PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Skikda — Department of Computer Science
Master's in Information Security — Semester 3 (M2) — UEF322
Web and Mobile Application Security
Updated on 2026-09-22 at 23:28:22

Lab 4 — Auditing a Website's TLS Configuration

Theoretical reminder

Chapter 4 distinguishes two things that are often confused: **enabling HTTPS** (having a valid certificate) and **configuring HTTPS properly** (choosing recent TLS versions, enabling HSTS, avoiding mixed content). A site can display a perfectly valid padlock while remaining vulnerable to *SSL stripping* or quietly loading a resource over HTTP. This lab takes you from theory ("TLS provides confidentiality, integrity, authentication") to the concrete audit of a real site, exactly as a security auditor would do before a penetration test.

Objective

Assess the robustness of a real site's HTTPS configuration and spot any weaknesses (see chapter 4).

Prerequisites

Having studied chapter 4: TLS handshake, X.509 certificates, PKI, HSTS, mixed content, SSL stripping.

Tools and installation

- `openssl`: usually preinstalled on Linux/macOS (`openssl version` to check); on Windows, install Git for Windows (`https://git-scm.com`), whose terminal includes `openssl`.
- `testssl.sh`: `git clone -depth 1 https://github.com/drwetter/testssl.sh.git` then `cd testssl.sh` – no compilation needed, it is a bash script.
- The online service **SSL Labs** (`https://www.ssllabs.com/ssltest/`), with no installation, as an alternative if `testssl.sh` is not available.

Test target

Only audit **your own site** or a site explicitly intended for this type of test. A TLS scan is generally harmless, but it is still an action sent to a third-party server: always ask for permission if in doubt.

Procedure

Step 1 — Inspect the certificate from the command line

```
openssl s_client -connect exemple.dz:443 -servername exemple.dz
```

In the output, find the `Certificate chain` block: identify the issuing certificate authority, the `notAfter` date (expiry), and the domain name covered (`CN` or `Subject Alternative Name`).

Step 2 — Automated audit

```
testssl.sh exemple.dz
```

Note in the report:

- the protocol versions accepted (are SSLv3 and TLS 1.0 still enabled? – if so, it is a weakness, chapter 4 section 4.2);
- whether *forward secrecy* is supported;
- whether the `Strict-Transport-Security` header is returned, and with which `max-age`.

Step 3 — Check for mixed content

Load the site in a browser, open the *Security* tab of the DevTools (Chrome) or the shield icon in the address bar (Firefox). Check whether any mixed-content resources (loaded over `http://` on an `https://` page) are reported.

Step 4 — Compare with a reference

Repeat steps 1 and 2 on a site known for its exemplary configuration (for example `google.com`), and list three differences observed with the site audited in steps 1–3.

Comprehension questions

1. Why does a valid padlock in the address bar not, on its own, guarantee a robust TLS configuration?

2. How does *forward secrecy* protect sessions intercepted and recorded *before* a future compromise of the server's private key?

3. Is a site without an HSTS header still vulnerable to SSL stripping even if all its internal links point to `https://`? Justify.

Deliverable

A comparison table (site tested – TLS versions enabled – HSTS present? – mixed content detected?) for the two sites audited, and a fix recommendation for each weakness identified on the first site.