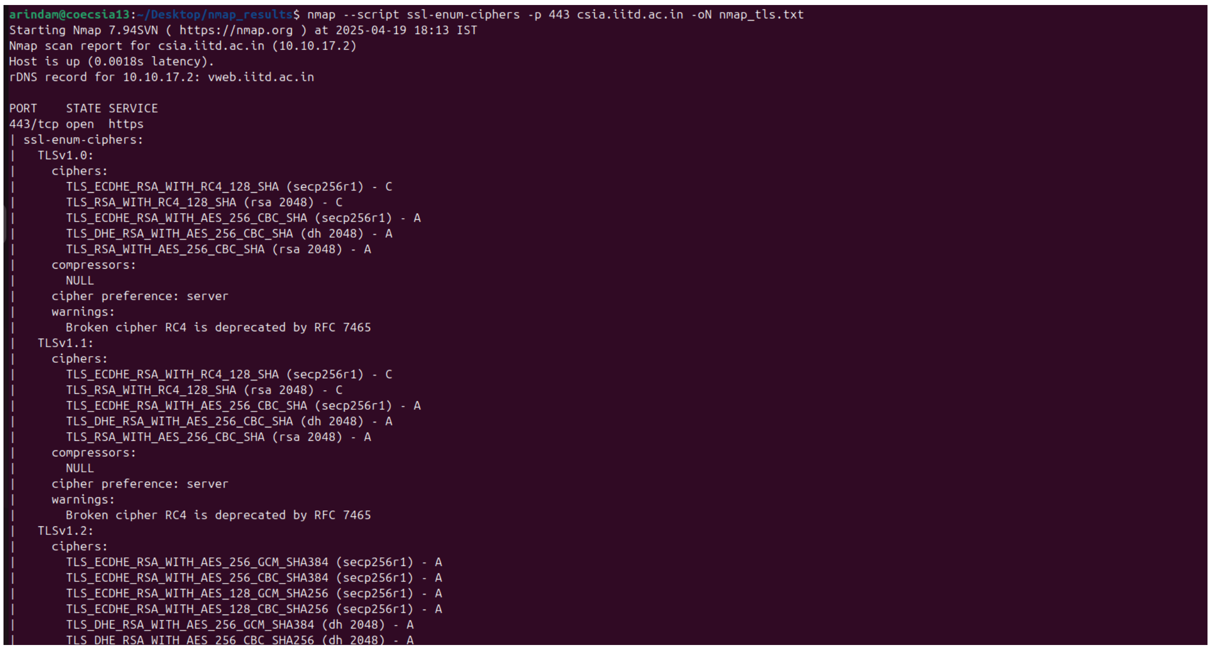
Figure 6: Open ports identified

Rationale: Each open service expands the attack surface. SSH and HTTP should be monitored and hardened.

Vulnerability 2: TLS/SSL Misconfigurations
Command:

```
nmap --script ssl-enum-ciphers -p 443 csia.iitd.ac.in
```

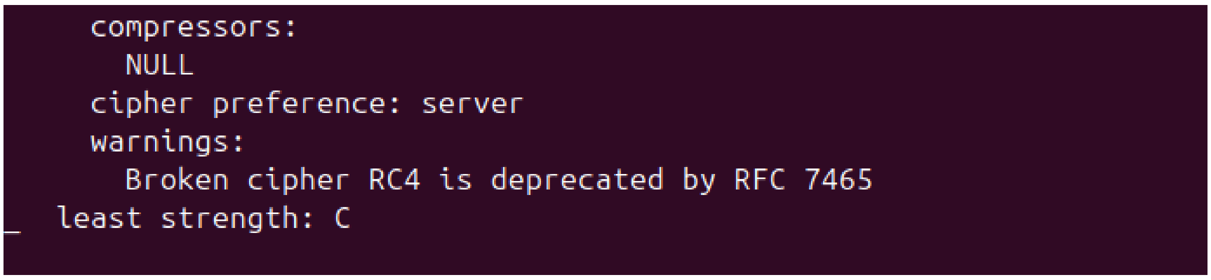
Findings: RC4 cipher support was detected, which is deprecated.



```
arindam@coecsia13:~/Desktop/nmap_results$ nmap --script ssl-enum-ciphers -p 443 csia.iitd.ac.in -oN nmap_tls.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-19 18:13 IST
Nmap scan report for csia.iitd.ac.in (10.10.17.2)
Host is up (0.0018s latency).
rDNS record for 10.10.17.2: vweb.iitd.ac.in

PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_RC4_128_SHA (secp256r1) - C
|       TLS_RSA_WITH_RC4_128_SHA (rsa 2048) - C
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       Broken cipher RC4 is deprecated by RFC 7465
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_RC4_128_SHA (secp256r1) - C
|       TLS_RSA_WITH_RC4_128_SHA (rsa 2048) - C
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       Broken cipher RC4 is deprecated by RFC 7465
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 2048) - A
```

