|  |  |
| --- | --- |
|  | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**ĐỒ ÁN CHUYÊN NGÀNH**

**THIẾT KẾ VÀ TRIỂN KHAI CÔNG CỤ QUÉT TỰ ĐỘNG CHO CÁC SAI SÓT CẤU HÌNH MÃ HÓA TRÊN ỨNG DỤNG WEB**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **AN NINH MẠNG**

Giảng viên hướng dẫn : KS. Nguyễn Hoàng Duy

Sinh viên thực hiện :

Phạm Quốc Tường MSSV: 2280603627 Lớp:22DTHD7

Nguyễn Lưu Trọng Phước MSSV: 2280602502 Lớp: 22DTHD7

TP. Hồ Chí Minh, 2025

MỤC LỤC

**[MỤC LỤC I](#_Toc14290)**

**[DANH MỤC TỪ VIẾT TẮT IV](#_Toc32030)**

**[MỤC LỤC HÌNH VI](#_Toc27199)**

**[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI 1](#_Toc1239)**

[1.1 Lý do chọn đề tài 1](#_Toc22081)

[1.1.1 Bối cảnh và tầm quan trọng của HTTPS/TLS 1](#_Toc27521)

[1.1.2 Vấn đề thực tế và nhu cầu cấp thiết 1](#_Toc17212)

[1.1.3 Tính ứng dụng của công cụ quét tự động 2](#_Toc6601)

[1.2 Mục tiêu đề tài 2](#_Toc20566)

[1.2.1 Mục tiêu tổng quát 2](#_Toc17344)

[1.2.2 Mục tiêu cụ thể 2](#_Toc4931)

[1.2.3 Kết quả mong đợi 3](#_Toc6817)

[1.3 Phạm vi đề tài 3](#_Toc2372)

**[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 4](#_Toc15319)**

[2.1 Tổng quan về giao thức HTTP và HTTPS 4](#_Toc17046)

[2.1.1 HTTP 4](#_Toc17701)

[2.1.2 HTTPS 4](#_Toc25220)

[2.2 Chứng chỉ số và vai trò trong HTTPS 5](#_Toc7921)

[2.2.1 Chứng chỉ số (SSL/TLS Certificate) 5](#_Toc24830)

[2.2.2 Chuỗi chứng chỉ (Certificate Chain) 6](#_Toc22020)

[2.3 Tìm hiểu về các lỗi sai sót cấu hình 7](#_Toc8669)

[2.3.1 HSTS (HTTP Strict-Transport-Security) 7](#_Toc26690)

[2.3.2 Dùng phiên bản TLS đã lỗi thời 7](#_Toc22326)

[2.3.3 Sử dụng bộ cipher yếu 8](#_Toc17442)

[2.3.4 Thiếu thuộc tính bảo mật cho cookie 8](#_Toc3954)

[2.3.5 Chứng chỉ số không hợp 9](#_Toc30965)

[2.3.6 Thiếu các HTTP security headers quan trọng 9](#_Toc5383)

[2.4 Các tiêu chuẩn và thực hành tốt trong cấu hình HTTPS/TLS 10](#_Toc17969)

[2.4.1 Tiêu chuẩn TLS hiện đại 10](#_Toc13567)

[2.4.2 Cấu hình HSTS 10](#_Toc27237)

[2.4.3 Cấu hình HTTP headers 11](#_Toc27314)

[2.5 Phân loại mức độ nghiêm trọng của sai sót cấu hình 11](#_Toc22145)

[2.6 Vai trò của công cụ quét tự động trong bảo mật web 12](#_Toc8800)

[2.6.1 Phát hiện và phân tích sai sót cấu hình 12](#_Toc20623)

[2.6.2 Tự động hóa quy trình kiểm tra 12](#_Toc4494)

[2.6.3 Hỗ trợ giáo dục và nghiên cứu 12](#_Toc10521)

**[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 13](#_Toc23379)**

[3.1 Kiến trúc tổng quan hệ thống 13](#_Toc200)

[3.2 Mô tả hoạt động 13](#_Toc10226)

[3.3 Kịch bản thực hiện đề tài 15](#_Toc18438)

[3.3.1 Chuẩn bị môi trường và công cụ 15](#_Toc21118)

[3.3.2 Thiết kế cấu trúc dự án 15](#_Toc12459)

[3.3.3 Triển khai chức năng quét HTTP 16](#_Toc9301)

[3.3.4 Phân tích TLS và Cipher Suites 16](#_Toc18931)

[3.3.5 Kiểm tra chứng chỉ số 16](#_Toc5556)

[3.3.6 Phân tích Header và Cookie 16](#_Toc342)

[3.3.7 Đánh giá rủi ro 17](#_Toc948)

[3.3.8 Xuất báo cáo kết quả 17](#_Toc22266)

[3.3.9 Đánh giá kết quả 17](#_Toc3956)

[3.3.10 Trình bày và hoàn thiện đồ án 17](#_Toc19158)

**[CHƯƠNG 4: TRIỀN KHAI HỆ THỐNG 18](#_Toc14133)**

[4.1 Công nghệ và công cụ sử dụng 18](#_Toc28602)

[4.1.1 Ngôn ngữ lập trình 18](#_Toc434)

[4.1.2 Công cụ hỗ trợ phát triển 18](#_Toc15327)

[4.1.3 Môi trường máy chủ kiểm thử 18](#_Toc5385)

[4.2 Thư viện 18](#_Toc10922)

[4.2.1 Thư viện lõi (Core Dependencies) 18](#_Toc7642)

[4.2.2 Thư viện phân tích TLS/SSL 19](#_Toc8640)

[4.2.3 Thư viện kiểm thử (Testing) 19](#_Toc20630)

[4.2.4 Thư viện xử lý dữ liệu và cấu hình 19](#_Toc23468)

[4.2.5 Thư viện phục vụ Web Crawler 19](#_Toc24083)

[4.2.6 Thư viện tiện ích 20](#_Toc27799)

[4.3 Kết quả chương trình 20](#_Toc15830)

[4.3.1 Quét cấu hình https://www.youtube.com/ 20](#_Toc23014)

[4.3.2 Quét cấu hình https://example.com/ 22](#_Toc961)

[4.3.3 Quét các domain để kiểm tra chứng chỉ số 25](#_Toc17345)

**[CHƯƠNG 5: ĐÁNH GIÁ VÀ PHÂN TÍCH KẾT QUẢ 28](#_Toc25594)**

[5.1 Đánh giá 28](#_Toc22477)

[5.2 Phân tích kết quả 28](#_Toc24397)

[5.2.1 Kịch bản và môi trường thử nghiệm 28](#_Toc16021)

[5.2.2 Kết quả thu được 28](#_Toc9044)

**[CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 31](#_Toc386)**

[6.1 Kết luận 31](#_Toc17704)

[6.2 Hướng phát triển 31](#_Toc5458)

**[TÀI LIỆU THAM KHẢO 33](#_Toc29299)**

DANH MỤC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Từ đầy đủ** | **Ý nghĩa** |
| **1** | HTTP | HyperText Transfer Protocol | Giao thức truyền tải dữ liệu không mã hóa. |
| **2** | HTTPS | HyperText Transfer Protocol Secure | HTTP kết hợp TLS để mã hóa. |
| **3** | TLS | Transport Layer Security | Giao thức mã hóa bảo vệ dữ liệu khi truyền tải. |
| **4** | SSL | Secure Sockets Layer | Phiên bản tiền nhiệm của TLS, đã lỗi thời. |
| **5** | CA | Certificate Authority | Tổ chức phát hành chứng chỉ số. |
| **6** | DV | Domain Validation | Chứng chỉ xác thực tên miền. |
| **7** | OV | Organization Validation | Chứng chỉ xác thực tổ chức. |
| **8** | EV | Extended Validation | Chứng chỉ xác thực mở rộng, độ tin cậy cao. |
| **9** | CSP | Content-Security-Policy | Header bảo mật kiểm soát nguồn tài nguyên. |
| **10** | HSTS | HTTP Strict Transport Security | Buộc trình duyệt sử dụng HTTPS. |
| **11** | MITM | Man-in-the-Middle | Tấn công nghe lén/chèn sửa dữ liệu. |
| **12** | CSRF | Cross-Site Request Forgery | Giả mạo yêu cầu người dùng gửi đi. |
| **13** | XSS | Cross-Site Scripting | Chèn mã độc chạy trên trình duyệt. |
| **14** | SAN | Subject Alternative Name | Danh sách tên miền trong chứng chỉ TLS. |
| **15** | RSA | Rivest–Shamir–Adleman | Thuật toán mật mã khóa công khai. |
| **16** | ECDHE | Elliptic Curve Diffie-Hellman Ephemeral | Thuật toán trao đổi khóa hỗ trợ Forward Secrecy. |
| **17** | PFS | Perfect Forward Secrecy | Đảm bảo khóa phiên không bị lộ khi khóa chính bị lộ. |
| **18** | CSV | Comma-Separated Values | Định dạng dữ liệu dạng bảng. |
| **19** | JSON | JavaScript Object Notation | Định dạng dữ liệu phổ biến trong API. |
| **20** | HTML | HyperText Markup Language | Ngôn ngữ để xây dựng trang web. |
| **21** | XML | Extensible Markup Language | Định dạng dữ liệu có cấu trúc. |
| **22** | UI | User Interface | Giao diện người dùng. |
| **23** | CLI | Command Line Interface | Giao diện dòng lệnh. |
| **24** | API | Application Programming Interface | Giao diện lập trình ứng dụng. |
| **25** | URL | Uniform Resource Locator | Địa chỉ định vị tài nguyên trên Internet. |
| **26** | TXT | Text File | Tệp văn bản thuần. |

MỤC LỤC HÌNH

[Hình 2.1: Mô hình hoạt động của HTTP 4](#_Toc509)

[Hình 2.2: So sánh cơ chế truyền dữ liệu giữa HTTP và HTTPS 6](#_Toc2019)

[Hình 3.1: Workflow 13](#_Toc23819)

[Hình 4.1: Phần TLS khi quét url https://www.youtube.com/ 20](#_Toc5687)

[Hình 4.2: Phần phát hiện cấu hình khi quét url https://www.youtube.com/ 21](#_Toc14649)

[Hình 4.3: Phần TLS và header khi quét url https://example.com/ 23](#_Toc15187)

[Hình 4.4: Phần phát hiện và gợi ý khi quét url https://example.com/ 24](#_Toc12186)

[Hình 4.5: Quét chứng chỉ của domain expired.badssl.com 26](#_Toc28469)

[Hình 4.6: Quét chứng chỉ của domain self-signed.badssl.com 27](#_Toc30338)

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

* 1. Lý do chọn đề tài
     1. Bối cảnh và tầm quan trọng của HTTPS/TLS

Trong thời đại số hóa, các giao dịch trực tuyến, từ mua sắm, ngân hàng đến các dịch vụ công, đều yêu cầu mức độ bảo mật cao để bảo vệ thông tin cá nhân và dữ liệu nhạy cảm của người dùng. HTTPS với nền tảng là giao thức TLS (Transport Layer Security), đã trở thành tiêu chuẩn bảo mật không thể thiếu cho các website.

