SIL765 Assignment 5

Naman Nirwan

2021CS50593

1. TLS Handshake (Task A1)

We used a Python client program to initiate a TLS handshake with `www.github.com`. The program was modified to print the cipher suite used and the server certificate.

Cipher Used

The cipher suite used in the connection was displayed using `ssock.cipher()` which comes out to be ('TLS_AES_128_GCM_SHA256', 'TLSv1.3', 128). This confirms that the TLS handshake successfully negotiated a secure cipher suite.

Server Certificate

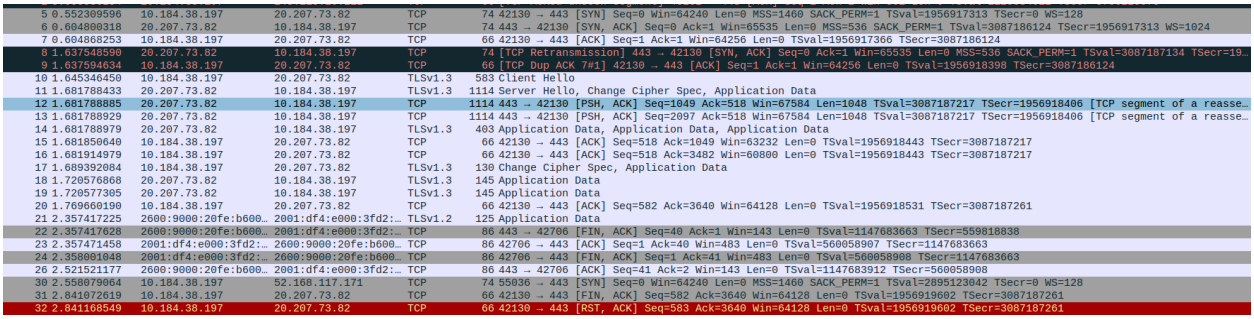
The server certificate was printed using `ssock.getpeercert()`. It showed details like issuer, validity period, and subject.

```
After making TCP connection. Press any key to continue ...
{'OCSP': ('http://ocsp.sectigo.com',),
 'caIssuers': ('http://crt.sectigo.com/SectigoECCDomainValidationSecureServerCA.crt',),
 'issuer': ((('countryName', 'GB'),),
            (('stateOrProvinceName', 'Greater Manchester'),),
            (('localityName', 'Salford'),),
            (('organizationName', 'Sectigo Limited'),),
            (('commonName',
              'Sectigo ECC Domain Validation Secure Server CA'),)),
 'notAfter': 'Feb  5 23:59:59 2026 GMT',
 'notBefore': 'Feb  5 00:00:00 2025 GMT',
 'serialNumber': 'AB6686B5627BE80596821330128649F5',
 'subject': ((('commonName', 'github.com'),),),
 'subjectAltName': (('DNS', 'github.com'), ('DNS', 'www.github.com')),
 'version': 3}
('TLS_AES_128_GCM_SHA256', 'TLSv1.3', 128)
After handshake. Press any key to continue ...
```

Wireshark Analysis

Wireshark was used to capture the traffic during the handshake. The handshake included:

- TCP 3-way handshake (SYN, SYN-ACK, ACK)
- TLS handshake messages (Client Hello, Server Hello, Certificate, Finished)



2. Certificate Authority Verification (Task A2)

We created a new folder `certs/` and set `cadir = './certs'`. Initially, the TLS handshake failed because the directory had no trusted CA certificates.