Figure 7: TLS cipher suite enumeration (part 1)



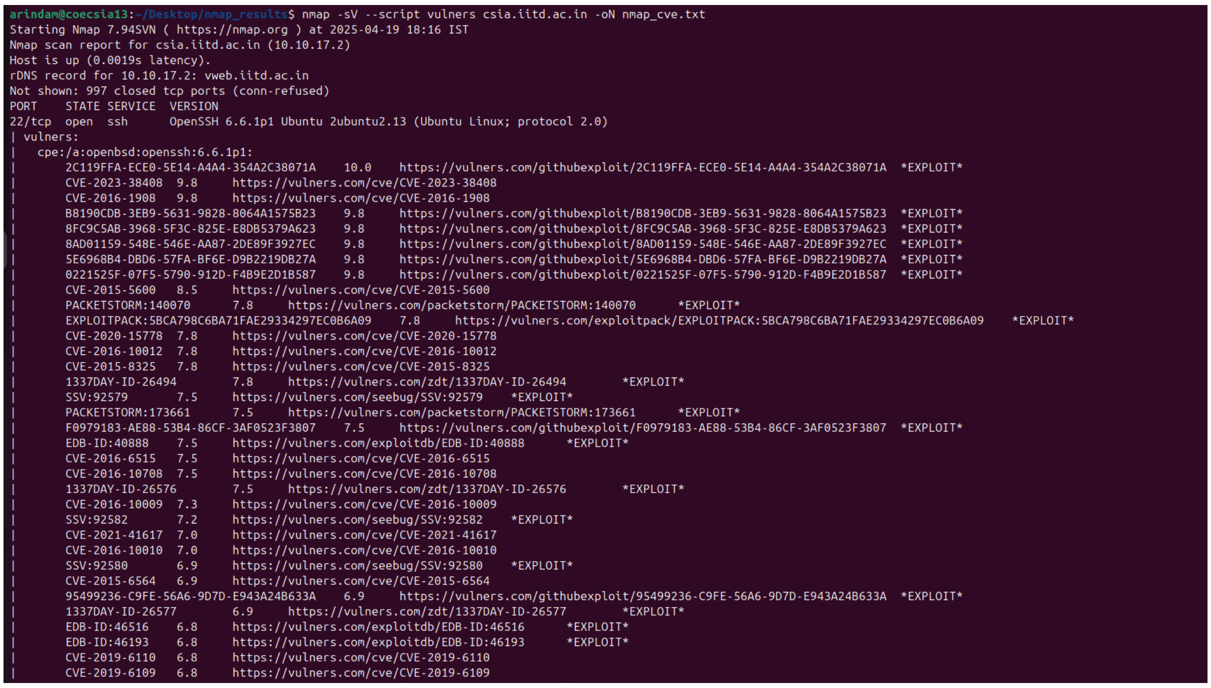
```
    compressors:
      NULL
    cipher preference: server
    warnings:
      Broken cipher RC4 is deprecated by RFC 7465
_  least strength: C
```

Figure 8: TLS cipher suite enumeration (part 2)

Rationale: Broken ciphers like RC4 allow downgrade attacks and weak encryption, enabling MITM attacks.