Tuy nhiên, không phải hệ thống nào cũng được cấu hình đúng chuẩn. Nhiều trang web vẫn dùng phiên bản TLS cũ, thiếu các header bảo mật quan trọng hoặc thiết lập cookie chưa đúng, khiến dữ liệu có nguy cơ bị tấn công hoặc đánh cắp trong quá trình truyền tải. Vì vậy, em quyết định chọn đề tài xây dựng một bộ công cụ tự động để kiểm tra các sai sót phổ biến trong cấu hình bảo mật của website.

Công cụ này không chỉ giúp người quản trị nhận biết sớm các điểm yếu mà còn hỗ trợ sinh viên và người học nắm rõ hơn cách thức hoạt động của HTTP, TLS và các tiêu chuẩn bảo mật đi kèm.

* + 1. Vấn đề thực tế và nhu cầu cấp thiết

Một số vấn đề phổ biến liên quan đến cấu hình HTTPS/TLS bao gồm:

* Sử dụng phiên bản TLS lỗi thời: TLS 1.0 và 1.1 đã bị coi là không an toàn và không còn được các trình duyệt hiện đại hỗ trợ.
* Cipher suite yếu: Các thuật toán mã hóa như RC4, 3DES vẫn được sử dụng trong một số hệ thống cũ, dễ bị tấn công brute-force.
* Thiếu các header bảo mật quan trọng: HSTS (HTTP Strict Transport Security) giúp ngăn chặn downgrade attack, nhưng nhiều website vẫn chưa triển khai hoặc cấu hình sai.
* Cấu hình cookie không đúng chuẩn: Cookie thiếu các thuộc tính như Secure, HttpOnly, SameSite có thể dẫn đến các cuộc tấn công như session hijacking hoặc cross-site request forgery (CSRF).
* Chứng chỉ TLS không hợp lệ: Chứng chỉ hết hạn, không đáng tin cậy hoặc không khớp với hostname có thể khiến người dùng gặp rủi ro khi truy cập.

Những vấn đề này không chỉ ảnh hưởng đến tính bảo mật của hệ thống mà còn làm giảm uy tín của tổ chức/doanh nghiệp, đặc biệt trong bối cảnh các quy định như GDPR yêu cầu nghiêm ngặt về bảo vệ dữ liệu cá nhân.

1.1.3 Tính ứng dụng của công cụ quét tự động

Việc xây dựng một công cụ tự động để kiểm tra các sai sót trong cấu hình HTTPS/TLS không chỉ giúp người quản trị hệ thống phát hiện và khắc phục sớm các lỗ hổng mà còn hỗ trợ sinh viên, nhà nghiên cứu hiểu rõ hơn về các tiêu chuẩn bảo mật. Công cụ này có thể được sử dụng trong các khóa học về an toàn thông tin, các dự án nghiên cứu hoặc trong thực tế để kiểm tra và cải thiện cấu hình bảo mật của các website.

1.2 Mục tiêu đề tài

1.2.1 Mục tiêu tổng quát

Xây dựng một công cụ tự động để kiểm tra và phát hiện các sai sót phổ biến trong cấu hình HTTPS/TLS của các website, từ đó đưa ra các khuyến nghị cải thiện nhằm nâng cao mức độ an toàn của hệ thống.

1.2.2 Mục tiêu cụ thể

**Phát triển công cụ quét tự động:**

* Xây dựng một ứng dụng Python hỗ trợ cả giao diện dòng lệnh (CLI) và giao diện web (UI).
* Công cụ có khả năng quét danh sách các URL và phân tích các vấn đề liên quan đến cấu hình HTTPS/TLS.

**Phát hiện và phân tích các sai sót bảo mật:**

* Kiểm tra các header HTTP quan trọng như Strict-Transport-Security (HSTS), X-Content-Type-Options, Referrer-Policy.
* Kiểm tra thuộc tính cookie: Secure, HttpOnly, SameSite.
* Phân tích phiên TLS: phiên bản, cipher suite, chứng chỉ TLS (hợp lệ, hết hạn, chuỗi chứng chỉ).

**Báo cáo kết quả và đề xuất cải thiện:**

* Tổng hợp kết quả quét dưới dạng bảng, báo cáo chi tiết (CSV, JSON, HTML).
* Xếp hạng mức độ nghiêm trọng của các phát hiện và đưa ra khuyến nghị sửa chữa.

**Tạo môi trường thử nghiệm và demo:**

* Xây dựng lab thử nghiệm để kiểm tra công cụ.
* Cung cấp mã nguồn, tài liệu hướng dẫn và báo cáo chi tiết để người dùng dễ dàng sử dụng và kiểm chứng.

1.2.3 Kết quả mong đợi

Một công cụ quét tự động hoàn chỉnh, dễ sử dụng, có khả năng phát hiện các sai sót phổ biến trong cấu hình HTTPS/TLS.

Báo cáo chi tiết về các phát hiện, kèm theo các khuyến nghị cụ thể để cải thiện cấu hình bảo mật.

Tài liệu hướng dẫn sử dụng và mã nguồn mở để cộng đồng có thể đóng góp và phát triển thêm.

1.3 Phạm vi đề tài

* **Bao gồm:**

Kiểm tra headers HTTP (Strict-Transport-Security, Public-Key-Pins nếu có, X-Content-Type-Options, Referrer-Policy).

Kiểm tra thuộc tính cookie: Secure, HttpOnly, SameSite.

Kiểm tra phiên TLS: phiên bản TLS, cipher suite (phát hiện TLS1.0/1.1, ciphers yếu như RC4, 3DES, EXPORT), chuỗi chứng chỉ (valid, expiry), hostname mismatches.

Tự động hóa quét theo danh sách URLs (single + domain crawl basic).

Báo cáo kết quả (CSV/JSON/HTML) và xếp hạng mức độ nghiêm trọng.

* **Không bao gồm:**

Exploit hay tấn công thực tế vào dịch vụ (vượt phạm vi pháp lý).

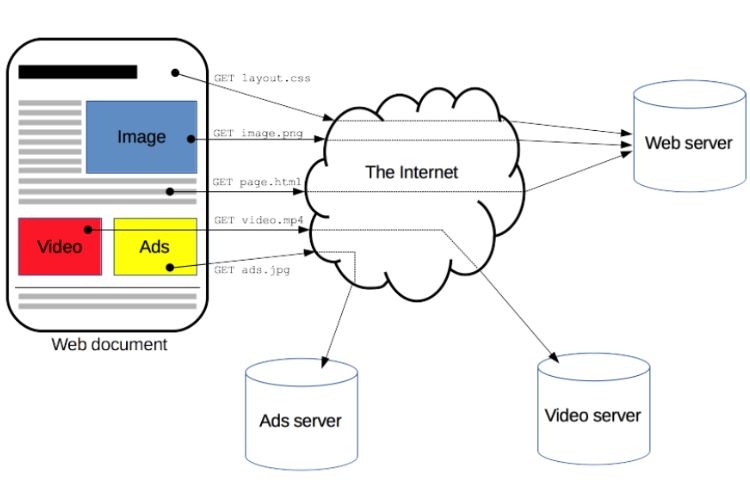
Quét toàn mạng internet (chỉ lab / được phép).

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về giao thức HTTP và HTTPS

2.1.1 HTTP

HTTP là giao thức truyền tải siêu văn bản, được sử dụng để trao đổi dữ liệu giữa trình duyệt và máy chủ web. Đây là giao thức không mã hóa, nghĩa là dữ liệu được truyền tải dưới dạng plaintext, dễ bị nghe lén hoặc sửa đổi bởi kẻ tấn công [1].



Hình 2.1: Mô hình hoạt động của HTTP

* **Đặc điểm của HTTP:**
  + Không mã hóa dữ liệu.
  + Không đảm bảo tính toàn vẹn của dữ liệu.
  + Không xác thực danh tính của máy chủ.
* **Nhược điểm của HTTP:**
  + Dễ bị tấn công man-in-the-middle (MITM).
  + Dữ liệu nhạy cảm như mật khẩu, thông tin cá nhân có thể bị đánh cắp.

2.1.2 HTTPS

HTTP/TLS: là phiên bản an toàn của HTTP, trong đó dữ liệu được mã hóa bởi giao thức TLS trước khi truyền đi qua mạng [2].

* **HTTPS đảm bảo:**
  + Bảo mật dữ liệu (Confidentiality): Dữ liệu được mã hóa, kẻ tấn công không đọc được.
  + Xác thực (Authentication): Trình duyệt có thể xác minh đúng server thông qua chứng chỉ số (SSL/TLS certificate).
  + Toàn vẹn dữ liệu (Integrity): Dữ liệu không bị sửa đổi trong quá trình truyền.
* **Ưu điểm của HTTPS:**
  + Bảo mật dữ liệu (Confidentiality): Dữ liệu được mã hóa, ngăn chặn việc bị nghe lén.
  + Xác thực (Authentication): Đảm bảo rằng người dùng đang giao tiếp với đúng máy chủ thông qua chứng chỉ số.
  + Toàn vẹn dữ liệu (Integrity): Đảm bảo dữ liệu không bị sửa đổi trong quá trình truyền tải.
* **Nguyên lý:**

- Khi người dùng truy cập trang HTTPS, trình duyệt và server thực hiện quy trình TLS Handshake để thiết lập kết nối bảo mật:

- Quy trình Handshake cơ bản:

* + Client gửi yêu cầu bắt đầu kết nối: Trình duyệt gửi danh sách phiên bản TLS và Cipher Suite mà nó hỗ trợ.
  + Server phản hồi và gửi chứng chỉ: Server chọn phiên bản TLS, Cipher Suite và gửi chứng chỉ số (SSL/TLS Certificate) cho client.
  + Client kiểm tra chứng chỉ: Kiểm tra chứng chỉ có hợp lệ không (thời hạn, chữ ký, CA, hostname).
    - Nếu hợp lệ → tiếp tục.
    - Nếu không → trình duyệt cảnh báo “Kết nối không an toàn”.
  + Trao đổi khóa mã hóa: Client và server thực hiện thuật toán trao đổi khóa (thường là ECDHE) để tạo ra khóa bí mật chung.
  + Bắt đầu phiên mã hóa: Tất cả dữ liệu HTTP sau đó sẽ được mã hóa bằng khóa chung này.

