

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

*Đề tài: Xây dựng ứng dụng bảo vệ ảnh dùng Digital
Watermarking và phát hiện tấn công*

Người hướng dẫn : TS. NGUYỄN HỒNG SƠN
Sinh viên thực hiện : NGUYỄN NGỌC HÙNG
NGÔ HUỲNH VĨNH PHÚ
Lớp : D20CQAT01-N
Khoá : 2020-2025
Ngành : AN TOÀN THÔNG TIN
Hệ : ĐẠI HỌC CHÍNH QUY

TP.HCM, tháng 12/2024

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

*Đề tài: Xây dựng ứng dụng bảo vệ ảnh dùng Digital
Watermarking và phát hiện tấn công*

Người hướng dẫn : TS. NGUYỄN HỒNG SƠN
Sinh viên thực hiện : NGUYỄN NGỌC HÙNG
NGÔ HUỲNH VĨNH PHÚ
Lớp : D20CQAT01-N
Khoá : 2020-2025
Ngành : AN TOÀN THÔNG TIN
Hệ : ĐẠI HỌC CHÍNH QUY

TP.HCM, tháng 12/2024

MỤC LỤC

| | |
|--|-----------|
| MỤC LỤC | i |
| DANH MỤC CÁC HÌNH | iv |
| MỞ ĐẦU..... | 1 |
| 1. Lý do chọn đề tài..... | 1 |
| 2. Mục đích của đề tài..... | 1 |
| 3. Phạm vi nghiên cứu | 1 |
| 4. Phương pháp nghiên cứu | 1 |
| 5. Kết cấu của đề tài | 2 |
| PHẦN I: BẢO VỆ ẢNH DÙNG DIGITAL WATERMARKING..... | 3 |
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT..... | 3 |
| 1.1 Khái niệm và vai trò của Digital Watermarking | 3 |
| 1.1.1. Digital Watermarking là gì?..... | 3 |
| 1.1.2. Vai trò của Digital Watermarking. | 3 |
| 1.2 Các kỹ thuật Digital Watermarking phổ biến..... | 4 |
| 1.3 Phân tích các kỹ thuật và thuật toán Digital Watermarking phổ biến..... | 6 |
| 1.3.1. Scale-Invariant Feature Transform (SIFT). | 6 |
| 1.3.2. Discrete Wavelet Transform (DWT)..... | 6 |
| 1.3.3. Discrete Cosine Transform (DCT)..... | 7 |
| 1.3.4. Singular Value Decomposition (SVD)..... | 7 |
| 1.3.5. Least Significant Bit (LSB). | 8 |
| 1.4 Tiêu chí đánh giá Watermark..... | 9 |
| CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ..... | 10 |
| 2.1 Phân tích yêu cầu hệ thống | 10 |
| 2.2 Thiết kế thuật toán nhúng và rút trích watermark..... | 10 |
| 2.3 Thiết kế kiến trúc hệ thống..... | 11 |
| 2.3.1. Giao diện người dùng. | 11 |
| 2.3.2. Hệ thống xử lý phía server..... | 16 |
| CHƯƠNG 3: TRIỂN KHAI | 17 |
| 3.1 Cài đặt hệ thống..... | 17 |
| 3.1.1. Cấu hình môi trường phát triển..... | 17 |
| 3.1.2. Tích hợp thuật toán với ứng dụng web..... | 17 |