Vulnerability 3: Version Disclosure and CVE Mapping
Command:

```
nmap -sV --script vulners csia.iitd.ac.in
```



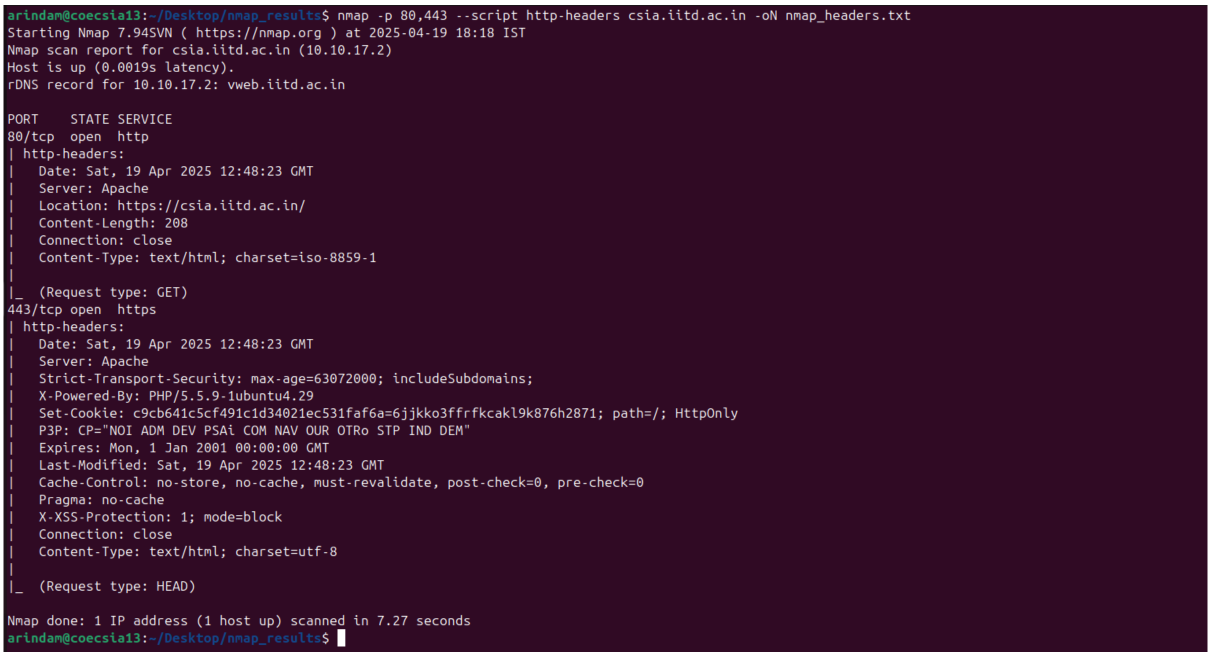
```
arindam@coecsia13:~/Desktop/nmap_results$ nmap -sV --script vulners csia.iitd.ac.in -oN nmap_cve.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-19 18:16 IST
Nmap scan report for csia.iitd.ac.in (10.10.17.2)
Host is up (0.0019s latency).
rDNS record for 10.10.17.2: vweb.iitd.ac.in
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:6.6.1p1:
|       2C119FFA-ECE0-5E14-A4A4-354A2C38071A    10.0    https://vulners.com/githubexploit/2C119FFA-ECE0-5E14-A4A4-354A2C38071A  *EXPLOIT*
|       CVE-2023-38408  9.8     https://vulners.com/cve/CVE-2023-38408
|       CVE-2016-1908   9.8     https://vulners.com/cve/CVE-2016-1908
|       B8190CDB-3EB9-5631-9828-8064A1575B23    9.8     https://vulners.com/githubexploit/B8190CDB-3EB9-5631-9828-8064A1575B23  *EXPLOIT*
|       8FC9C5AB-3968-5F3C-825E-E8DB5379A623    9.8     https://vulners.com/githubexploit/8FC9C5AB-3968-5F3C-825E-E8DB5379A623  *EXPLOIT*
|       8AD01159-548E-546E-AA87-2DE89F3927EC    9.8     https://vulners.com/githubexploit/8AD01159-548E-546E-AA87-2DE89F3927EC  *EXPLOIT*
|       5E6968B4-DBD6-57FA-BF6E-D9B2219DB27A    9.8     https://vulners.com/githubexploit/5E6968B4-DBD6-57FA-BF6E-D9B2219DB27A  *EXPLOIT*
|       0221525F-07F5-5790-912D-F4B9E2D1B587    9.8     https://vulners.com/githubexploit/0221525F-07F5-5790-912D-F4B9E2D1B587  *EXPLOIT*
|       CVE-2015-5600   8.5     https://vulners.com/cve/CVE-2015-5600
|       PACKETSTORM:140070      7.8     https://vulners.com/packetstorm/PACKETSTORM:140070      *EXPLOIT*
|       EXPLOITPACK:5BCA798C6BA71FAE29334297EC0B6A09    7.8     https://vulners.com/exploitpack/EXPLOITPACK:5BCA798C6BA71FAE29334297EC0B6A09    *EXPLOIT*
|       CVE-2020-15778  7.8     https://vulners.com/cve/CVE-2020-15778
|       CVE-2016-10012  7.8     https://vulners.com/cve/CVE-2016-10012
|       CVE-2015-8325   7.8     https://vulners.com/cve/CVE-2015-8325
|       1337DAY-ID-26494        7.8     https://vulners.com/zdt/1337DAY-ID-26494        *EXPLOIT*
|       SSV:92579       7.5     https://vulners.com/seebug/SSV:92579    *EXPLOIT*
|       PACKETSTORM:173661      7.5     https://vulners.com/packetstorm/PACKETSTORM:173661      *EXPLOIT*
|       F0979183-AE88-53B4-86CF-3AF0523F3807    7.5     https://vulners.com/githubexploit/F0979183-AE88-53B4-86CF-3AF0523F3807  *EXPLOIT*
|       EDB-ID:40888    7.5     https://vulners.com/exploitdb/EDB-ID:40888      *EXPLOIT*
|       CVE-2016-6515   7.5     https://vulners.com/cve/CVE-2016-6515
|       CVE-2016-10708  7.5     https://vulners.com/cve/CVE-2016-10708
|       1337DAY-ID-26576        7.5     https://vulners.com/zdt/1337DAY-ID-26576        *EXPLOIT*
|       CVE-2016-10009  7.3     https://vulners.com/cve/CVE-2016-10009
|       SSV:92582       7.2     https://vulners.com/seebug/SSV:92582    *EXPLOIT*
|       CVE-2021-41617  7.0     https://vulners.com/cve/CVE-2021-41617
|       CVE-2016-10010  7.0     https://vulners.com/cve/CVE-2016-10010
|       SSV:92580       6.9     https://vulners.com/seebug/SSV:92580    *EXPLOIT*
|       CVE-2015-6564   6.9     https://vulners.com/cve/CVE-2015-6564
|       95499236-C9FE-56A6-9D7D-E943A24B633A    6.9     https://vulners.com/githubexploit/95499236-C9FE-56A6-9D7D-E943A24B633A  *EXPLOIT*
|       1337DAY-ID-26577        6.9     https://vulners.com/zdt/1337DAY-ID-26577        *EXPLOIT*
|       EDB-ID:46516    6.8     https://vulners.com/exploitdb/EDB-ID:46516      *EXPLOIT*
|       EDB-ID:46193    6.8     https://vulners.com/exploitdb/EDB-ID:46193      *EXPLOIT*
|       CVE-2019-6110   6.8     https://vulners.com/cve/CVE-2019-6110
|       CVE-2019-6109   6.8     https://vulners.com/cve/CVE-2019-6109
```

