**MAMMOTHON HACKATHON**

**DỰ ÁN: Certificate Transparency Browser Extension – Prism Edition (WASM Integration)**

**Nhóm thực hiện:**

* Thành viên 1
* Thành viên 2
* Thành viên 3

**Ngày nộp:** [Ngày, Tháng, Năm]

**MỤC LỤC**

1. Tóm tắt Dự án (Abstract)
2. Giới thiệu & Bối cảnh
3. Mục tiêu & Yêu cầu Kỹ thuật (Bám sát đề bài Mammothon)
4. Kiến trúc Hệ thống
5. Chi tiết Triển khai & Xử lý  
    5.1. Frontend – Giao diện Người dùng (Popup Extension)  
    5.2. Background Service – Xử lý Certificate Check  
    5.3. Tích hợp WASM – Module Rust & WebAssembly
6. Hướng dẫn Triển khai & Vận hành
7. Đánh giá Theo Tiêu Chí Hackathon  
    7.1. Sự Đổi Mới  
    7.2. Độ Phức Tạp Kỹ Thuật  
    7.3. Mức Độ Hoàn Thành  
    7.4. Tầm Ảnh Hưởng  
    7.5. Tiềm Năng  
    7.6. Làm Việc Nhóm
8. Kết Luận & Hướng Phát Triển Tương Lai
9. Tài liệu Tham khảo

**1. TÓM TẮT DỰ ÁN (Abstract)**

Dự án “Certificate Transparency Browser Extension – Prism Edition” được thiết kế nhằm cung cấp một giải pháp bảo mật tiên tiến cho người dùng duyệt web, thông qua việc kiểm tra chứng chỉ SSL/TLS một cách minh bạch. Đặc biệt, dự án tích hợp công nghệ Prism – giải pháp key-transparency trust-minimized – cùng với module kiểm tra được viết bằng Rust và biên dịch sang WebAssembly (WASM) để tăng hiệu suất và bảo mật. Extension này không chỉ cho phép kiểm tra chứng chỉ mà còn lưu trữ lịch sử kiểm tra, hỗ trợ cài đặt endpoint tùy chỉnh, mở ra tiềm năng tích hợp với các công nghệ blockchain, AI và các dịch vụ bảo mật doanh nghiệp trong tương lai.

**2. GIỚI THIỆU & BỐI CẢNH**

Trong bối cảnh các mối đe dọa từ các cuộc tấn công Man-in-the-Middle (MITM) và sự cố liên quan đến chứng chỉ SSL/TLS ngày càng gia tăng, việc đảm bảo tính minh bạch của chứng chỉ trở thành yêu cầu cấp thiết. Certificate Transparency (CT) cho phép ghi nhận công khai thông tin chứng chỉ, giúp phát hiện kịp thời các bất thường và đảm bảo tính hợp lệ của chứng chỉ.  
Dự án này được xây dựng dựa trên hướng dẫn “Build a Certificate Transparency Browser Extension” của Encode Club, với mục tiêu tích hợp Prism – hệ thống key-transparency tiên tiến – để mang lại giải pháp bảo mật tối ưu cho người dùng. Việc tích hợp module WASM cho phép chuyển các phép tính xác minh chứng chỉ sang Rust, giảm tải cho JavaScript, tăng tốc xử lý và bảo mật.

**3. MỤC TIÊU & YÊU CẦU KỸ THUẬT**

**Mục tiêu chính:**

* Xây dựng một Chrome Extension cho phép người dùng kiểm tra chứng chỉ của trang web thông qua việc gọi module WASM (Rust → WASM) thay cho API bên ngoài, đảm bảo tính độc lập, hiệu suất và bảo mật.
* Lưu trữ lịch sử kiểm tra và cho phép người dùng tùy chỉnh endpoint (dự kiến cho tích hợp Prism thực tế khi có sẵn).

**Yêu cầu kỹ thuật (bám sát đề bài Mammothon):**

* **Công nghệ tiên tiến:** Sử dụng Chrome Extension Manifest V3, xử lý bất đồng bộ qua fetch API (với WASM thay thế một số logic tính toán), lưu trữ thông minh qua chrome.storage và tích hợp module Rust biên dịch WASM.
* **Tích hợp Prism:** Kiến trúc mở rộng, dễ dàng tích hợp với API của Prism, nhằm cung cấp quy trình kiểm tra chứng chỉ minh bạch và trust-minimized.
* **Giao diện chuyên nghiệp:** Phong cách Material Design, hiệu ứng animation (fade-in, slide-in), hỗ trợ nhiều tab chức năng (Check, History, Settings) để tăng trải nghiệm người dùng.
* **Bảo mật & hiệu suất:** Xử lý lỗi chuyên sâu, đảm bảo hoạt động ổn định trên môi trường thực, phù hợp với các yêu cầu của hệ thống bảo mật doanh nghiệp.