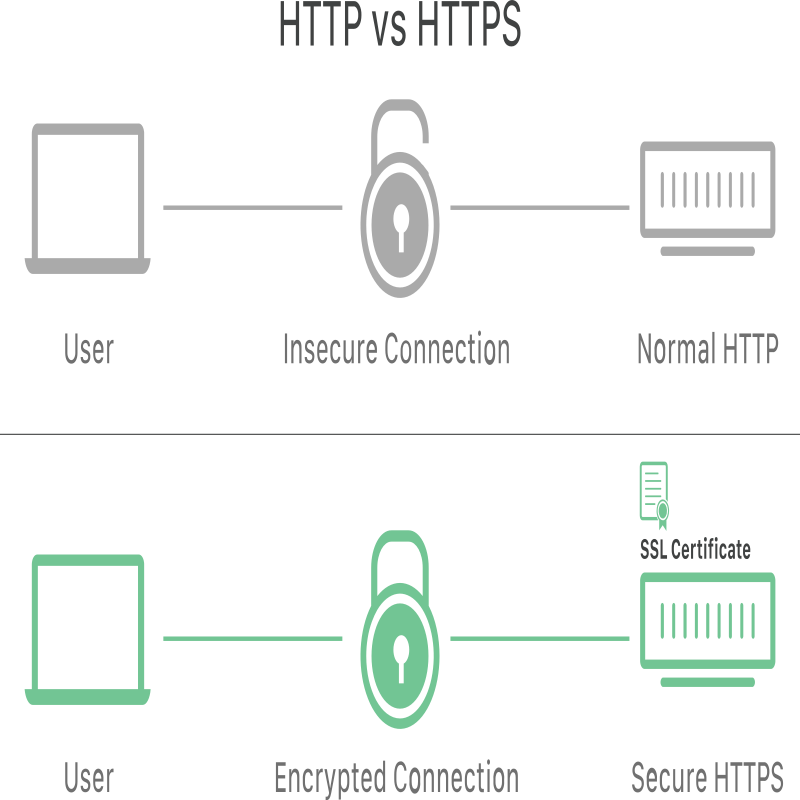
**→** Từ thời điểm này trở đi, HTTP được truyền trong dạng mã hóa, gọi là HTTPS.

2.2 Chứng chỉ số và vai trò trong HTTPS

2.2.1 Chứng chỉ số (SSL/TLS Certificate)

SSL, hay Secure Sockets Layer, là một giao thức bảo mật Internet dựa trên mã hóa . Giao thức này được Netscape phát triển lần đầu tiên vào năm 1995 nhằm mục đích đảm bảo quyền riêng tư, xác thực và tính toàn vẹn dữ liệu trong giao tiếp Internet. SSL là tiền thân của mã hóa TLS hiện đại được sử dụng ngày nay [2].

Chứng chỉ số là một tệp dữ liệu được cấp bởi một tổ chức chứng thực (Certificate Authority - CA), dùng để xác thực danh tính của máy chủ và thiết lập kết nối an toàn.



Hình 2.2: So sánh cơ chế truyền dữ liệu giữa HTTP và HTTPS

* **Thành phần của chứng chỉ số:**
  + Tên miền của website.
  + Tên của tổ chức sở hữu chứng chỉ.
  + Ngày hết hạn của chứng chỉ.
  + Khóa công khai (Public Key).
  + Chữ ký số của CA.
* **Các loại chứng chỉ số:**
  + Domain Validation (DV): Xác thực tên miền.
  + Organization Validation (OV): Xác thực tổ chức sở hữu tên miền.
  + Extended Validation (EV): Xác thực mở rộng, cung cấp mức độ tin cậy cao nhất.

2.2.2 Chuỗi chứng chỉ (Certificate Chain)

Chuỗi chứng chỉ là tập hợp các chứng chỉ được sử dụng để xác thực danh tính của máy chủ. Chuỗi này bao gồm:

* Chứng chỉ gốc (Root Certificate): Được cấp bởi CA gốc.
* Chứng chỉ trung gian (Intermediate Certificate): Được cấp bởi CA trung gian.
* Chứng chỉ máy chủ (Server Certificate): Được cấp cho máy chủ web.

2.3 Tìm hiểu về các lỗi sai sót cấu hình

Mặc dù HTTPS đã trở thành tiêu chuẩn đối với các website hiện nay, cách mỗi hệ thống triển khai lại khác nhau khá nhiều. Nếu cấu hình không đúng, cơ chế mã hóa gần như mất tác dụng, khiến dữ liệu người dùng dễ bị nghe lén, sửa đổi hoặc giả mạo. Những lỗi cấu hình phổ biến có thể được phân loại như sau:

2.3.1 HSTS (HTTP Strict-Transport-Security)

- HSTS là một header HTTP (Strict-Transport-Security) được server trả về để yêu cầu trình duyệt chỉ kết nối bằng HTTPS với domain đó trong một khoảng thời gian xác định [3]. Khi website không bật HSTS hoặc thiết lập không hợp lý các website sẽ dễ bị tấn công như:

* **Tấn công hạ cấp:** Buộc kết nối chuyển từ HTTPS xuống HTTP.
* **SSL stripping:** Dữ liệu bị lộ vì bị ép truyền qua giao thức không mã hóa.

- Một số sai sót thường thấy:

* Không cấu hình header Strict-Transport-Security.
* Thời gian max-age quá thấp.
* Không đăng ký HSTS preload với trình duyệt.

- Rủi ro:

* Hacker có thể “chặn” yêu cầu đầu tiên và chuyển hướng sang HTTP, khiến dữ liệu truyền đi không còn được mã hóa.
* Một dạng tấn công phổ biến là SSL Stripping, nơi kẻ tấn công bí mật hạ cấp kết nối và đọc toàn bộ dữ liệu.
* Người dùng khó phát hiện vì trang giả mạo thường hiển thị giống hệt trang thật.

- Hậu quả:

* Lộ thông tin đăng nhập và cookie phiên.
* Dễ bị dẫn sang website giả mạo phục vụ mục đích lừa đảo.

2.3.2 Dùng phiên bản TLS đã lỗi thời

- TLS là giao thức quan trọng trong việc mã hóa dữ liệu khi truyền qua Internet. Tuy nhiên những phiên bản cũ như TLS 1.0 và TLS 1.1 đã bị coi là không an toàn.

- Các lỗi phổ biến:

* Hệ thống vẫn cho phép TLS 1.0/1.1 hoạt động.
* Chỉ sử dụng RSA key exchange và không có Perfect Forward Secrecy (PFS).

- Rủi ro nếu vẫn cho phép TLS 1.0/1.1:

* Một số lỗ hổng như BEAST hay POODLE cho phép hacker từng bước giải mã dữ liệu.
* Không hỗ trợ Perfect Forward Secrecy, khiến lưu lượng đã ghi lại trước đó có thể bị giải mã nếu khóa máy chủ bị lộ.
* Một số trình duyệt hiện đại sẽ hiển thị cảnh báo hoặc từ chối kết nối.

- Hậu quả:

* Dữ liệu nhạy cảm có thể bị đọc lén.
* Website bị đánh giá bảo mật thấp khi kiểm thử, ảnh hưởng uy tín.

2.3.3 Sử dụng bộ cipher yếu

- Cipher suite quyết định cách dữ liệu được mã hóa. Nếu website vẫn cho phép những thuật toán cũ như RC4, 3DES hay các cipher EXPORT thì dữ liệu có thể bị giải mã dễ dàng.

- Rủi ro khi vẫn bật các cipher yếu:

* RC4 có thể bị phá chỉ trong thời gian ngắn.
* 3DES dễ bị tấn công Sweet32, gây rò rỉ dữ liệu khi truyền tải lượng lớn.
* Các cipher EXPORT có khóa quá ngắn và dễ bị brute-force.

- Hậu quả:

* Tấn công đọc cookie, giải mã dữ liệu hoặc chiếm phiên người dùng.
* Website bị đánh giá kém trên các công cụ kiểm tra bảo mật.

2.3.4 Thiếu thuộc tính bảo mật cho cookie

- Cookie lưu trữ trạng thái phiên bản đăng nhập. Nếu không cấu hình đúng, kẻ tấn công có thể đánh cắp cookie và chiếm quyền tài khoản [4].

- Các sai sót thường gặp:

* Không đặt Secure, cookie vẫn gửi qua HTTP dễ bị MITM lấy cắp.
* Không có HttpOnly, JavaScript có thể đọc cookie dẫn đến nguy cơ XSS.
* Không có SameSite dẫn đến dễ bị tấn công CSRF.
* Cookie chứa dữ liệu nhạy cảm dạng plaintext.

- Hậu quả:

* Bị chiếm tài khoản mà người dùng không hề biết.
* Hacker có thể thực hiện hành động thay người dùng

2.3.5 Chứng chỉ số không hợp

**-** Chứng chỉ số là “ giấy chứng nhận danh tính ” của website. Nếu chứng chỉ có vấn đề, trình duyệt sẽ cảnh báo và người dùng sẽ khó mà tin tưởng trang web đó.

- Các lỗi phổ biến:

* Chứng chỉ quá hạn.
* Tên miền trên cert không khớp.
* Chuỗi chứng chỉ.
* Độ dài khóa thấp (1024 bit hoặc thấp hơn).

- Rủi ro của chứng chỉ không hợp lệ:

* Cert hết hạn khiến trình duyệt hiển thị cảnh báo và dễ bị MITM mạo danh.
* Tên miền trên cert không khớp với website thật, hacker dễ chèn chứng chỉ giả.
* Chuỗi chứng chỉ không đầy đủ khiến một số thiết bị từ chối kết nối.
* Self-signed certificate nếu dùng trong môi trường thực tế rất dễ bị giả mạo.

- Hậu quả:

* Người dùng e ngại khi truy cập website.
* Doanh nghiệp đánh mất uy tín và có thể bị gián đoạn dịch vụ.

2.3.6 Thiếu các HTTP security headers quan trọng

- Ngoài HSTS, còn nhiều header bảo mật giúp giảm nguy cơ tấn công nhưng hay bị bỏ sót:

* X-Content-Type-Options: nosniff
* X-Frame-Options: DENY hoặc SAMEORIGIN
* Referrer-Policy
* Content-Security-Policy

- Rủi ro khi thiếu các header quan trọng:

* Không có X-Content-Type-Options, hacker có thể ép trình duyệt chạy file độc hại.
* Không bật X-Frame-Options, website có thể bị clickjacking.
* Không có Referrer-Policy, token hoặc đường dẫn nhạy cảm bị lộ qua URL.
* Không có Content-Security-Policy, XSS rất dễ xảy ra.

- Hậu quả:

* Website bị chèn mã độc.
* Người dùng bị lừa tương tác vào giao diện giả.
* Dữ liệu nhạy cảm vô tình bị rò rỉ.