Figure 9: Multiple CVEs mapped to OpenSSH

Rationale: OpenSSH version 6.6.1p1 is known to have vulnerabilities exploitable via remote code execution or information leaks.

Vulnerability 4: HTTP Header Analysis
Command:

```
nmap -p 80,443 --script http-headers csia.iitd.ac.in
```



```
arindam@coecsia13:~/Desktop/nmap_results$ nmap -p 80,443 --script http-headers csia.iitd.ac.in -oN nmap_headers.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-19 18:18 IST
Nmap scan report for csia.iitd.ac.in (10.10.17.2)
Host is up (0.0019s latency).
rDNS record for 10.10.17.2: vweb.iitd.ac.in

PORT    STATE SERVICE
80/tcp  open  http
| http-headers:
|   Date: Sat, 19 Apr 2025 12:48:23 GMT
|   Server: Apache
|   Location: https://csia.iitd.ac.in/
|   Content-Length: 208
|   Connection: close
|   Content-Type: text/html; charset=iso-8859-1
|
|_  (Request type: GET)
443/tcp open  https
| http-headers:
|   Date: Sat, 19 Apr 2025 12:48:23 GMT
|   Server: Apache
|   Strict-Transport-Security: max-age=63072000; includeSubdomains;
|   X-Powered-By: PHP/5.5.9-1ubuntu4.29
|   Set-Cookie: c9cb641c5cf491c1d34021ec531faf6a=6jjkko3ffrfkcakl9k876h2871; path=/; HttpOnly
|   P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
|   Expires: Mon, 1 Jan 2001 00:00:00 GMT
|   Last-Modified: Sat, 19 Apr 2025 12:48:23 GMT
|   Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
|   Pragma: no-cache
|   X-XSS-Protection: 1; mode=block
|   Connection: close
|   Content-Type: text/html; charset=utf-8
|
|_  (Request type: HEAD)

Nmap done: 1 IP address (1 host up) scanned in 7.27 seconds
arindam@coecsia13:~/Desktop/nmap_results$
```