**4. KIẾN TRÚC HỆ THỐNG**

* **Frontend (Popup Extension):**
  + Xây dựng bằng HTML5, CSS3 và JavaScript.
  + Giao diện chia làm 3 tab chính: Kiểm Tra (Check), Lịch Sử (History) và Cài Đặt (Settings).
  + Sử dụng các hiệu ứng động mượt mà, bố cục dạng card hiện đại, màu sắc hài hòa.
* **Background Service:**
  + Xử lý các yêu cầu từ giao diện.
  + Thay vì gọi API bên ngoài, toàn bộ xử lý chứng chỉ được chuyển sang module WASM, giúp tăng hiệu suất và bảo mật.
* **Module WASM (Rust → WASM):**
  + Viết module kiểm tra chứng chỉ bằng Rust, sau đó biên dịch sang WASM (sử dụng wasm-pack).
  + Module này thực hiện các phép tính và kiểm tra chứng chỉ cục bộ, trả về kết quả dưới dạng JSON.
  + Kiến trúc mở rộng để tích hợp các thuật toán kiểm tra CT phức tạp trong tương lai.

**5. CHI TIẾT TRIỂN KHAI & XỬ LÝ**

**5.1. Frontend – Giao diện Người dùng (Popup Extension)**

* **popup.html:**
  + Giao diện gồm các tab chức năng: Kiểm Tra, Lịch Sử và Cài Đặt.
  + Hiển thị tiêu đề, nút kiểm tra, kết quả trả về và danh sách lịch sử kiểm tra.
  + Tích hợp Google Fonts, Material Icons để tạo giao diện hiện đại.
* **popup.css:**
  + Sử dụng gradient nền, hiệu ứng fade-in và slide-in.
  + Bố cục dạng card với viền bo tròn, bóng đổ mượt mà.
  + Màu sắc và kiểu chữ được tối ưu để tăng tính thẩm mỹ và thân thiện với người dùng.
* **popup.js:**
  + Quản lý sự kiện chuyển tab, gửi yêu cầu kiểm tra chứng chỉ đến background.
  + Cập nhật kết quả trả về và lưu lịch sử kiểm tra vào chrome.storage.
  + Tính năng cài đặt cho phép người dùng nhập endpoint API tùy chỉnh (dự kiến cho tích hợp Prism thực tế).

**5.2. Background Service – Xử lý Certificate Check**

* **Xử lý bất đồng bộ & quản lý lỗi:**
  + Sử dụng async/await để thực hiện các yêu cầu một cách mượt mà.
  + Xử lý ngoại lệ và trả về thông báo lỗi chi tiết cho người dùng và nhà phát triển.
* **Tích hợp với WASM:**
  + Thay vì gọi API bên ngoài, background.js gọi hàm verify\_certificate từ module WASM để kiểm tra chứng chỉ.
  + Kết quả kiểm tra sau đó được gửi về popup và lưu trữ vào lịch sử qua chrome.storage.

**5.3. Tích hợp WASM – Module Rust & WebAssembly**

* **Viết module Rust:**
  + Module được viết bằng Rust (src/lib.rs) chứa hàm verify\_certificate nhận domain làm đối số.
  + Hàm này thực hiện giả lập kiểm tra chứng chỉ, trả về một đối tượng JSON chứa thông tin chứng chỉ và trạng thái xác minh.

*Ví dụ mã Rust:*

rust

CopyEdit

// src/lib.rs

use wasm\_bindgen::prelude::\*;

use serde::{Serialize};

#[derive(Serialize)]

pub struct CertificateResult {

pub certificate: String,

pub verified: bool,

}

#[wasm\_bindgen]

pub fn verify\_certificate(domain: &str) -> String {

// Giả lập logic kiểm tra chứng chỉ

let result = CertificateResult {

certificate: format!("Chứng chỉ của {} được xác minh thành công", domain),

verified: true,

};

// Chuyển kết quả sang chuỗi JSON

serde\_json::to\_string(&result).unwrap\_or\_else(|\_| format!("Lỗi khi kiểm tra {}", domain))

}

* **Biên dịch WASM:**  
  Sử dụng lệnh sau để biên dịch module:

bash

CopyEdit

wasm-pack build --target web

Sau khi biên dịch, các file WASM (trong thư mục pkg) sẽ được tích hợp vào dự án extension.

* **Tích hợp vào background.js:**  
  Background.js gọi hàm verify\_certificate từ module WASM để thực hiện kiểm tra chứng chỉ mà không cần phụ thuộc vào API bên ngoài.