2.4 Các tiêu chuẩn và thực hành tốt trong cấu hình HTTPS/TLS

2.4.1 Tiêu chuẩn TLS hiện đại

TLS (Transport Layer Security) là giao thức mã hóa được sử dụng để bảo vệ dữ liệu truyền tải qua mạng. Tuy nhiên, không phải tất cả các phiên bản TLS đều an toàn. Các tiêu chuẩn hiện đại yêu cầu chỉ hỗ trợ các phiên bản TLS mới nhất, cụ thể là TLS 1.2 và TLS 1.3, đồng thời loại bỏ các phiên bản cũ và các thuật toán mã hóa yếu.

* **Chỉ hỗ trợ TLS 1.2 và TLS 1.3:**

- TLS 1.2 và TLS 1.3 là hai phiên bản được khuyến nghị sử dụng hiện nay. TLS 1.2 đã được sử dụng rộng rãi trong hơn một thập kỷ và vẫn đảm bảo an toàn nếu được cấu hình đúng. TLS 1.3, ra mắt vào năm 2018, mang lại nhiều cải tiến về bảo mật và hiệu suất, bao gồm việc loại bỏ các thuật toán mã hóa lỗi thời và giảm thời gian thiết lập kết nối (handshake).

* **Loại bỏ các cipher suite yếu:**

**-** Cipher suite là tập hợp các thuật toán mã hóa được sử dụng trong quá trình thiết lập kết nối TLS. Một số cipher suite cũ như RC4 và 3DES đã bị phát hiện có lỗ hổng bảo mật nghiêm trọng, khiến chúng dễ bị tấn công brute-force hoặc giải mã. Các tiêu chuẩn hiện đại yêu cầu loại bỏ hoàn toàn các cipher suite này và chỉ sử dụng các thuật toán mã hóa mạnh.

* **Vô hiệu hóa các giao thức lỗi thời:**

- Các giao thức cũ như SSL 3.0, TLS 1.0 và TLS 1.1 không còn an toàn và đã bị hầu hết các trình duyệt hiện đại ngừng hỗ trợ. Việc vô hiệu hóa các giao thức này trên máy chủ là một bước quan trọng để ngăn chặn các cuộc tấn công như POODLE và BEAST.

2.4.2 Cấu hình HSTS

* **Sử dụng Strict-Transport-Security với max-age=31536000:**

- Header Strict-Transport-Security cần được cấu hình với giá trị max-age đủ lớn để đảm bảo trình duyệt ghi nhớ chính sách HSTS trong thời gian dài.

* **Đăng ký HSTS preload:**

- HSTS preload là một danh sách các website được trình duyệt hiện đại (Chrome, Firefox, Edge, Safari) tự động áp dụng HSTS ngay từ lần đầu tiên người dùng truy cập. Điều này giúp bảo vệ người dùng khỏi các cuộc tấn công SSL stripping ngay cả khi họ chưa từng truy cập website trước đó.

* **Lợi ích của HSTS:**

- Ngăn chặn hoàn toàn các cuộc tấn công SSL stripping.

- Bảo vệ người dùng khỏi việc vô tình truy cập vào phiên bản HTTP không an toàn của website.

- Tăng cường uy tín và độ tin cậy của website.

2.4.3 Cấu hình HTTP headers

HTTP headers đóng vai trò quan trọng trong việc bảo vệ ứng dụng web khỏi các cuộc tấn công phổ biến như XSS, clickjacking, và rò rỉ dữ liệu. Dưới đây là các HTTP headers bảo mật cần được cấu hình:

* **Strict-Transport-Security:** header này buộc trình duyệt chỉ kết nối với máy chủ qua HTTPS.
* **X-Content-Type-Options:** Header này ngăn chặn trình duyệt tự động đoán định kiểu nội dung (MIME type sniffing), giảm nguy cơ tấn công XSS. Cấu hình khuyến nghị: X-Content-Type-Options: nosniff.
* **Content-Security-Policy (CSP):** CSP là một header mạnh mẽ giúp kiểm soát các nguồn tài nguyên mà trang web có thể tải. CSP ngăn chặn các cuộc tấn công XSS bằng cách chỉ cho phép tải các script từ các nguồn đáng tin cậy.
* **X-Frame-Options:** Header này ngăn chặn clickjacking bằng cách kiểm soát việc nhúng trang web trong iframe. Cấu hình khuyến nghị: X-Frame-Options: DENY.
* Ngoài những header nêu trên, hệ thống còn có thể sử dụng thêm nhiều header bảo mật khác.
* Lợi ích của việc cấu hình HTTP headers:
  + Giảm nguy cơ bị tấn công XSS, clickjacking, và rò rỉ dữ liệu.
  + Tăng cường bảo mật tổng thể của ứng dụng web.
  + Đáp ứng các tiêu chuẩn bảo mật hiện đại và yêu cầu pháp lý (như GDPR).

2.5 Phân loại mức độ nghiêm trọng của sai sót cấu hình

Công cụ quét có thể gán mức độ rủi ro như sau:

**- Critical:**

* Cert hết hạn.
* Cert sai domain.
* Cho phép TLS 1.0 / SSL 3.0.
* Không có mã hóa.

**- High:**

* Không bật HSTS.
* Cho phép cipher yếu (RC4, 3DES).
* Cookie không Secure hoặc HttpOnly.

**- Medium:**

* Không có CSP.
* Không bật X-Frame-Options.

**- Low:**

* Lộ server banner.
* Header không mang tính bảo mật cao.

2.6 Vai trò của công cụ quét tự động trong bảo mật web

2.6.1 Phát hiện và phân tích sai sót cấu hình

Công cụ quét tự động giúp phát hiện các sai sót cấu hình HTTPS/TLS và HTTP headers một cách nhanh chóng và chính xác. Điều này giúp người quản trị hệ thống:

* Xác định các vấn đề bảo mật tiềm ẩn.
* Đánh giá mức độ nghiêm trọng của từng vấn đề.
* Đưa ra các khuyến nghị cụ thể để cải thiện cấu hình.

2.6.2 Tự động hóa quy trình kiểm tra

So với việc kiểm tra thủ công, công cụ quét tự động mang lại nhiều lợi ích:

* Tiết kiệm thời gian và công sức.
* Giảm thiểu sai sót do con người.
* Hỗ trợ kiểm tra trên nhiều URL cùng lúc.

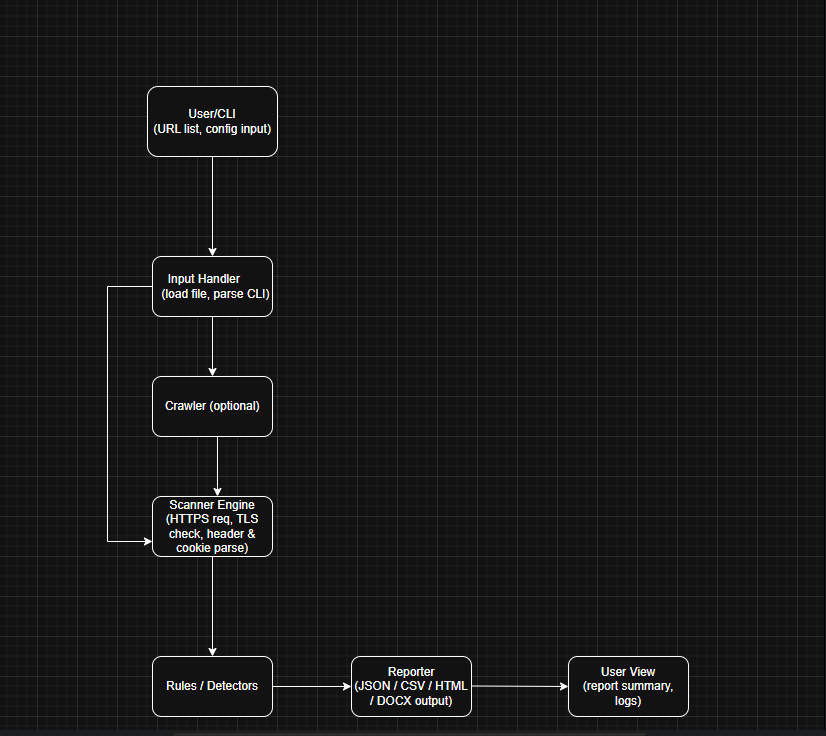
2.6.3 Hỗ trợ giáo dục và nghiên cứu

Ngoài việc hỗ trợ người quản trị hệ thống, công cụ còn là một công cụ hữu ích cho sinh viên và nhà nghiên cứu:

* Giúp hiểu rõ hơn về các tiêu chuẩn bảo mật HTTPS/TLS.
* Cung cấp ví dụ thực tế về các lỗi cấu hình và cách khắc phục.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Kiến trúc tổng quan hệ thống

****

Hình 3.1: Workflow

3.2 Mô tả hoạt động

Hệ thống TLS Scanner hoạt động theo một quy trình gồm bảy (07) bước chính, tương ứng với các thành phần trong kiến trúc tổng quan. Mỗi bước đảm nhận một vai trò độc lập nhưng liên kết chặt chẽ với nhau nhằm đảm bảo quá trình quét, phân tích và đánh giá cấu hình HTTP/TLS được thực hiện chính xác và đầy đủ.

**1. User/CLI**

Người dùng nhập vào:

* URL đơn lẻ hoặc danh sách URL.
* Tệp chứa URL (TXT/CSV).
* Các tham số cấu hình như timeout, số lượng luồng, lựa chọn bật/tắt từng module quét.

Hệ thống tiếp nhận yêu cầu và chuyển sang bước xử lý đầu vào.

**2. Input Handler**

Thành phần này chịu trách nhiệm:

* Đọc và tải danh sách URL từ file hoặc tham số CLI.
* Chuẩn hóa định dạng URL.
* Kiểm tra tính hợp lệ của địa chỉ.
* Loại bỏ URL bị trùng lặp hoặc sai cấu trúc.

Mục đích là đảm bảo dữ liệu vào chính xác trước khi tiến hành quét.

**3. Crawler (tùy chọn)**

Khi được kích hoạt, crawler thực hiện:

* Thu thập các liên kết trong cùng domain (same-site).
* Giới hạn độ sâu thu thập để tránh crawl quá rộng.
* Trả về danh sách endpoint mở rộng để phục vụ quá trình quét.

Crawler giúp công cụ bao quát được nhiều điểm kiểm tra hơn trong cùng website.

**4. Scanner Engine**

Là thành phần lõi của hệ thống, thực hiện các nhiệm vụ chính:

* Gửi yêu cầu HTTPS đến từng URL.
* Thu thập HTTP response headers, cookies và trạng thái máy chủ.
* Thực hiện TLS handshake để lấy thông tin phiên bản TLS, cipher suite, certificate chain.
* Phân tích dữ liệu thô và chuẩn bị cho bước phát hiện lỗi.

Scanner Engine là nơi xử lý trực tiếp toàn bộ quá trình giao tiếp với server.