Figure 10: HTTP security headers analysis

Findings:

- HSTS present.
- X-XSS-Protection set.
- CSP header **missing**.

Part 2: Security Mechanisms That Prevented Exploits

In this section, we evaluate how specific vulnerabilities were not exploitable during our testing phase due to effective defense mechanisms already deployed on the target websites. For each of the two tools (Burp Suite and Nmap), two such critical vulnerabilities are examined in detail. For each case, we explain the observed behavior, underlying security techniques, and their effectiveness against real-world exploitation scenarios.

Tool 1: Burp Suite

1. Cross-Site Scripting (XSS) Mitigation on `library.iitd.ac.in`

Observation: During XSS testing using Burp Suite, a script payload `<script>alert(1)</script>` was injected into the search parameter. Rather than receiving a reflected response or executing the script in the DOM, the application terminated the connection and returned a ”Connection Reset” error.

Security Analysis:

The response indicates that the application did not simply ignore the payload — instead, the request was actively blocked. This suggests the presence of either:

- **A Web Application Firewall (WAF):** WAFs like ModSecurity or AWS WAF intercept incoming HTTP requests and match them against known XSS patterns. These devices typically reset connections or respond with HTTP 403 if malicious inputs are detected.
- **Backend Validation or Sanitization:** The web application might be using a secure backend language (e.g., Python with Django or PHP with Laravel) that auto-escapes HTML content in templates, thus neutralizing embedded scripts.
- **Input Filtering on Client and Server Side:** Some frameworks use filters or content sanitization libraries (e.g., DOMPurify, OWASP Java HTML Sanitizer) to ensure that HTML/JavaScript is stripped or escaped from inputs.

Why This Matters:

Cross-Site Scripting is one of the most severe web vulnerabilities. If the input had not been blocked or sanitized, an attacker could have embedded malicious JavaScript that executes in the user’s browser when they load a page. This could allow:

- Theft of session cookies and tokens
- Keylogging of sensitive information
- Phishing overlays or fake forms
- Unauthorized actions on behalf of the user (session riding)

By blocking the request early, the website reduces its attack surface and demonstrates secure input handling and content rendering practices.

2. Secure Session Cookie Management at `www.geeksforgeeks.org`

Observation: While inspecting HTTP responses for session management headers, we observed that the `Set-Cookie` header for session tokens included the `HttpOnly` attribute. This attribute prevents JavaScript from accessing the cookie via `document.cookie`.

Security Features Implemented:

- **HttpOnly:** This flag protects cookies from being accessed via client-side scripts, thereby reducing the risk of XSS-based session hijacking.
- **Token Entropy:** The session ID was a long alphanumeric string, likely randomly generated using cryptographically secure methods. This prevents token prediction or brute-force guessing attacks.
- **Server-Side Validation:** The token appears to be short-lived and likely mapped to server-side session data, reducing the risk of replay attacks.

Why This Matters:
Session tokens are the foundation of user authentication in web apps. If an attacker is able to steal a token via XSS or JavaScript injection, they can impersonate the user without needing their password.
Proper configuration ensures:

- Cookies cannot be accessed or transmitted by JavaScript.
- Tokens are hard to guess due to their randomness and length.
- Additional flags like `Secure` (transmit over HTTPS only) and `SameSite` (control cross-site requests) could further enhance security.