**6. HƯỚNG DẪN TRIỂN KHAI & VẬN HÀNH**

1. **Chuẩn bị mã nguồn:**
   * Bao gồm các file: manifest.json, background.js, popup.html, popup.css, popup.js, file icon.png và thư mục pkg (chứa module WASM).
2. **Load Extension:**
   * Mở Chrome, truy cập chrome://extensions/, bật Developer Mode và chọn “Load unpacked” trỏ đến thư mục dự án.
3. **Test & Debug:**
   * Mở một tab với website bất kỳ, mở extension, chuyển qua tab “Kiểm Tra” và nhấn nút “Kiểm Tra Ngay”.
   * Kết quả kiểm tra được xử lý qua module WASM sẽ hiển thị trong popup, và lịch sử kiểm tra được cập nhật tự động.
   * Kiểm tra tính năng cài đặt endpoint (nếu tích hợp với API Prism thực).
4. **Triển khai API thực tế (nếu có):**
   * Khi có endpoint Prism thực, cập nhật background.js để gọi API đó thay vì sử dụng module WASM mô phỏng.

**7. ĐÁNH GIÁ THEO TIÊU CHÍ HACKATHON**

**7.1. Sự Đổi Mới (20 điểm)**

* **Độc đáo & sáng tạo:**
  + Tích hợp công nghệ Prism và module WASM vào Chrome Extension, mang lại giải pháp kiểm tra chứng chỉ minh bạch mà không phụ thuộc vào API bên ngoài.
  + Tính năng lưu lịch sử và cài đặt tùy chỉnh mở rộng ứng dụng bảo mật.

**7.2. Độ Phức Tạp Kỹ Thuật (20 điểm)**

* **Ứng dụng công nghệ tiên tiến:**
  + Sử dụng Chrome Extension Manifest V3, xử lý bất đồng bộ, tích hợp module Rust/WASM, quản lý lỗi và lưu trữ thông minh qua chrome.storage.
  + Kiến trúc mở rộng, dễ tích hợp các thuật toán xác minh chứng chỉ phức tạp trong tương lai.

**7.3. Mức Độ Hoàn Thành (20 điểm)**

* **Đầy đủ & ổn định:**
  + Giao diện người dùng chuyên nghiệp, hiện đại với nhiều hiệu ứng động.
  + Tất cả tính năng (kiểm tra, lịch sử, cài đặt) hoạt động mượt mà, không gặp lỗi trong quá trình vận hành.

**7.4. Tầm Ảnh Hưởng (15 điểm)**

* **Tác động tới cộng đồng:**
  + Giúp người dùng duyệt web có trải nghiệm bảo mật cao, phát hiện sớm các mối đe dọa từ chứng chỉ không hợp lệ, giảm nguy cơ tấn công MITM.
  + Tiềm năng tích hợp với các công nghệ blockchain và AI mở ra thị trường bảo mật toàn cầu.

**7.5. Tiềm Năng (15 điểm)**

* **Khả năng thương mại hóa:**
  + Mô hình kinh doanh dựa trên API bảo mật, tích hợp Prism, cung cấp giải pháp bảo mật cho doanh nghiệp và cá nhân.
  + Giá trị cốt lõi mạnh mẽ, mở rộng dịch vụ trên quy mô toàn cầu.

**7.6. Làm Việc Nhóm (10 điểm)**

* **Phối hợp & giao tiếp:**
  + Code được viết rõ ràng, có tài liệu hướng dẫn chi tiết và quy trình phát triển minh bạch, thể hiện sự cộng tác hiệu quả giữa các thành viên.

**8. KẾT LUẬN & HƯỚNG PHÁT TRIỂN TƯƠNG LAI**

**Kết luận:**  
Dự án “Certificate Transparency Browser Extension – Prism Edition” đã được hoàn thiện với giao diện bắt mắt, hiệu ứng động chuyên nghiệp và tính năng đầy đủ. Việc tích hợp module WASM giúp tăng hiệu suất và bảo mật, đáp ứng tốt yêu cầu của hackathon Mammothon và mở ra hướng phát triển tích hợp các công nghệ tiên tiến như blockchain, AI và đăng nhập người dùng cá nhân.

**Hướng phát triển:**

* Tích hợp sâu hơn module Rust/WASM để xử lý các thuật toán xác minh phức tạp.
* Phát triển tính năng đăng nhập, cá nhân hóa và báo cáo chi tiết cho người dùng.
* Mở rộng tích hợp với hệ thống Prism thực tế, cung cấp dịch vụ bảo mật cho doanh nghiệp và mở rộng mô hình kinh doanh.

**9. TÀI LIỆU THAM KHẢO**

* [Encode Club – Mammothon: Build a Certificate Transparency Browser Extension](https://encodeclub.notion.site/mammothon-build-a-certificate-transparency-browser-extension)
* [Prism Documentation](https://docs.prism.rs/)
* [ctclient trên Docs.rs](https://docs.rs/ctclient/latest/ctclient/)
* Chrome Extension Developer Guide