**5. Rules / Detectors**

Module này đối chiếu dữ liệu thu thập được với bộ quy tắc (rule-set) đã xây dựng, nhằm phát hiện:

* Thiếu hoặc cấu hình sai HSTS.
* Cookie không an toàn (thiếu Secure/HttpOnly/SameSite).
* TLS phiên bản cũ (1.0, 1.1) hoặc cipher yếu (RC4, 3DES).
* Chứng chỉ hết hạn, self-signed hoặc hostname mismatch.
* Thiếu header bảo mật như X-Frame-Options, X-Content-Type-Options, CSP,…

Mỗi phát hiện được phân loại mức độ rủi ro: Critical, High, Medium, Low hoặc Info.

**6. Reporter**

Sau khi phân tích, toàn bộ kết quả được xuất ra các định dạng:

* JSON (chi tiết đầy đủ).
* CSV (tóm tắt).

Reporter đảm nhiệm vai trò tổng hợp và chuẩn hóa báo cáo đầu ra.

**7. User View**

Người dùng được cung cấp giao diện báo cáo và log bao gồm:

* Danh sách URL đã quét.
* Thông tin TLS.
* Header và cookie thu thập được.
* Mức độ rủi ro.
* Gợi ý cải thiện theo từng hạng mục.

Đây là nơi người dùng đánh giá kết quả và đưa ra quyết định xử lý

3.3 Kịch bản thực hiện đề tài

3.3.1 Chuẩn bị môi trường và công cụ

- Mục tiêu: Xây dựng môi trường phát triển ổn định. Chạy công cụ kiểm tra bảo mật HTTP/TLS.

- Thực hiện:

* Cài đặt python, Visual Studio Code và các công cụ hỗ trợ khác.
* Cài đặt các thư viện cần thiết: httpx, aiohttp, cryptography, pytest,...

3.3.2 Thiết kế cấu trúc dự án

- Mục tiêu: Phân chia hệ thống thành các module rõ ràng, thuận tiện cho phát triển và bảo trì.

- Thực hiện:

* Input Handler: nhận danh sách URL/hostname (txt/csv hoặc input trực tiếp).
* Fetcher: gửi request HTTPS (synchronous/async), thu header, cookie, certificate. (requests / aiohttp / ssl)
* TLS Checker: sử dụng SSLyze hoặc pyOpenSSL để kiểm tra phiên bản TLS và cipher suite.
* Detection Engine: áp dụng rule để phát hiện Missing HSTS, Weak TLS, Cookie thiếu Secure/HttpOnly/SameSite.
* Reporter: xuất kết quả sang JSON, CSV.

3.3.3 Triển khai chức năng quét HTTP

- Mục tiêu: Thu thập phản hồi để phân tích bảo mật.

- Thực hiện:

* Sử dụng HTTP client (như httpx) để gửi request song song đến các website.
* Lưu kết quả để phục vụ phân tích ở bước sau.

3.3.4 Phân tích TLS và Cipher Suites

- Mục tiêu: Đảm bảo chứng chỉ HTTPS hợp lệ, còn hạn và khớp với tên miền.

- Thực hiện:

* Thực hiện TLS handshake để lấy thông tin phiên TLS (1.2, 1.3,...) và cipher được chọn.
* Ghi nhận server có hỗ trợ phiên TLS cũ (TLS 1.0, 1.1) hoặc mã hóa yếu (RC4, 3DES) hay không.
* Lưu kết quả để phục vụ phân tích ở bước sau.

3.3.5 Kiểm tra chứng chỉ số

- Mục tiêu: Đánh giá các biện pháp bảo mật trong phần phản hồi HTTP.

- Thực hiện:

* Phân tích các trường trong chứng chỉ: Issuer, Subject, Ngày hết hạn, SAN.
* Phát hiện các vấn đề:
* Chứng chỉ hết hạn hoặc sắp hết hạn.
* Chứng chỉ self-signed.
* Hostname mismatch (tên miền không trùng).

3.3.6 Phân tích Header và Cookie

- Mục tiêu: Đánh giá các biện pháp bảo mật trong phần phản hồi HTTP.

- Thực hiện:

* Kiểm tra header Strict-Transport-Security (HSTS): có tồn tại, max-age, preload,...
* Phân tích Set-Cookie: có gắn cờ Secure, HttpOnly,SameSite hay không.

3.3.7 Đánh giá rủi ro

- Mục tiêu: Phân loại và chấm điểm mức độ nghiêm trọng cho từng lỗ hổng.

- Thực hiện:

Xây dựng bộ quy tắc (rules) cho từng lỗi.

Xác định mức độ:

* Critical: chứng chỉ hết hạn, hỗ trợ TLS 1.0.
* High: thiếu HSTS, cookie thiếu Secure.
* Medium: thiếu HttpOnly, CSP yếu.
* Low/Info: cảnh báo cấu hình không tối ưu.

3.3.8 Xuất báo cáo kết quả

- Mục tiêu: Tổng hợp kết quả quét thành báo cáo phục vụ trình bày và lưu trữ.

- Thực hiện:

* Xuất file JSON (chi tiết) và CSV (tóm tắt).
* Tạo báo cáo HTML bằng Jinja2 có giao diện trực quan.
* Ghi kèm thời gian quét, cấu hình hệ thống và bằng chứng (ảnh log/header).

3.3.9 Đánh giá kết quả

- Mục tiêu: Đảm bảo công cụ hoạt động đúng và phát hiện chính xác.

- Thực hiện: Viết unit test cho từng chức năng (header, cookie, TLS,...).

3.3.10 Trình bày và hoàn thiện đồ án

- Mục tiêu: Hoàn thiện toàn bộ sản phẩm và báo cáo để bảo vệ.

- Thực hiện:

* Chuẩn bị lab demo gồm 2 server: một cấu hình yếu, một chuẩn bảo mật.
* Chạy demo công cụ trực tiếp và trình bày kết quả quét.
* Hoàn thiện báo cáo Word, file PoC (ảnh, video) và mã nguồn.
* Tổng hợp phần đánh giá, hướng phát triển, và kiến nghị cải tiến.

CHƯƠNG 4: TRIỀN KHAI HỆ THỐNG

4.1 Công nghệ và công cụ sử dụng

Trong quá trình xây dựng và triển khai hệ thống, nhóm sử dụng nhiều công nghệ và công cụ khác nhau nhằm đáp ứng các yêu cầu về hiệu năng, tính chính xác và khả năng mở rộng của công cụ quét. Các công nghệ được lựa chọn dựa trên tính ổn định, độ phổ biến và sự hỗ trợ cộng đồng mạnh mẽ.

4.1.1 Ngôn ngữ lập trình

**Python 3.10+**

* Là ngôn ngữ chính để phát triển hệ thống.
* Hỗ trợ xử lý song song, bất đồng bộ (async/await) giúp tăng tốc độ quét.
* Có hệ sinh thái thư viện mạnh để phân tích HTTPS và TLS.

4.1.2 Công cụ hỗ trợ phát triển

**Visual Studio Code**

* Trình soạn thảo chính.
* Hỗ trợ plugin Python, linting, debugging.

**Git & GitHub**

* Quản lý mã nguồn.
* Lưu phiên bản và tài liệu dự án.

4.1.3 Môi trường máy chủ kiểm thử

* Docker Desktop

4.2 Thư viện

Hệ thống sử dụng nhiều thư viện Python, được chia theo từng nhóm chức năng như sau:

4.2.1 Thư viện lõi (Core Dependencies)

**- aiohttp (3.8.4)**

* Thư viện gửi HTTP/HTTPS request theo cơ chế bất đồng bộ (async).
* Tăng tốc độ quét khi phải xử lý nhiều URL cùng lúc.

**- jinja2 (3.1.2)**

* Template engine mạnh mẽ để tạo báo cáo HTML.
* Cho phép trình bày kết quả quét rõ ràng, trực quan.

**- typer (0.10.0)**

* Thư viện tạo giao diện dòng lệnh (CLI).
* Hỗ trợ người dùng chạy lệnh dễ dàng.

4.2.2 Thư viện phân tích TLS/SSL

**- cryptography (>=35.0.0)**

* Phân tích chứng chỉ số (X.509): Subject, Issuer, SAN, ngày hết hạn.
* Kiểm tra certificate chain, chữ ký số.
* Thư viện bảo mật mạnh, được sử dụng rộng rãi trong công nghiệp.

**- pyOpenSSL (23.2.0)**

* Hỗ trợ chạy TLS handshake nâng cao.
* Lấy thông tin phiên TLS và cipher suite server sử dụng.
* Đối chiếu với các chuẩn mã hóa hiện tại để phát hiện cipher yếu.

4.2.3 Thư viện kiểm thử (Testing)

**- pytest (7.4.0)**

* Framework kiểm thử tự động cho Python.
* Dùng test từng module như: phân tích TLS, HTTP header, cookie,…

**- pytest-asyncio (0.21.0)**

* Hỗ trợ viết test cho hàm bất đồng bộ (async).
* Đảm bảo hệ thống chạy ổn định trong môi trường async.

4.2.4 Thư viện xử lý dữ liệu và cấu hình

**- pydantic (>=1.7,<2.0)**

* Dùng khai báo và validate dữ liệu cấu hình.
* Hạn chế lỗi input do sai định dạng URL, rule hoặc tham số quét.

**- pyyaml (6.0)**

* Dùng để đọc file YAML chứa:
  + Bộ quy tắc (rules)
  + Cấu hình hệ thống (config)
* Giúp tách logic rule ra khỏi mã nguồn, dễ bảo trì.

4.2.5 Thư viện phục vụ Web Crawler

**- beautifulsoup4 (4.12.0)**

* Dùng để phân tích HTML, trích xuất link phục vụ crawl shallow.

**- lxml (4.9.0)**

* Parser HTML/XML tốc độ cao, nâng cao hiệu năng của BeautifulSoup.
* Hỗ trợ phân tích trang web với cú pháp phức tạp.