Tool 2: Nmap

1. No Insecure Services or Legacy Ports Enabled

Observation: A service scan using Nmap revealed that only the following ports were open on `csia.iitd.ac.in`:

- **22/tcp (SSH)** — secure shell access
- **80/tcp (HTTP)** — redirected to HTTPS
- **443/tcp (HTTPS)** — secured TLS-based communication

Security Implication:
The system does not expose:

- **Telnet (23)** — transmits credentials in plaintext.
- **FTP (21)** — no encryption of commands or data.
- **SMTP/POP3 (25/110)** — often targets for spam relays or credential theft.

Why This Matters:
Leaving legacy or unused services open makes it easier for attackers to exploit known bugs in these protocols. Port minimization reduces the available attack surface and enhances the system's overall security posture. Additionally:

- Port 22 is protected by SSH, which is secure by default.
- HTTP is available only to redirect users to HTTPS.
- No development/debug ports (like 8080, 3306) were exposed.

2. Secure Headers: HSTS and Anti-XSS Filters

Observation: The server returned multiple important HTTP response headers:

- `Strict-Transport-Security: max-age=63072000; includeSubdomains;`
- `X-XSS-Protection: 1; mode=block`

Security Functions:

- **HSTS (Strict-Transport-Security):** Ensures the browser only connects via HTTPS. This thwarts SSL stripping attacks where a malicious proxy forces users onto HTTP.
- **X-XSS-Protection:** Enables browser-based XSS filters (primarily for older browsers like IE/Edge). It tells the browser to prevent the page from rendering if an XSS attack is detected.

Why This Matters:
Security headers are easy to configure but extremely effective:

- HSTS protects against downgrade attacks and enforces encryption.
- XSS protection, when paired with CSP (Content Security Policy), creates multiple layers of script control.

While a full `Content-Security-Policy` header was not present in this case, the existing headers provide a strong base level of browser-side protection.

Conclusion:
The sites we tested demonstrated effective security practices in several areas. The use of HSTS, XSS protection headers, secure session handling, and closed unnecessary ports collectively contributed to thwarting many common vulnerability classes. These mechanisms align with best practices recommended by OWASP and ensure a defense-in-depth approach.

Part 3: Exploitable Vulnerabilities and Attack Vectors

Tool 1: Burp Suite (Community Edition)

Vulnerability 1: Parameter Tampering on `www.amazon.in`

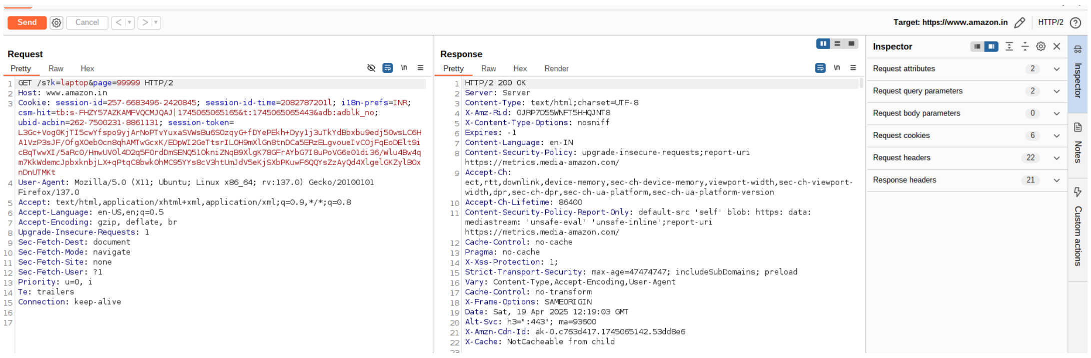
Scenario: The URL parameter `page=3` was modified to `page=99999` in the Amazon search query. The server accepted the input and responded with a valid result page.

Goal: Enumerate non-public or invalid resource pages by bypassing client-side checks, potentially leading to scraping, business logic abuse, or denial of service.