Error Observed

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

```
py www.github.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "/home/arthur/Downloads/SEMVIII/SIL765/Assignment_5/tls_client.py", line 22, in <module>
    ssock.do_handshake() # Start the handshake
    ^^^^^^^^^^^^^^^^^^^^
  File "/home/arthur/miniconda3/lib/python3.12/ssl.py", line 1319, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certif
icate (_ssl.c:1010)
```

Fix

We added valid intermediate certificates from the directory /etc/ssl/certs to ./certs directory and then re-running the client with `load_verify_locations`, the handshake succeeded.

```
py www.github.com
After making TCP connection. Press any key to continue ...
{'OCSP': ('http://ocsp.sectigo.com',),
 'caIssuers': ('http://crt.sectigo.com/SectigoECCDomainValidationSecureServerCA.crt',),
 'issuer': ((('countryName', 'GB'),),
            (('stateOrProvinceName', 'Greater Manchester'),),
            (('localityName', 'Salford'),),
            (('organizationName', 'Sectigo Limited'),),
            (('commonName',
              'Sectigo ECC Domain Validation Secure Server CA'),)),
 'notAfter': 'Feb  5 23:59:59 2026 GMT',
 'notBefore': 'Feb  5 00:00:00 2025 GMT',
 'serialNumber': 'AB6686B5627BE80596821330128649F5',
 'subject': ((('commonName', 'github.com'),),),
 'subjectAltName': (('DNS', 'github.com'), ('DNS', 'www.github.com')),
 'version': 3}
After handshake. Press any key to continue ...
```

3. Hostname Verification (Task A3)

We tested the effect of the `check_hostname` flag in the TLS context.

Setup

- We used `dig github.com +short` to get the IP of github.com .
- Mapped the IP to `anything.com` in `/etc/hosts`.

Observations

- `check_hostname = False` allowed the connection to proceed.
- `check_hostname = True` caused the connection to fail with a mismatch error.

```
py anything.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "/home/arthur/Downloads/SEMVIII/SIL765/Assignment_5/tls_client.py", line 22, in <module>
    ssock.do_handshake() # Start the handshake
    ^^^^^^^^^^^^^^^^^^^^
  File "/home/arthur/miniconda3/lib/python3.12/ssl.py", line 1319, 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:1010)
```

Security Implication

Disabling hostname checking opens the door to Man-in-the-Middle (MitM) attacks since the certificate's Common Name is not verified against the domain.

4. Sending and Receiving Data (Task A4)

Sending HTTP Request

We sent an HTTP GET request over TLS to `github.com`. The response was a `301 Moved Permanently` redirect.

```
HTTP/1.1 301 Moved Permanently
Location: https://github.com/
```

```
🌐 HTTP Response:
[b'HTTP/1.1 301 Moved Permanently',
 b'Content-Length: 0',
 b'Location: https://github.com/',
 b'Strict-Transport-Security: max-age=31536000; includeSubDomains; preload',
 b'connection: close',
 b'',
 b'']
```

Fetching an Image

We then modified the request to fetch `/favicon.ico`, and saved the binary data to a file.

Image Saved

`github_favicon.ico` was successfully saved from the HTTPS response.

```
After handshake. Press any key to continue ...
[b'HTTP/1.1 301 Moved Permanently',
 b'Content-Length: 0',
 b'Location: https://github.com/favicon.ico',
 b'Strict-Transport-Security: max-age=31536000; includeSubDomains; preload',
 b'connection: close',
 b'',
 b'']
```

END FOR PART A REPORT

PART B REPORT

Website Security Analysis – Tool 1: Nmap

1. Tool Introduction

Nmap (Network Mapper) is a widely-used open-source tool designed for network discovery and security auditing. It allows administrators and testers to identify open ports, running services, service versions, and known vulnerabilities by sending specially crafted packets and analyzing the responses. Through its Nmap Scripting Engine (NSE), users can automate vulnerability scans and collect detailed insights about a target system. In this assignment, Nmap is used to analyze a deliberately vulnerable website: `http://testphp.vulnweb.com`.