| | |
|---|-----------|
| 3.2 Thử nghiệm và đánh giá. | 17 |
| 3.2.1. Đánh giá chất lượng ảnh sau khi nhúng watermark. | 17 |
| 3.2.2. Kiểm tra tính bền vững của watermark. | 18 |
| KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... | 20 |
| 4.1 Kết luận | 20 |
| 4.1.1. Tổng kết nội dung nghiên cứu: | 20 |
| 4.1.2. Kết quả đạt được:..... | 20 |
| 4.1.3. Đóng góp của nghiên cứu: | 20 |
| 4.2 Hướng phát triển | 20 |
| 4.2.1. Cải tiến thuật toán và hệ thống | 20 |
| 4.2.2. Mở rộng khả năng chống tấn công | 20 |
| 4.2.3. Ứng dụng thực tế: | 21 |
| PHẦN II: PHÁT HIỆN TẤN CÔNG..... | 22 |
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT..... | 22 |
| 5.1 Tổng quan về an ninh mạng | 22 |
| 5.2 Một số loại tấn công mạng phổ biến | 22 |
| 5.2.1. Tấn công từ chối dịch vụ (DoS/DDoS): | 22 |
| 5.2.2. Tấn công man-in-the-middle (MITM):..... | 22 |
| 5.2.3. Tấn công SQL Injection..... | 22 |
| 5.2.4. Tấn công thực thi mã từ xa (RCE)..... | 23 |
| 5.2.5. Các loại tấn công khác | 23 |
| 5.3 Các lỗ hổng bảo mật nổi bật | 23 |
| 5.3.1. CVE-2020-0796 (SMBGhost) | 23 |
| 5.3.2. CVE-2020-24932..... | 23 |
| 5.4 Phát hiện và phòng chống tấn công | 24 |
| 5.4.1. Phát hiện tấn công..... | 24 |
| 5.4.2. Phòng chống tấn công..... | 24 |
| 5.5 Các công cụ và kỹ thuật phát hiện tấn công | 24 |
| CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ..... | 26 |
| 5.1 Phân tích yêu cầu..... | 26 |
| 6.1.1. Yêu cầu chức năng..... | 26 |
| 6.1.2. Yêu cầu phi chức năng..... | 26 |

| | | |
|--|---|-----------|
| 6.2 | Mô hình hệ thống..... | 26 |
| 6.2.1. | Kiến trúc hệ thống..... | 26 |
| 6.2.2. | Lưu đồ hoạt động của hệ thống | 26 |
| 6.3 | Phân tích lỗ hổng | 27 |
| 6.3.1. | CVE-2020-0796..... | 27 |
| 6.3.2. | CVE-2020-24932..... | 28 |
| 6.4 | Thiết kế mô hình học máy..... | 29 |
| 6.4.1. | Dữ liệu huấn luyện..... | 29 |
| 6.4.2. | Trích xuất đặc trưng..... | 29 |
| 6.4.3. | Lựa chọn mô hình..... | 29 |
| 6.5 | Công nghệ và công cụ sử dụng | 29 |
| CHƯƠNG 3: TRIỂN KHAI | | 31 |
| 7.1 | Huấn luyện mô hình học máy | 31 |
| 7.2 | CVE-2020-0796 | 34 |
| 7.2.1. | Cài đặt môi trường | 34 |
| 7.2.2. | Mô phỏng tấn công | 35 |
| 7.2.3. | Thu thập và phân tích | 36 |
| 7.3 | CVE-2020-24932 | 36 |
| 7.3.1. | Cài đặt môi trường | 36 |
| 7.3.2. | Mô phỏng tấn công | 37 |
| 7.3.3. | Thu thập và phân tích | 39 |
| 7.4 | So sánh và đánh giá | 40 |
| KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... | | 41 |
| 8.1 | Kết luận | 41 |
| 8.1.1. | Tổng kết nội dung nghiên cứu: | 41 |
| 8.1.2. | Kết quả đạt được | 41 |
| 8.1.3. | Đóng góp của nghiên cứu | 41 |
| 8.2 | Hướng phát triển | 41 |
| TÀI LIỆU THAM KHẢO..... | | 43 |