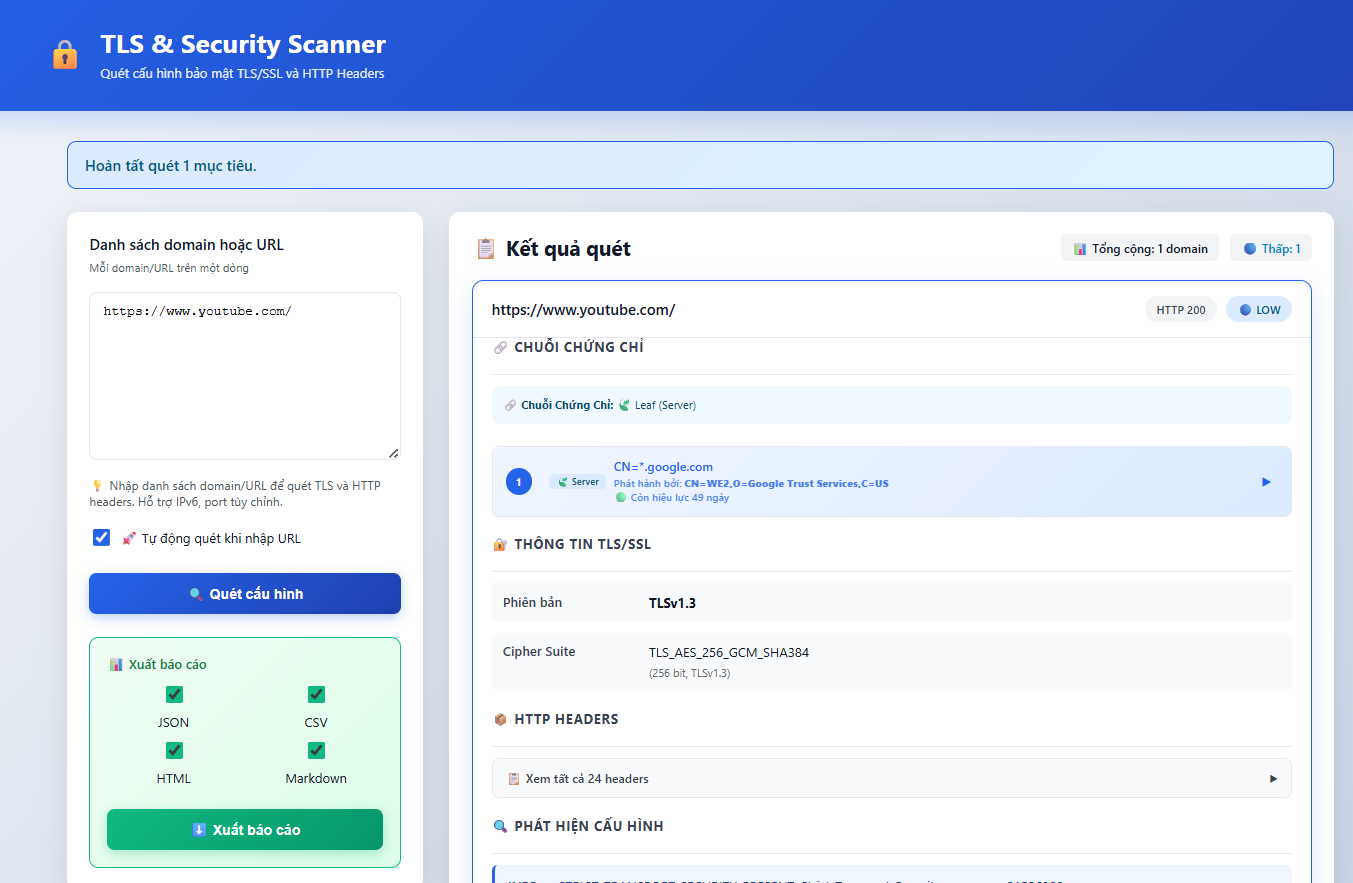
4.2.6 Thư viện tiện ích

**- requests (2.41.0)**

* Gửi request HTTP/HTTPS theo cách đồng bộ.
* Dùng cho kiểm thử thủ công hoặc fallback khi aiohttp lỗi.

4.3 Kết quả chương trình

4.3.1 Quét cấu hình https://www.youtube.com/



Hình 4.1: Phần TLS khi quét url https://www.youtube.com/



Hình 4.2: Phần phát hiện cấu hình khi quét url https://www.youtube.com/

- Người dùng nhập danh sách domain hoặ URL cần kiểm tra. Có tùy chọn xuất báo với nhiều định dạng như JSON, CSV, HTML và Markdown, giúp linh hoạt trong việc lưu trữ và trình bày kết quả.

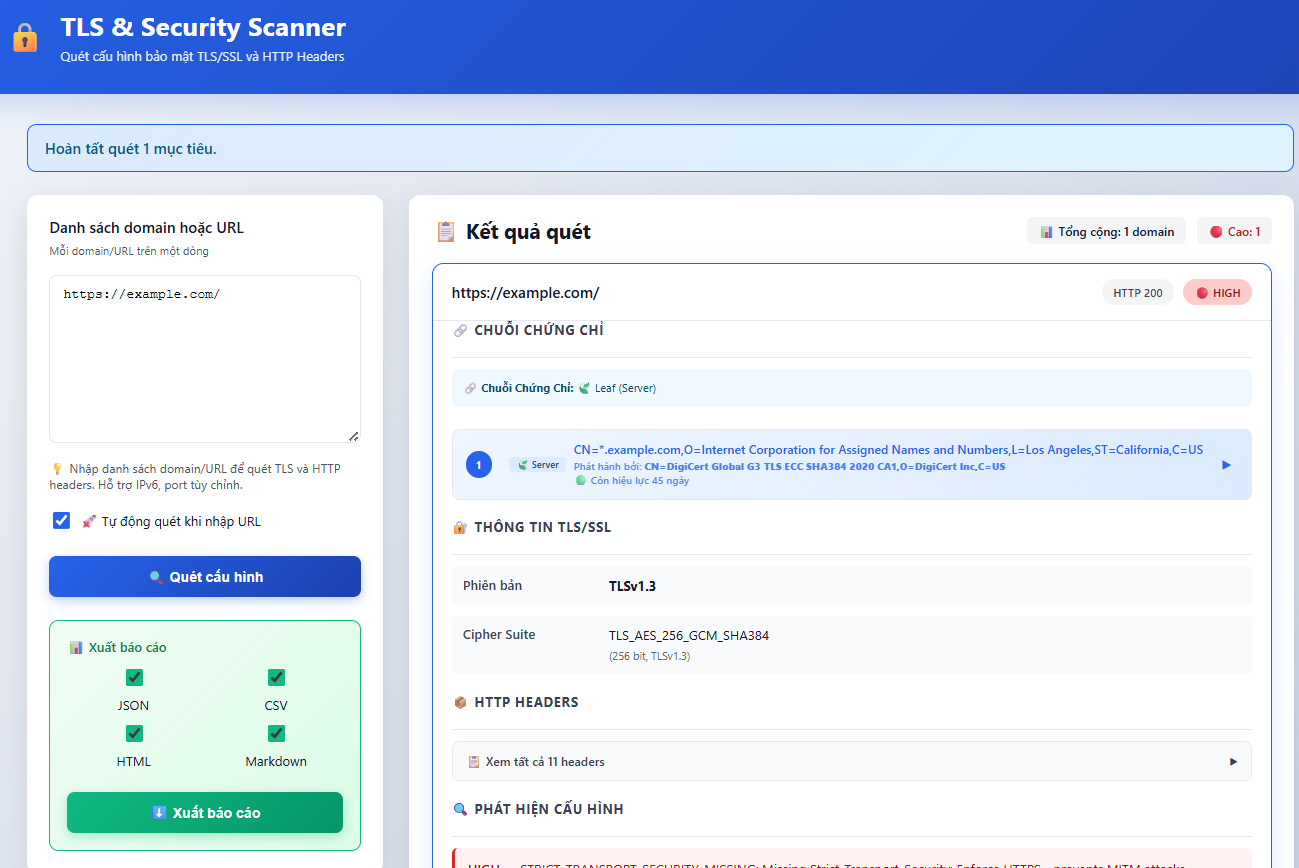
- **Kết quả quét:** Kết quả cho thấy máy chủ dùng:

* TLS phiên bản 1.3.
* Cipher suite: TLS\_AES\_256\_GCM\_SHA384. Đây là những cấu hình mã hóa hiện đại và an toàn.
* Các HTTP Header trả về từ server
* Công cụ có đánh giá và gắn nhãn mức độ.
* Công cụ tự động đưa ra khuyến nghị giúp người quản trị dễ hiểu sai sót và biết cách khắc phục.
* Các cấu hình được triển khai:
  + Strict-Transport-Security:Youtube đã kích hoạt HSTS với thời gian max-age=31536000, đảm bảo trình duyệt luôn sử dụng HTTPS và ngăn tấn công hạ thấp giao thức.
  + X-Content-Type-Options: nosniff: Ngăn trình duyệt tự đoán loại file, giảm nguy cơ XSS dựa trên MIME Sniffing.
  + X-Frame-Options: SAMEORIGIN: ngăn trang web bị nhúng iframe của domain khác. SAMEORIGIN cho phép chính website tự embed mình, nhưng chặn bên ngoài.
  + Content-Security-Policy: Máy chủ có thiết lập CSP giúp hạn chế nguồn script độc hại.
  + X-XSS-Protection: Đã bật bộ lọc ngăn chặn tấn công XSS ở phía trình duyệt.
  + Permissions-Policy: Kiểm soát quyền truy cập của trình duyệt vào các tính năng phần cứng (camera, microphone, v.v…).
  + Cookie có gắn Secure và HttpOnly, đảm bảo:
  + Secure → chỉ gửi qua HTTPS.
  + HttpOnly → JavaScript không thể truy cập cookie, chống XSS.
* Các vấn đề ở mức độ thấp cần cải thiện:
  + Thiếu Referrer-Policy
    - Dữ liệu referrer có thể bị rò rỉ sang bên thứ ba.

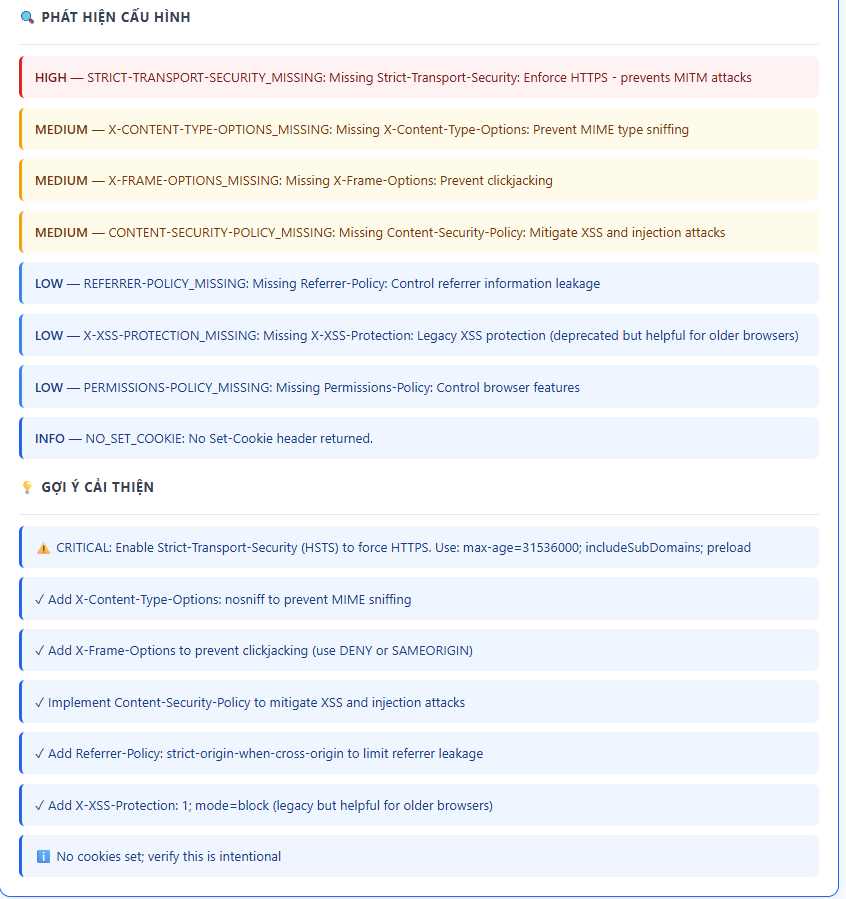
- Khuyến nghị: Referrer-Policy: strict-origin-when-cross-origin

* + Cookie SameSite=None nhưng chấp nhận được vì đi kèm Secure
    - Có thể cân nhắc SameSite=Strict hoặc Lax tùy trường hợp để chống CSRF tốt hơn.