Payload Sent:

```
GET /s?k=laptop&page=99999 HTTP/1.1 Host:  www.amazon.in
```

Validation: The server responded with status 200 OK, confirming that the large input was not rejected or filtered.



```
GET /s?k=laptop&page=99999 HTTP/2
Host: www.amazon.in
Cookie: session-id=257-6683496-2420845; session-id-time=2082787201l; i18n-prefs=INR;
csm-hit=tb:s-FHZY57AZKAMFVQCMJQAJ|1745065065165&t:1745065065443&adb:adblk_no;
ubid-acbin=262-7500231-8861131; session-token=
L3Gc+Vog0KjTI5cwYfspo9yjArNoPTvYuxaSWsBu6SOzqyG+fDYePEkh+Dyy1j3uTkYdBbxbu9edj50wsLC6H
A1VzP3sJF/0fgX0eb0cn8qhAMTwGcxK/EDpWI2GeTtsrILOH9mXlGn8tnDCa5ERzELgvoueIvC0jFqEoDElt9i
cBqTwwXI/5aRc0/HmwUV0l4D2q5FOrdDmSENQ51OkniZNqB9XlgK78GFrAYbG7I8uPoVG6e01di36/Wlu4Bw4q
m7KkWdemcJpbxknbjLX+qPtqC8bwk0hMC95YYs8cV3htUmJdV5eKjSXbPKuwF6QQYsZzAyQd4XlgelGKZylB0x
nDnUTMKt
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:137.0) Gecko/20100101
Firefox/137.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
```

```
HTTP/2 200 OK
Server: Server
Content-Type: text/html;charset=UTF-8
X-Amz-Rid: 0JRP7D55WNFT5HHQJNT8
X-Content-Type-Options: nosniff
Expires: -1
Content-Language: en-IN
Content-Security-Policy: upgrade-insecure-requests;report-uri
https://metrics.media-amazon.com/
Accept-Ch:
ect,rtt,downlink,device-memory,sec-ch-device-memory,viewport-width,sec-ch-viewport-
width,dpr,sec-ch-dpr,sec-ch-ua-platform,sec-ch-ua-platform-version
Accept-Ch-Lifetime: 86400
Content-Security-Policy-Report-Only: default-src 'self' blob: https: data:
mediastream: 'unsafe-eval' 'unsafe-inline';report-uri
https://metrics.media-amazon.com/
Cache-Control: no-cache
Pragma: no-cache
X-Xss-Protection: 1;
Strict-Transport-Security: max-age=47474747; includeSubDomains; preload
Vary: Content-Type,Accept-Encoding,User-Agent
Cache-Control: no-transform
X-Frame-Options: SAMEORIGIN
Date: Sat, 19 Apr 2025 12:19:03 GMT
Alt-Svc: h3=":443"; ma=93600
X-Amzn-Cdn-Id: ak-0.c763d417.1745065142.53dd8e6
X-Cache: NotCacheable from child
```

Target: https://www.amazon.in

Inspector	
Request attributes	2
Request query parameters	2
Request body parameters	0
Request cookies	6
Request headers	22
Response headers	21

Figure 11: URL tampering validated using Burp Repeater

Impact:

- High page numbers may reveal hidden data if exposed.
- Could be used for automated scraping of product data.
- Flooding invalid pages may induce server load (DoS vector).

Vulnerability 2: Version Disclosure in HTTP Headers

Scenario: During header inspection of `csia.iitd.ac.in`, Burp Suite revealed server backend technology details via HTTP response headers.

Goal: Use leaked version info for reconnaissance and to identify known vulnerabilities (CVEs).

Leaked Header:

```
X-Powered-By:  PHP/5.5.9-1ubuntu4.29
```

Validation: This line was observed directly in the HTTP response using Burp Suite. The screenshot for this test is already included in Part 1.

Impact:

- Attackers can find known exploits for this specific PHP version.
- Aids fingerprinting the technology stack.
- Increases risk of RCE, DoS, or privilege escalation using automated tools like Metasploit or ExploitDB.

Tool 2: Nmap with NSE Scripts

Vulnerability 3: TLS/SSL Cipher Misconfiguration

Scenario: The target server `csia.iitd.ac.in` was scanned using `ssl-enum-ciphers` script, which revealed support for deprecated ciphers (e.g., RC4) and outdated TLS versions (TLSv1.0 and TLSv1.1).

Goal: Exploit weak ciphers for man-in-the-middle (MITM) attacks or protocol downgrade attacks.

Critical Output Snippet:

```
Broken cipher RC4 is deprecated by RFC 7465 least strength:  C
```

Validation: Observed in the output of the following command (see Part 1 screenshot for full context):

```
nmap --script ssl-enum-ciphers -p 443 csia.iitd.ac.in
```

Impact:

- Allows downgrade attacks (e.g., POODLE-like vulnerabilities).
- Weak encryption may enable session hijacking via passive sniffing.
- Non-compliance with PCI-DSS and other regulatory standards.