DANH MỤC CÁC HÌNH

| | |
|--|----|
| Hình 1: Hình ảnh sau khi áp dụng biến đổi SIFT..... | 6 |
| Hình 2: Biến đổi DWT mức 1 | 7 |
| Hình 3: Thuật toán LSB..... | 8 |
| Hình 4: Giao diện upload ảnh..... | 12 |
| Hình 5: Giao diện trích xuất watermark | 12 |
| Hình 6: Trang chủ | 12 |
| Hình 7: Trang đăng nhập | 13 |
| Hình 8: Trang đăng ký | 13 |
| Hình 9: Trang xác thực email | 14 |
| Hình 10: Trang chi tiết ảnh và thanh toán | 14 |
| Hình 11: Trang kết quả key của ảnh | 15 |
| Hình 12: Trang bộ sưu tập | 15 |
| Hình 13: Trang hồ sơ cá nhân..... | 16 |
| Hình 14: Logo..... | 18 |
| Hình 15: Hình sau khi thêm watermark ả..... | 18 |
| Hình 16: Hình ảnh trước khi thêm watermark | 18 |
| Hình 17: Google Colab..... | 31 |
| Hình 18: Import thư viện chương trình phát hiện tấn công..... | 32 |
| Hình 19: Khởi tạo Flask app và tải mô hình | 33 |
| Hình 20: Khởi tạo danh sách lưu log và cảnh báo..... | 33 |
| Hình 21: Danh sách các cột đặc trưng chính | 33 |
| Hình 22: Hàm mã hóa cột dạng chuỗi | 33 |
| Hình 23: Hàm xử lý gói tin..... | 33 |
| Hình 24: Hàm bắt gói tin và dự đoán tấn công..... | 34 |
| Hình 25: Thông báo cảnh báo và log tấn công | 34 |
| Hình 26: Phiên bản máy nạn nhân..... | 35 |
| Hình 27: Cảnh báo tấn công CVE-20202-0796 | 36 |
| Hình 28: Cài đặt XAMPP | 37 |
| Hình 29: Trang web Sourcecodester Complaint Management System 1.0..... | 37 |
| Hình 30: Tấn công thành công | 38 |
| Hình 31: Cảnh báo tấn công CVE-2020-24932 | 39 |
| Hình 32: Kết quả train mô hình..... | 40 |

MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại công nghệ số hiện nay, vấn đề bảo mật thông tin và bảo vệ bản quyền nội dung số ngày càng trở nên cấp thiết. Các cuộc tấn công mạng, đặc biệt là khai thác lỗ hổng bảo mật, không chỉ gây tổn hại về tài chính mà còn ảnh hưởng nghiêm trọng đến uy tín và an toàn dữ liệu của cá nhân, tổ chức. Song song đó, việc bảo vệ quyền sở hữu trí tuệ, đặc biệt là bản quyền hình ảnh, đang đối mặt với nhiều thách thức từ các hành vi sao chép và sử dụng trái phép.

Hai đề tài được chọn phản ánh những vấn đề nóng hổi này:

- Đề tài 1: Xây dựng ứng dụng bảo vệ ảnh bằng Digital Watermarking nhằm đảm bảo bản quyền hình ảnh thông qua công nghệ nhúng thông tin bảo mật vào ảnh.
- Đề tài 2: Xây dựng ứng dụng phát hiện tấn công dựa trên lỗ hổng CVE-2020-0796 và CVE-2020-24932, giúp tăng cường an ninh mạng bằng cách phát hiện sớm các hành vi khai thác lỗ hổng nghiêm trọng.

2. Mục đích của đề tài

- Đề tài 1: Phát triển một ứng dụng giúp bảo vệ bản quyền hình ảnh, hỗ trợ các cá nhân và tổ chức sáng tạo nội dung số.
- Đề tài 2: Xây dựng hệ thống giám sát và phát hiện tấn công nhằm giảm thiểu thiệt hại từ các cuộc tấn công mạng, đặc biệt là khai thác hai lỗ hổng bảo mật nghiêm trọng.

3. Phạm vi nghiên cứu

Đề tài 1:

- Nghiên cứu các kỹ thuật Digital Watermarking phổ biến như Least Significant Bit (LSB), Discrete Cosine Transform (DCT), và Discrete Wavelet Transform (DWT).
- Xây dựng ứng dụng bảo vệ ảnh kỹ thuật số và kiểm tra tính bền của watermark trước các dạng tấn công phổ biến.

Đề tài 2:

- Tập trung nghiên cứu hai lỗ hổng bảo mật CVE-2020-0796 và CVE-2020-24932.
- Phát triển mô hình học máy dựa trên dữ liệu lưu lượng mạng để phát hiện các cuộc tấn công.

4. Phương pháp nghiên cứu

- Nghiên cứu lý thuyết: Tìm hiểu và phân tích các khái niệm, thuật toán và kỹ thuật liên quan đến Digital Watermarking và phát hiện tấn công mạng.
- Thực nghiệm:

- + Xây dựng ứng dụng nhúng và trích xuất watermark trên hình ảnh.
- + Mô phỏng các cuộc tấn công mạng và thu thập dữ liệu để huấn luyện mô hình học máy.
- Đánh giá: Kiểm tra hiệu quả của ứng dụng bảo vệ ảnh và mô hình phát hiện tấn công thông qua các thí nghiệm thực tế.