4.3.2 Quét cấu hình https://example.com/



Hình 4.3: Phần TLS và header khi quét url https://example.com/



Hình 4.4: Phần phát hiện và gợi ý khi quét url https://example.com/

- **Kết quả quét:** Kết quả cho thấy máy chủ dùng:

* TLS phiên bản 1.3.
* Cipher suite: TLS\_AES\_256\_GCM\_SHA384. Đây là những cấu hình mã hóa hiện đại và an toàn.
* Nhiều header bảo mật HTTP thiếu: Strict-Transport-Security (HSTS, CRITICAL), X-Content-Type-Options, X-Frame-Options, Content-Security-Policy, cùng các header khác. Ngoài ra response không trả Set-Cookie. Những thiếu sót này làm giảm rào chắn bảo mật phía client.
* Các HTTP Header trả về từ server
* Các cấu hình thiếu:

- Thiếu **Strict-Transport-Security**: Trình duyệt không bị ép buộc sử dụng HSTS dẫn đến có thể bị tấn công hạ cấp giao thức và MITM.

- Thiếu **X-Content-Type-Options**: Trình duyệt có thể MIME sniffing và thực thi nội dung không mong muốn.

- Thiếu **X-Frame-Options**: Dễ bị tấn công Clickjacking qua iframe.

- Thiếu **Content-Security-Policy**: Tăng rủi ro XSS và injection script nguồn không tin cậy.

- Thiếu Referrer-Policy: Tiết lộ referrer và dữ liệu nhạy cảm khi chuyển hướng đến bên thứ ba.

- Thiếu X-XSS-Protection: Không kích hoạt bộ lọc XSS cũ trên các trình duyệt đời thấp.

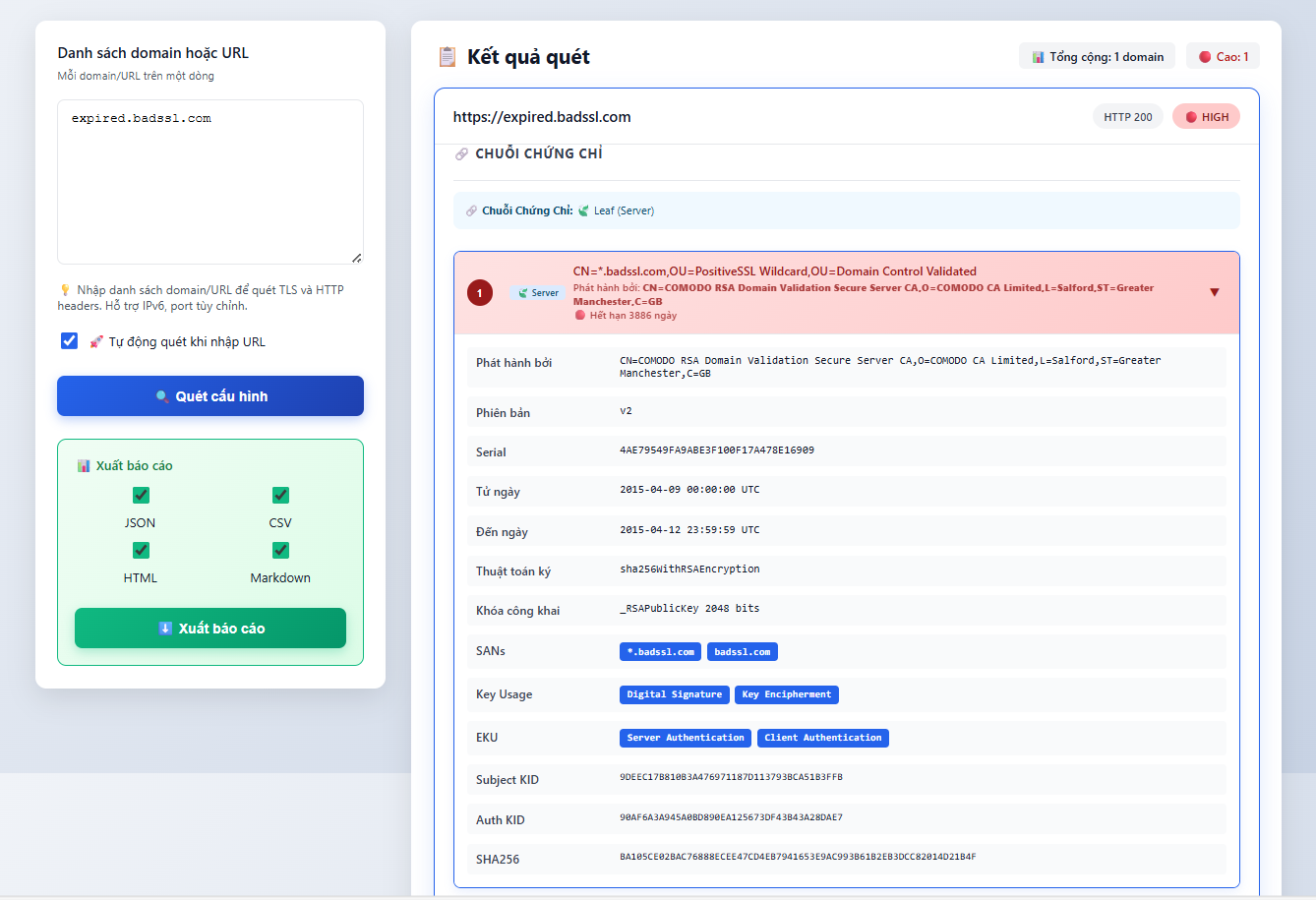
- Thiếu Premissions-Policy: Trình duyệt vẫn có quyền truy cập các API/hardware mà web không kiểm soát.

- Không có Set-Cookie: Không đánh giá được mức độ an toàn của phiên đăng nhập.

* Các gợi ý cải thiện.

4.3.3 Quét các domain để kiểm tra chứng chỉ số

Domain expired.badssl.com: Domain với chứng chỉ đã hết hạn.

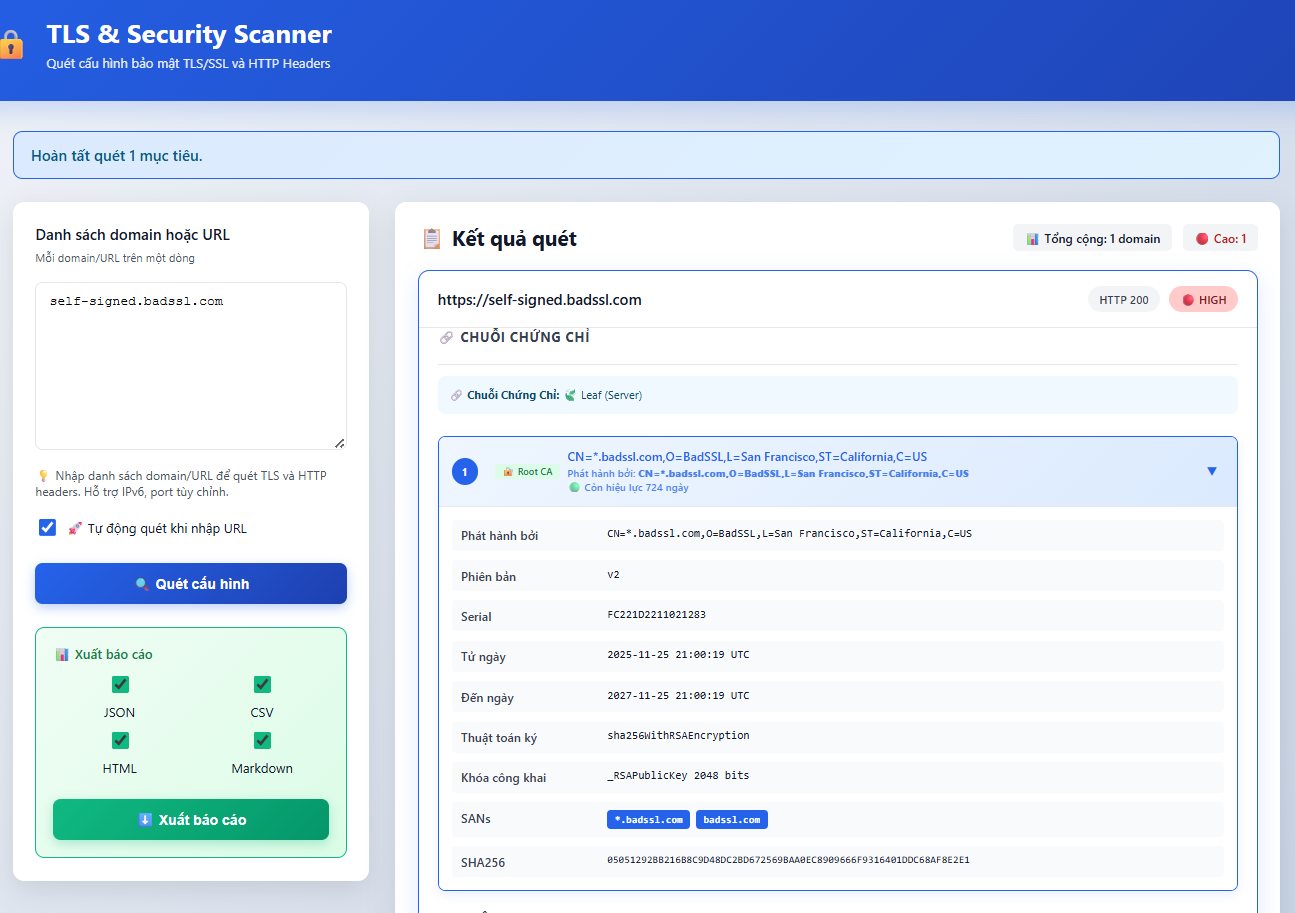


Hình 4.5: Quét chứng chỉ của domain expired.badssl.com

**- Kết quả quét chứng chỉ:**

* Issuer (CA phát hành): COMODO RSA Domain Validation Secure Server CA.
* Valid From: 2015-08-10.
* Valid To: 2016-09-05 (hết hạn).
* Thuật toán ký: sha256WithRSAEncryption.
* RSA Public Key: 2048 bits.
* SANs: \*.badssl.com, badssl.com.