2. Vulnerabilities Tested and Found

The scan was conducted using the command `nmap --script vuln testphp.vulnweb.com`. The following vulnerabilities were identified:

1. **Cross-Domain Policy Misconfiguration:** The presence of `crossdomain.xml` and `clientaccesspolicy.xml` with wildcard domain access poses a significant security risk. These files allow any third-party domain to interact with the site's content, potentially exposing user sessions and sensitive data.
2. **Cross-Site Request Forgery (CSRF):** Multiple forms across the site were detected without CSRF protection mechanisms. These include login, user info update, and guestbook forms. Absence of CSRF tokens makes the site vulnerable to unauthorized actions initiated from malicious third-party websites.
3. **SQL Injection (SQLi):** Numerous input parameters are vulnerable to SQL injection attacks. The scanner discovered endpoints like `/search.php`, `/listproducts.php`, and `/artists.php` that accept unsanitized user inputs directly into SQL queries.
4. **Exposed Administrative and Sensitive Directories:** Folders such as `/admin/`, `/CVS/`, and `/vendor/` were found to be publicly accessible. These may reveal configuration files, debug logs, or backend source code.

3. Critical Vulnerabilities Tested but Not Found

1. **DOM-based XSS:** The scanner attempted to exploit DOM-based cross-site scripting, where malicious scripts are executed by manipulating the Document Object Model on the client-side. The target website did not reflect any injected scripts in the browser DOM, indicating this vulnerability was not present. One possible reason the vulnerability was not exposed is that the application uses client-side JavaScript libraries or encoding mechanisms that neutralize injected content before execution.

2. **Stored XSS:** Stored XSS attempts were made on persistent forms like the guestbook. However, the payloads did not persist or execute upon retrieval, suggesting that the server either escapes stored content or prevents script injection. The server might be employing input sanitization and output encoding techniques, such as stripping HTML tags or converting characters like '¡' and '¿' to safe equivalents, thus neutralizing malicious scripts.

These results reflect partial adherence to security best practices, with certain filters or encoding mechanisms in place to mitigate advanced attack vectors.

4. Critical Vulnerabilities Found and Their Exploitation Potential

1. **SQL Injection (SQLi):** SQLi allows attackers to manipulate backend database queries. For example, by entering `' OR '1'='1` in a search field, an attacker may bypass authentication, extract user data, or even modify the database. This can be confirmed through error messages or observing unexpected data exposure in query results.

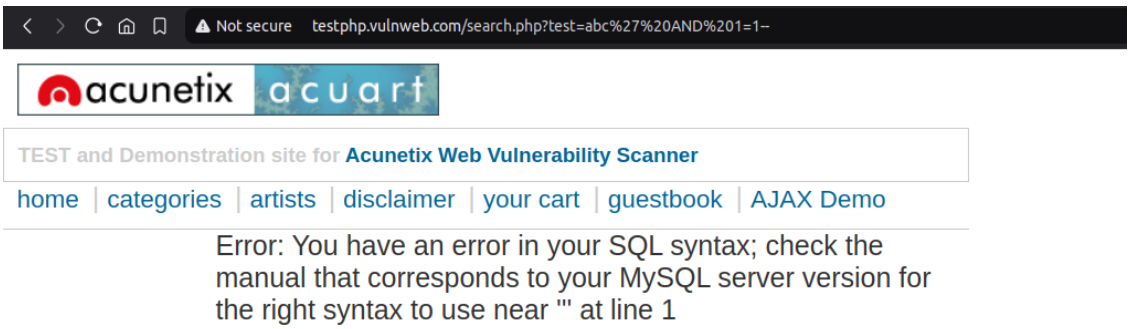


Figure 1: SQL Injection Error Message on `/search.php` Endpoint

2. **Cross-Domain Policy Misconfiguration:** This issue enables client-side scripts from malicious domains to access data on the vulnerable site, bypassing the Same-Origin Policy. It can lead to session hijacking, data leakage, or unintended actions performed on behalf of authenticated users.

5. Mitigation Strategies

- **Input Validation and Parameterized Queries:** All user inputs should be validated and sanitized. SQL queries must use parameterized statements to separate code from data.
- **CSRF Token Implementation:** Every form that modifies server-side data should include a cryptographically strong CSRF token validated with each submission.