5. Kết cấu của đề tài

Bố cục báo cáo gồm hai phần chính, mỗi phần gồm ba chương:

Phần I: Bảo vệ ảnh dùng Digital Watermarking

- Chương 1: Cơ sở lý thuyết về Digital Watermarking.
- Chương 2: Phân tích thiết kế hệ thống bảo vệ ảnh.
- Chương 3: Triển khai và đánh giá ứng dụng.
- Kết luận và hướng phát triển

Phần II: Phát hiện tấn công

- Chương 1: Cơ sở lý thuyết về lỗ hổng bảo mật và các kỹ thuật phát hiện tấn công.
- Chương 2: Phân tích thiết kế hệ thống phát hiện tấn công.
- Chương 3: Triển khai và đánh giá mô hình phát hiện.
- Kết luận và hướng phát triển

PHẦN I: Bảo vệ ảnh dùng Digital Watermarking

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 Khái niệm và vai trò của Digital Watermarking

1.1.1. Digital Watermarking là gì?

Digital Watermarking (Thủy vân số) ra đời vào những năm 1990, là một công nghệ nhúng thông tin vô hình vào phương tiện kỹ thuật số như hình ảnh, âm thanh và video. Công nghệ này xuất hiện để giải quyết các vấn đề liên quan đến bảo vệ bản quyền trong kỷ nguyên số, đặc biệt là với sự phát triển của internet và khả năng sao chép, phân phối nội dung dễ dàng. Mục đích ban đầu của digital watermarking là bảo vệ bản quyền và kiểm soát việc sao chép, nhưng nó đã phát triển để bao gồm nhiều ứng dụng tiềm năng khác, như phát hiện giả mạo, giám sát phát sóng và phát hiện tin giả [1]. Khi thế giới ngày càng trở nên kết nối và nội dung kỹ thuật số phát triển, nhu cầu về các kỹ thuật watermark hiệu quả và mạnh mẽ chưa bao giờ lớn hơn thế. Trong nội dung của quyển báo cáo này, chúng ta sẽ tiến hành xây dựng ứng dụng web bảo vệ hình ảnh kỹ thuật số bằng công nghệ Digital Watermarking.

1.1.2. Vai trò của Digital Watermarking.

Digital Watermarking đóng vai trò như một công cụ bảo mật toàn diện, giúp bảo vệ quyền sở hữu trí tuệ, xác thực nội dung, và chống giả mạo. Bên cạnh đó, nó còn hỗ trợ quản lý, theo dõi, và đảm bảo an toàn dữ liệu số trong nhiều lĩnh vực. Với tiềm năng phát triển mạnh mẽ, DWM hứa hẹn sẽ là một trong những công nghệ cốt lõi trong việc quản lý và bảo vệ thông tin số trong tương lai. Dưới đây là các vai trò chính của digital watermarking:

- Bảo vệ bản quyền số (Digital Copyright Protection):
 - + Vai trò: DWM nhúng thông tin bản quyền (ví dụ: tên tác giả, tổ chức sở hữu) vào các đối tượng đa phương tiện (hình ảnh, âm thanh, video) một cách không ảnh hưởng đến nội dung gốc. Điều này đảm bảo quyền sở hữu trí tuệ, ngăn chặn việc sử dụng hoặc phân phối trái phép nội dung.
 - + Ứng dụng: Quản lý và bảo vệ nội dung số, đặc biệt trong ngành công nghiệp sáng tạo như phim, âm nhạc và xuất bản [2] [3].
- Xác thực nội dung số (Content Authentication)
 - + Vai trò: Kiểm tra tính toàn vẹn của nội dung, đảm bảo rằng nội dung không bị chỉnh sửa hoặc giả mạo so với bản gốc.
 - + Ứng dụng: Hữu ích trong các hệ thống yêu cầu độ tin cậy cao như tài liệu pháp lý, hình ảnh chứng cứ, và dữ liệu y tế.
- Truy vết nguồn gốc và theo dõi giao dịch (Transaction Tracing & Fingerprinting)
 - + Vai trò: Nhúng các dấu vân tay kỹ thuật số (digital fingerprints) để theo dõi và xác định nguồn phát tán trái phép hoặc rò rỉ nội dung.