Domain self-signed.badssl.com: Domain dùng chứng chỉ tự-ký.



Hình 4.6: Quét chứng chỉ của domain self-signed.badssl.com

**- Kết quả quét chứng chỉ:**

* Subject: CN=\*.badssl.com, O=BadSSL
* Issuer (phát hành bởi): chính cùng Subject → Self-Signed
* Serial: FC221D22…
* Valid From: 25/11/2025
* Valid To: 25/11/2027
* Thuật toán ký: sha256WithRSAEncryption
* Độ dài khóa: RSA 2048 bits
* SANs: \*.badssl.com, badssl.com

- Dù thời hạn chứng chỉ vẫn còn hiệu lực, nhưng vấn đề cốt lõi là không có CA tin cậy xác thực danh tính máy chủ.

CHƯƠNG 5: ĐÁNH GIÁ VÀ PHÂN TÍCH KẾT QUẢ

5.1 Đánh giá

- Xác định mức độ hiệu quả của công cụ quét HTTP/TLS.

- Đánh giá khả năng phát hiện sai sót cấu hình.

- Phân tích hiệu năng xử lý khi quét nhiều URL.

- Đối chiếu kết quả với tiêu chuẩn bảo mật hiện hành.

- Xác định điểm mạnh và hạn chế thực tế khi triển khai.

5.2 Phân tích kết quả

5.2.1 Kịch bản và môi trường thử nghiệm

Môi trường:

* Python 3.10
* Hệ điều hành: Windows
* Kết nối HTTPS tiêu chuẩn
* Thư viện: aiohttp, cryptography, pyOpenSSL,…

Đối tượng thử nghiệm

* Server chuẩn bảo mật (TLS 1.3, HSTS đầy đủ, cookie an toàn).
* Server cấu hình yếu (TLS cũ, chứng chỉ self-signed, thiếu HSTS).
* Một số website phổ biến trên Internet (host hợp lệ, cho phép truy cập công khai).

Tiêu chí đánh giá

* Độ chính xác phát hiện lỗi.
* Số lượng lỗi phân loại theo mức độ (Critical → Low).
* Thời gian quét.
* Tính ổn định và khả năng xử lý lỗi.

5.2.2 Kết quả thu được

Dựa trên quá trình chạy thử nghiệm công cụ với hai website mẫu là **[https://www.youtube.com/](https://www.youtube.com/" \t "_new)** và **[https://example.com/](https://example.com/" \t "_new)**, hệ thống đã thu thập và phân tích được nhiều thông tin quan trọng liên quan đến cấu hình HTTPS/TLS và HTTP Security Headers. Kết quả thể hiện rõ khả năng phát hiện sai sót, phân loại rủi ro và hỗ trợ đánh giá mức độ an toàn của từng website.

**1. Khả năng thu thập và phân tích TLS/SSL**

* Công cụ đã trích xuất chính xác các thông tin:
* Phiên bản TLS 1.3 đang sử dụng.
* Cipher Suite (TLS\_AES\_256\_GCM\_SHA384) được máy chủ chọn.
* Trạng thái kết nối HTTPS và mã phản hồi HTTP.

**2. Phân tích HTTP Security Headers**

YouTube

* Có đầy đủ các header quan trọng như:
  + Strict-Transport-Security (HSTS)
  + X-Content-Type-Options: nosniff
  + X-Frame-Options: SAMEORIGIN
  + Content-Security-Policy
* Hệ thống còn phát hiện cookie được gắn Secure/HttpOnly, đáp ứng tiêu chuẩn bảo mật hiện đại.

→ Công cụ đánh giá mức rủi ro tổng quan ở mức LOW.

Example.com

* Thiếu hầu hết các header bảo mật quan trọng:
  + Không có HSTS (mức nghiêm trọng Critical)
  + Không có X-Frame-Options
  + Không có X-Content-Type-Options
  + Không có CSP
  + Không trả về bất kỳ Set-Cookie nào để đánh giá tính an toàn.

→ Công cụ đánh giá tổng rủi ro ở mức HIGH, cho thấy cấu hình kém an toàn cho môi trường thực tế.

**3. Phân loại mức độ rủi ro**

Công cụ đã tự động gắn nhãn mức độ rủi ro dựa trên rule-set nội bộ:

* Low với hệ thống có cấu hình chuẩn (ví dụ YouTube).
* High/Critical đối với hệ thống thiếu cấu hình bảo mật cơ bản (ví dụ Example.com).

**4. Trình bày kết quả trực quan**

Giao diện hiển thị đã chứng minh được:

* Kết quả được chia rõ thành các nhóm (TLS/SSL, HTTP Headers, Risk Level).
* Có hỗ trợ xuất báo cáo đa định dạng: JSON, CSV, HTML, Markdown.
* Hỗ trợ quét nhiều URL cùng lúc và tổng hợp số lượng mục tiêu trên giao diện.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Trong bối cảnh các cuộc tấn công mạng ngày càng tinh vi và phức tạp, việc đảm bảo cấu hình HTTPS/TLS đúng chuẩn là một yêu cầu cấp thiết đối với các hệ thống web. Đề tài "Thiết kế và triển khai công cụ quét tự động cho các sai sót cấu hình mã hóa trên ứng dụng web" đã giải quyết được bài toán phát hiện và phân tích các lỗi phổ biến trong cấu hình HTTPS/TLS, từ đó hỗ trợ người quản trị hệ thống cải thiện mức độ bảo mật của website.

Công cụ được xây dựng trong đề tài đã đáp ứng được các mục tiêu đề ra, bao gồm:

* Phát hiện các lỗi cấu hình phổ biến như thiếu HSTS, sử dụng TLS lỗi thời, cipher suite yếu, cookie không an toàn và chứng chỉ số không hợp lệ.
* Cung cấp giao diện thân thiện, hỗ trợ cả dòng lệnh (CLI) và giao diện web (UI), giúp người dùng dễ dàng sử dụng.
* Xuất báo cáo chi tiết dưới nhiều định dạng (CSV, JSON, HTML), kèm theo các khuyến nghị cụ thể để cải thiện cấu hình bảo mật.

Kết quả thử nghiệm cho thấy công cụ hoạt động ổn định, phát hiện chính xác các sai sót cấu hình trên các môi trường thử nghiệm khác nhau. Điều này chứng minh rằng công cụ không chỉ có tính ứng dụng cao trong thực tế mà còn là một tài liệu tham khảo hữu ích cho sinh viên và nhà nghiên cứu trong lĩnh vực bảo mật web.

Tuy nhiên, do giới hạn về thời gian và phạm vi nghiên cứu, công cụ vẫn còn một số hạn chế nhất định, chẳng hạn như chưa hỗ trợ kiểm tra toàn diện các lỗ hổng bảo mật phức tạp hoặc chưa tối ưu hóa hiệu suất khi quét số lượng lớn URL. Những hạn chế này sẽ là tiền đề để tiếp tục phát triển và hoàn thiện công cụ trong tương lai.

6.2 Hướng phát triển

Để nâng cao hiệu quả và tính ứng dụng của công cụ, trong tương lai, các hướng phát triển sau đây có thể được thực hiện:

1. **Mở rộng phạm vi kiểm tra**

Hiện tại, công cụ tập trung vào các sai sót phổ biến trong cấu hình HTTPS/TLS và HTTP headers. Trong tương lai, công cụ có thể được mở rộng để kiểm tra thêm các lỗ hổng bảo mật khác, chẳng hạn như:

* Phát hiện các lỗ hổng XSS, CSRF, SQL Injection.
* Kiểm tra các API endpoint và ứng dụng web phức tạp.
* Phân tích các giao thức khác như HTTP/3 hoặc QUIC.

1. **Tích hợp thêm các công cụ phân tích bảo mật**

Để tăng cường khả năng phát hiện và phân tích, công cụ có thể được tích hợp với các công cụ bảo mật phổ biến khác như OWASP ZAP, Burp Suite, hoặc Nikto. Việc tích hợp này sẽ giúp công cụ trở thành một giải pháp toàn diện hơn trong việc kiểm tra và đánh giá bảo mật web.

1. **Cải thiện hiệu suất và khả năng mở rộng**

Khi số lượng URL cần quét tăng lên, hiệu suất của công cụ có thể bị ảnh hưởng. Do đó, cần tối ưu hóa thuật toán và sử dụng các công nghệ như đa luồng (multithreading) hoặc xử lý song song (parallel processing) để tăng tốc độ quét. Ngoài ra, công cụ có thể được triển khai trên nền tảng đám mây để hỗ trợ quét đồng thời trên quy mô lớn.

1. **Phát triển giao diện người dùng nâng cao**

Giao diện web hiện tại của công cụ đã đáp ứng được các yêu cầu cơ bản, nhưng trong tương lai, có thể bổ sung thêm các tính năng nâng cao như:

* Biểu đồ trực quan hóa kết quả quét.
* Dashboard quản lý lịch sử quét và theo dõi tiến độ.
* Tùy chỉnh cấu hình quét theo nhu cầu của người dùng.

1. **Hỗ trợ kiểm tra tự động định kỳ**

Một tính năng hữu ích khác là cho phép người dùng thiết lập lịch trình quét tự động. Công cụ có thể được cấu hình để kiểm tra định kỳ các website và gửi báo cáo qua email hoặc thông báo qua các nền tảng như Slack, Microsoft Teams.

1. **Cải thiện khả năng tích hợp DevSecOps**

Để phù hợp với các quy trình phát triển phần mềm hiện đại, công cụ có thể được tích hợp vào các pipeline CI/CD (Continuous Integration/Continuous Deployment). Điều này giúp phát hiện và khắc phục các sai sót cấu hình ngay từ giai đoạn phát triển, giảm thiểu rủi ro bảo mật khi triển khai ứng dụng.

TÀI LIỆU THAM KHẢO

[1] <https://fptshop.com.vn/tin-tuc/danh-gia/http-la-gi-166491>

[2] <https://www.cloudflare.com/learning/ssl/what-is-ssl/>

[3] <https://owasp.org/www-project-secure-headers/#http-strict-transport-security>

[4] <https://chatgpt.com/c/692fc52b-1938-8321-bf3b-e476a6c4d544>