- **Restrict Cross-Domain Access:** Limit access in crossdomain.xml and clientaccesspolicy.xml to trusted domains only. Avoid using wildcard entries.
- **Disable Directory Listing:** Prevent directory indexing in the web server configuration to avoid leaking file structure or internal resources.

6. Scan Output and Screenshots

The scan results obtained from Nmap using the NSE vulnerability script are shown below:

```
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.31s latency).
PORT   STATE SERVICE
80/tcp open  http
| http-cross-domain-policy:
|   VULNERABLE: Cross-domain and Client Access policies.
| http-csrf:
|   Found possible CSRF vulnerabilities in multiple forms
| http-sql-injection:
|   Possible sqli for queries:
|     /search.php?test=query' OR sqlspider
|     /listproducts.php?cat=3' OR sqlspider
|     /artists.php?artist=1' OR sqlspider
| http-enum:
|   /admin/: Possible admin folder
|   /CVS/: Potentially interesting folder
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-20 19:15 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.027s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 12.49 seconds
```

Figure 2: Initial Nmap Port Scan

Website Security Analysis – Tool 2: Nikto

1. Tool Introduction

Nikto is a robust and widely used open-source web server vulnerability scanner. Its primary purpose is to identify potential vulnerabilities, insecure configurations, outdated software versions, and other security issues in web applications. Nikto works by sending HTTP requests to the target server and evaluating responses for known signatures that match patterns of vulnerabilities.

Nikto supports testing for over 6,700 potentially dangerous files/programs, checks for outdated server components, and detects common security headers or the lack thereof. Despite being noisy and easily detected by intrusion detection systems, Nikto remains an effective initial enumeration tool for security assessments.

In this analysis, Nikto was used against the intentionally vulnerable website: `http://testphp.vulnweb.com` hosted on IP `44.228.249.3`.

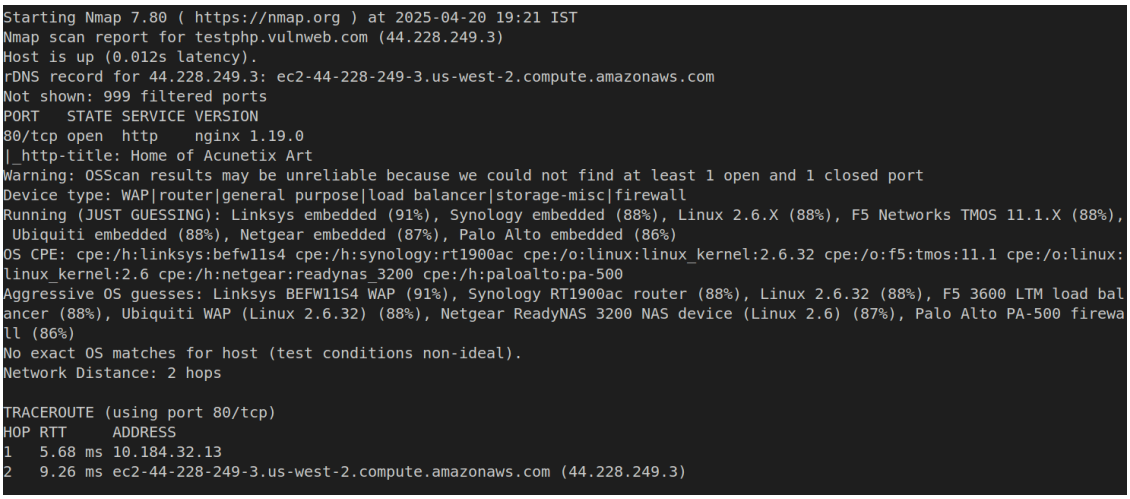


Figure 3: Nmap Vulnerability Detection Output

2. Vulnerabilities Tested and Found

Based on the scan results, the following key vulnerabilities and misconfigurations were identified:

- **Missing HTTP Security Headers:** The response lacks the anti-clickjacking `X-Frame-Options` header, increasing susceptibility to clickjacking attacks.
- **Leaking Server Metadata:** The server exposes its backend via the `X-Powered-By` header (PHP/5.6.40) and provides ETag values that could allow cache-based fingerprinting.
- **Wildcard Access to XML Policy Files:** Files like `/clientaccesspolicy.xml` and `/crossdomain.xml` allow wildcard domains. This introduces a high risk of cross-domain trust abuse or cross-site content leakage.
- **Directory Indexing Enabled:** Several directories including `/admin/` and `/images/` have directory listings enabled (OSVDB-3268), which could expose sensitive internal files.
- **Presence of Potential Backup/Repo Files:** Discovery of CVS entries or temporary files like `/CVS/Entries` might provide insight into source code or internal paths.
- **Injection Points with XSS and PHP Code Disclosure:** URLs with input parameters such as `index.php?dir=<script>` or `index.php?topic=<script>` indicate reflected Cross-Site Scripting (XSS) vulnerabilities.
- **Database String Disclosure Risks:** Multiple endpoints revealed patterns associated with exposed or test MySQL connection strings. These could disclose credentials or be used in SQL injection attempts.
- **Accessible Admin Login Page:** The page `/login.php` suggests an admin portal that could be brute-forced or probed.

3. Vulnerabilities Tested but Not Found

- **Shellshock (CVE-2014-6271):** Nikto did not detect vulnerability to Shellshock. This could be because:

- The server may not be using a vulnerable version of Bash.
- CGI scripts, which were typically exploited via Shellshock, may not be enabled or accessible on the target server.
- Modern web servers often have WAFs or input sanitization layers that neutralize Shellshock payloads.

- **TRACE Method Misuse (Cross-Site Tracing):** The HTTP TRACE method was not enabled on the target, possibly due to:
  - The server has been configured to disallow potentially dangerous HTTP methods by default.
  - Many modern web servers disable TRACE as a security best practice to prevent Cross-Site Tracing attacks.
  - Middleware or reverse proxies (like NGINX or load balancers) may be filtering unsupported methods before they reach the application layer.

4. Critical Vulnerabilities Found and Their Exploitation Potential

1. **Cross-Site Scripting (XSS):** Parameters like `index.php?dir=<script>` allow injection of JavaScript. An attacker could use this to hijack sessions, redirect users to malicious sites, or steal cookies via scripts such as:

```
<script>alert('Vulnerable')</script>
```

2. **Exposure of Database Connection Strings:** Parameters show signs of returning full or partial database connection strings. If exploited, attackers could use this information to attempt direct database access or enhance SQL injection payloads.

3. **Wildcard XML Files:** Allowing all domains in `crossdomain.xml` and `clientaccesspolicy.xml` may permit untrusted cross-domain access — especially dangerous in Flash- or Silverlight-based applications.

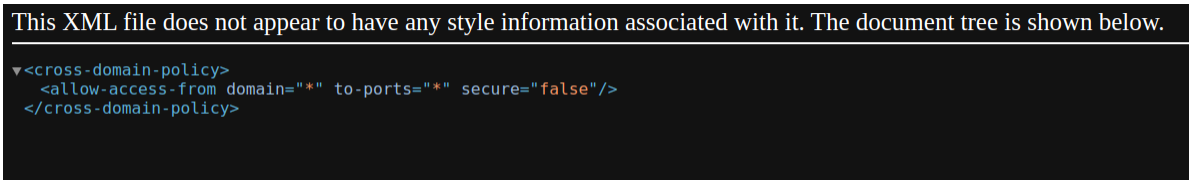


Figure 4: Wildcard XML file

4. **Directory Indexing:** Open directories like `/admin/` and `/images/` can unintentionally expose sensitive files (e.g., config backups, scripts, internal docs).