- + Ứng dụng: Quản lý nội dung trong phát sóng, truyền thông và hỗ trợ điều tra vi phạm bản quyền
- Chống sao chép và làm giả (Forgery and Copy Control)
 - + Vai trò: Ngăn ngừa hành vi sao chép trái phép và giả mạo nội dung bằng cách nhúng các dấu hiệu không thể nhìn thấy hoặc khó bị xóa.
 - + Ứng dụng: Bảo vệ tiền tệ, tài liệu nhạy cảm, và các tài sản trí tuệ khác khỏi bị làm giả hoặc sao chép.
- Giám sát phát sóng (Broadcast Monitoring)
 - + Vai trò: Giúp theo dõi và đảm bảo rằng các nội dung, như quảng cáo hoặc chương trình truyền hình, được phát sóng đúng thời gian và địa điểm đã định.
 - + Ứng dụng: Được sử dụng rộng rãi trong lĩnh vực quảng cáo và truyền thông.
- Tăng cường bảo mật dữ liệu (Enhanced Data Security)
 - + Vai trò: Kết hợp với các kỹ thuật khác như mã hóa (Cryptography) và ẩn dữ liệu (Steganography) để tăng cường bảo mật, chỉ cho phép người có quyền truy cập nội dung ẩn.
 - + Ứng dụng: Bảo vệ thông tin nhạy cảm và nâng cao mức độ an toàn trong lưu trữ, truyền tải dữ liệu số.
- Đồng bộ hóa và định danh nội dung (Synchronization & Identification)
 - + Vai trò: DWM hỗ trợ đồng bộ hóa nội dung và định danh chính xác các đối tượng đa phương tiện.
 - + Ứng dụng: Đồng bộ lời bài hát với nhạc hoặc video, theo dõi các nội dung phát sóng trên nhiều kênh khác nhau.

1.2 Các kỹ thuật Digital Watermarking phổ biến

Các kỹ thuật digital watermarking phổ biến được phân loại theo nhiều tiêu chí khác nhau, chủ yếu dựa trên tính chất hiển thị, độ bền và miền áp dụng

- Phân loại theo tính chất hiển thị:
 - + Watermark hữu hình (Visible watermarking): Là watermark được chèn vào đối tượng và người dùng có thể thấy rõ ràng, ví dụ như logo hoặc văn bản
 - + Watermark vô hình (Invisible watermarking): Là watermark được thêm vào đối tượng nhưng không thể nhìn thấy bằng mắt thường và cần thuật toán đặc biệt để trích xuất. Watermark vô hình thường được sử dụng để xác thực dữ liệu và ngăn chặn sao chép bất hợp pháp
 - + Watermark kép (Dual watermarking): Là sự kết hợp của watermark hữu hình và vô hình. Đầu tiên, một watermark hữu hình được thêm vào, sau đó một watermark vô hình được nhúng vào nội dung đã có watermark hữu hình [1].

- Phân loại theo độ bền:
 - + Watermark dễ vỡ (Fragile watermark): Loại watermark này rất nhạy cảm và dễ bị phá hủy bởi bất kỳ tấn công nào, có thể bị thao túng trước khi đến tay người nhận. Watermark dễ vỡ được sử dụng để kiểm tra tính toàn vẹn và xác thực nội dung.
 - + Watermark bán dễ vỡ (Semi-fragile watermark): Loại watermark này có thể chịu được một số loại tấn công nhất định, nhưng không thể chống lại các loại tấn công khác. Watermark bán dễ vỡ thường được sử dụng để xác minh nội dung.
 - + Watermark mạnh mẽ (Robust watermark): Loại watermark này có khả năng chống lại mọi loại tấn công, được thiết kế để bảo vệ bản quyền. Watermark mạnh mẽ có thể được phát hiện ngay cả sau khi dữ liệu bị tấn công.
- Phân loại theo miền (Domain):
 - + Watermark miền không gian (Spatial domain watermarks): Kỹ thuật này chèn watermark trực tiếp vào các pixel của hình ảnh. Bất kỳ thay đổi nào ở bất kỳ vị trí nào trong hình ảnh sẽ được phản ánh trong khung cảnh tương ứng. Các kỹ thuật phổ biến trong miền không gian bao gồm LSB (Least Significant Bit) và SSM (Spread Spectrum Modulation).
 - + Watermark miền tần số (Frequency/transform domain watermarks): Kỹ thuật này nhúng watermark vào các hệ số tần số sau khi chuyển đổi ảnh từ miền không gian sang miền tần số. Các kỹ thuật phổ biến trong miền tần số bao gồm DCT (Discrete Cosine Transform), DFT (Discrete Fourier Transform), DWT (Discrete Wavelet Transform), và SVD (Singular Value Decomposition).
 - + Watermark miền wavelet (Wavelet domain watermarks): Sử dụng các biến đổi wavelet để phân tích và nhúng watermark. Các biến đổi này tạo ra công cụ mạnh mẽ để phân tích miền gốc. Kỹ thuật này sử dụng DWT để phân tách hình ảnh thành ba độ phân giải khác nhau, sau đó watermark được nhúng vào các khu vực có độ phân giải cao.
- Một số loại kỹ thuật khác:
 - + Watermark dựa trên histogram (Histogram modification-based schemes): Sử dụng các kỹ thuật sửa đổi histogram của hình ảnh để nhúng watermark.
 - + Watermark có khả năng phục hồi (Reversible watermarking): Cho phép xác thực dữ liệu đồng thời khôi phục dữ liệu gốc một cách không mất mát. Kỹ thuật này được sử dụng trong các lĩnh vực như pháp y kỹ thuật số, quân sự, y tế.
 - + Watermark tự nhúng (Self-embedding watermarking): Sử dụng một phần của dữ liệu làm watermark để xác thực và phục hồi dữ liệu.