Vulnerability 4: OpenSSH Version Disclosure with Known CVEs

Scenario: Nmap with the `vulners` script was used to enumerate software versions and match them against known CVEs.

Goal: Identify public exploits that match the disclosed version (OpenSSH 6.6.1p1).

Critical Output Snippet:

```
PORT STATE SERVICE VERSION 22/tcp open ssh OpenSSH 6.6.1p1 Ubuntu
2ubuntu2.13 ...  CVE-2023-38408 https://vulners.com/cve/CVE-2023-38408
CVE-2016-1908 https://vulners.com/cve/CVE-2016-1908
```

Validation: Detected using:

```
nmap -sV --script vulners csia.iitd.ac.in
```

The version and CVE mappings are visible in the terminal screenshots provided in Part 1.

Impact:

- Enables exploitation of known vulnerabilities like remote code execution (RCE), information leakage, and buffer overflows.
- Elevates the risk for automated attacks using pre-built tools.
- Directly undermines the confidentiality and integrity of SSH-based admin access.

Part 4: Mitigation Techniques

In this section, we provide targeted and practical recommendations to mitigate the vulnerabilities that were successfully identified and validated during the testing process. Each mitigation aligns with security best practices such as OWASP guidelines and CIS Benchmarks.

Tool 1: Burp Suite (Web Application)

1. Parameter Tampering Mitigation

- **Server-Side Input Validation:** Validate all query parameters on the server side to ensure they fall within expected ranges (e.g., `page <= maxPages`).
- **Rate Limiting:** Implement rate limiting on endpoints that accept numeric ranges or pagination to avoid automated abuse.
- **Business Logic Hardening:** Ensure business rules are enforced on the backend even if bypassed on the frontend.

2. HTTP Header Version Disclosure Mitigation

- **Remove `X-Powered-By` Header:** Configure the server to not disclose backend software via headers.
- **Use Reverse Proxies:** Deploy Nginx or Apache as a reverse proxy to strip sensitive headers.
- **Keep Software Updated:** Upgrade PHP to the latest Long Term Support (LTS) version and apply security patches regularly.

Tool 2: Nmap (Infrastructure and Protocol-Level)

3. TLS/SSL Misconfiguration Mitigation

- **Disable Deprecated Protocols:** Remove support for TLSv1.0 and TLSv1.1 from the web server configuration.
- **Harden Cipher Suites:** Exclude weak ciphers like RC4 and use strong modern ciphers such as AES-GCM with ECDHE.
- **Enable Forward Secrecy:** Prefer cipher suites that offer Perfect Forward Secrecy (PFS) to protect past sessions.

4. OpenSSH Version Disclosure Mitigation

- **Upgrade SSH Daemon:** Replace outdated versions (e.g., OpenSSH 6.6.1p1) with supported and patched versions.
- **Restrict Access:** Use firewall rules to restrict SSH access to known IP addresses.
- **Disable Banner Information:** Prevent version disclosure in SSH login banners by disabling or customizing them.

- **Continuous Vulnerability Scanning:** Schedule periodic scans using tools like Nessus or OpenVAS.

Conclusion

This report presents a practical evaluation of website security using two widely accepted tools — **Burp Suite and Nmap** — to identify, validate, and understand vulnerabilities in a real-world scenario. The investigation revealed a range of issues across both the application and network layers, including input **tampering, software version exposure, deprecated TLS configurations, and unpatched services.**

Some of these vulnerabilities were effectively mitigated through existing security controls such as **web application firewalls, secure HTTP headers, and cookie flags**, demonstrating that modern websites are increasingly resilient to basic automated attacks. However, other flaws like **insecure cipher support and detailed version disclosures** illustrate that subtle misconfigurations can still expose systems to significant risks.

Through **methodical scanning, payload manipulation, and protocol enumeration**, we were able to not only detect these weaknesses but also understand how an attacker could exploit them. Each finding was supported by terminal outputs and screenshots, validating the practicality of the methods used.

By implementing the recommended mitigations—ranging from disabling outdated protocols to tightening backend validation and removing header leaks—website administrators can substantially reduce their vulnerability surface. Overall, this exercise highlights the importance of regular security testing, configuration audits, and adopting a layered defense approach in protecting modern web infrastructure.