5. **Missing Security Headers:** The absence of security-related headers weakens client-side defenses. For example, browsers will not block attempts at clickjacking or simple XSS vectors.

5. Mitigation Strategies

- **Implement Secure HTTP Headers:** Add headers such as `X-Frame-Options`, `Content-Security-Policy`, and `X-XSS-Protection`.
- **Validate and Sanitize Inputs:** Implement strict server-side validation to avoid XSS, SQLi, or other injection attacks.
- **Restrict Directory Listing:** Disable directory indexing in server configuration (e.g., `Options -Indexes` in Apache).
- **Secure Admin Interfaces:** Protect access to `/login.php` using authentication, rate-limiting, and possibly IP whitelisting.
- **Avoid Wildcard Crossdomain Trust:** Restrict XML policy files to trusted, required domains only.
- **Remove Insecure Disclosure Headers:** Avoid exposing PHP or server versions via HTTP headers to reduce fingerprinting.

6. Scan Output and Screenshot

```
+ Server: nginx/1.19.0
+ X-Powered-By: PHP/5.6.40
+ Missing X-Frame-Options header
+ Leaks inodes via ETags
+ Wildcard domain access via /clientaccesspolicy.xml and /crossdomain.xml
+ /index.php?dir=<script>alert('Vulnerable')</script>
+ Directory Indexing on /admin/ and /images/
+ Potential PHP MySQL connection strings found in multiple GET parameters
+ /login.php: Admin login interface
```

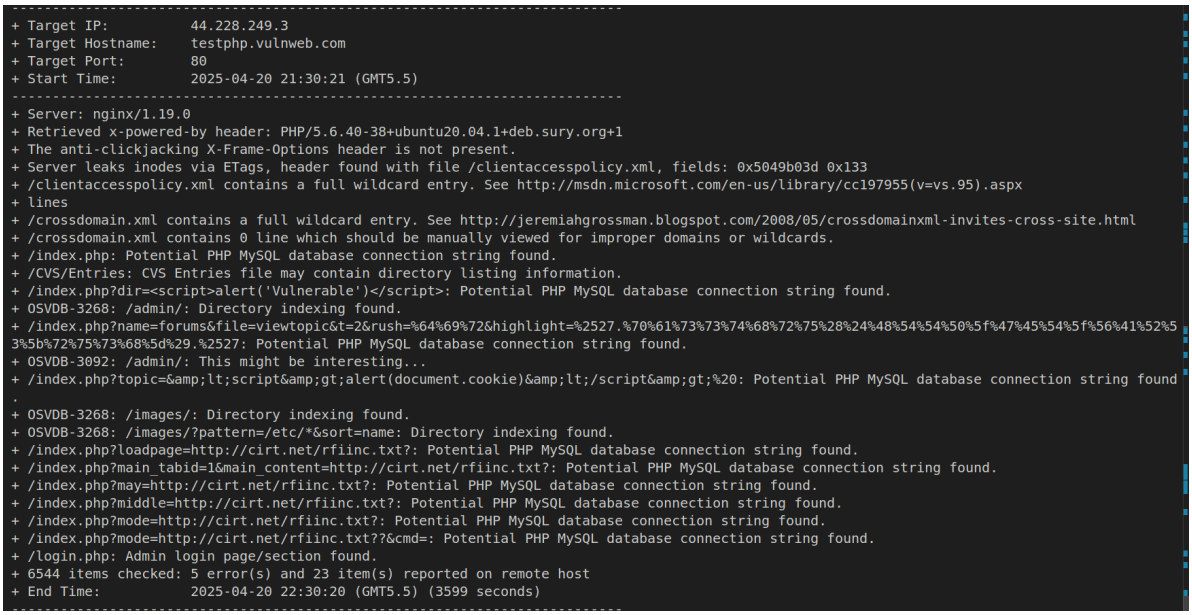


Figure 5: Nikto Scan Results Showing with command **nikto -h http://testphp.vulnweb.com**