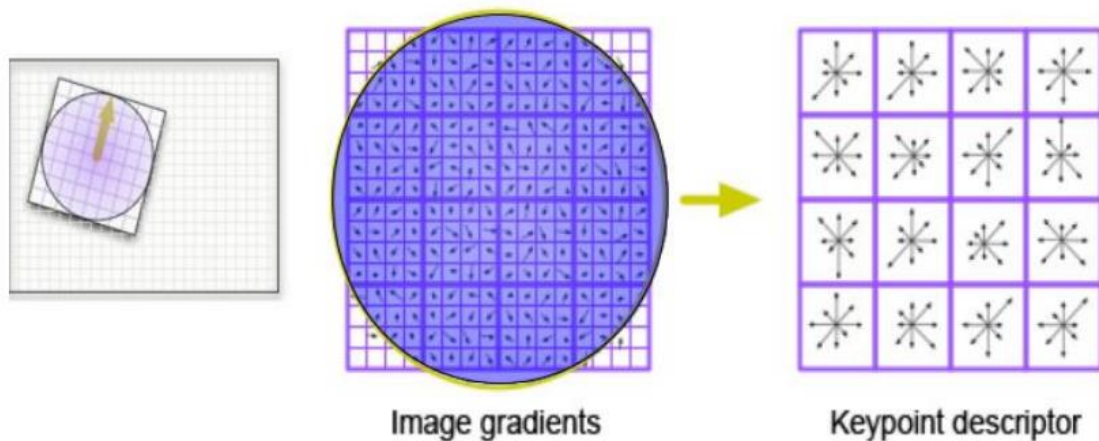
1.3 Phân tích các kỹ thuật và thuật toán Digital Watermarking phổ biến.

1.3.1. Scale-Invariant Feature Transform (SIFT).

SIFT (Scale Invariant Feature Transform) là đặc trưng cục bộ bất biến đối với những phép biến đổi tỷ lệ, tịnh tiến, phép quay, và không đổi một phần đối với những thay đổi về góc nhìn, đồng thời nó cũng rất mạnh với những thay đổi về độ sáng, sự che khuất, nhiễu. Phương pháp trích rút đặc trưng SIFT có thể được tóm tắt như sau:

- (1) Phát hiện các điểm cực trị trong không gian tỷ lệ (Scale-Space extrema detection): Sử dụng hàm sai khác Gaussian (Different of Gaussian - DoG) để xác định tất cả các điểm hấp dẫn tiềm năng mà bất biến với tỷ lệ và hướng của ảnh;
- (2) Định vị các điểm hấp dẫn (Keypoint localization): Ứng với mỗi vị trí tiềm năng, hàm kiểm tra sẽ được đưa ra để quyết định xem các điểm hấp dẫn tiềm năng có được lựa chọn hay không. Các điểm hấp dẫn được lựa chọn dựa trên việc đo lường tính ổn định của chúng;
- (3) Xác định hướng cho các điểm hấp dẫn (Orientation assignment): Một hoặc nhiều hướng được gán cho mỗi vị trí điểm hấp dẫn dựa trên hướng gradient cục bộ của ảnh;

Mô tả các điểm hấp dẫn (Keypoint descriptor): Các gradient ảnh cục bộ được xác định ở tỷ lệ được chọn trong vùng bao quanh mỗi điểm hấp dẫn. Các gradient được biểu diễn sang một dạng mà cho phép bất biến với sự thay đổi về hình dạng và điều kiện chiếu sáng.

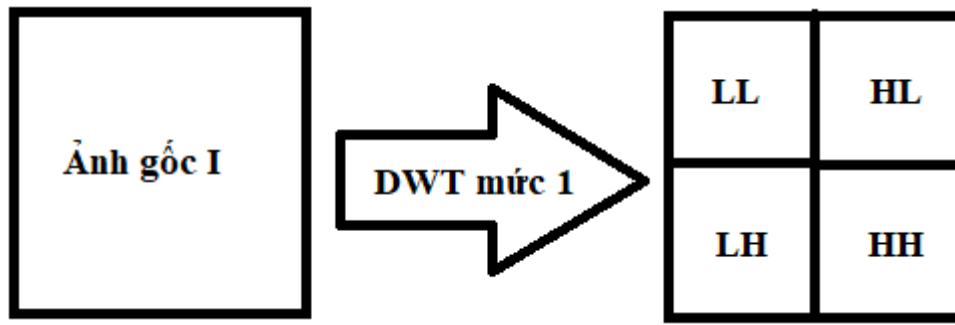


Hình 1: Hình ảnh sau khi áp dụng biến đổi SIFT

1.3.2. Discrete Wavelet Transform (DWT).

Discrete Wavelet Transform (DWT) là một kỹ thuật toán học thường được sử dụng trong xử lý tín hiệu và nén ảnh. Trong phép biến đổi DWT hai chiều, một ảnh gốc I sẽ được phân tích thành bốn băng tần có kích thước bằng $\frac{1}{2}$ ảnh gốc: LL (low frequency component in horizontal and vertical direction), LH (low frequency component in horizontal direction and high frequency in vertical direction), HL (high frequency component in horizontal direction and low frequency in vertical direction), HH (high frequency component in horizontal direction and high frequency in vertical direction).

Hình 1. biểu diễn cho sự phân tích ảnh gốc $I(N \times N)$ thành bốn băng tần LL, LH, HL, HH có kích thước $(N/2, N/2)$.



Hình 2: Biến đổi DWT mức 1

Các kỹ thuật thủy văn sử dụng phép biến đổi DWT thường nhúng watermark vào một hoặc một số băng tần với các hệ số tương quan khác nhau. Do LL có tần số thấp và chứa các thông tin quan trọng của ảnh nên mọi sự thay đổi nhỏ sẽ ảnh hưởng đến chất lượng hình ảnh. Tương tự băng tần HH có tần số cao nên cũng không bền vững trước các tấn công như nén JPEG. Do đó trong bài báo này các băng tần HL, LH sẽ được sử dụng để nhúng watermark. Sau đó việc xây dựng lại ảnh thủy văn I' từ các băng tần đã nhúng watermark được thực hiện bởi phép biến đổi ngược IDWT (Inverse Discrete Wavelet Transform).

1.3.3. Discrete Cosine Transform (DCT).

DCT là một kỹ thuật chuyển đổi ảnh từ miền không gian sang miền tần số. Trong phép biến đổi DCT, ảnh ban đầu được chia thành các khối có kích thước $N \times N$ giống nhau, và phép biến đổi DCT được áp dụng trên các khối $N \times N$ theo công thức (1).

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

Trong đó: $C(e) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{nếu } e = 0 \\ 1, & \text{ngược lại} \end{cases}$

Trong công thức biến đổi DCT, $f(x, y)$ là giá trị điểm ảnh tại vị trí (x, y) và $F(u, v)$ là hệ số DCT ở vị trí (u, v) của ma trận hệ số DCT [4].

1.3.4. Singular Value Decomposition (SVD).

SVD là một phép biến đổi thường được sử dụng trong xử lý tín hiệu và phân tích thành phần chính. Trong phép biến đổi SVD, một ma trận M được phân tích thành ba ma trận có cùng kích thước với M theo công thức sau:

$$M = U * S * V^T \quad (2)$$

Trong đó:

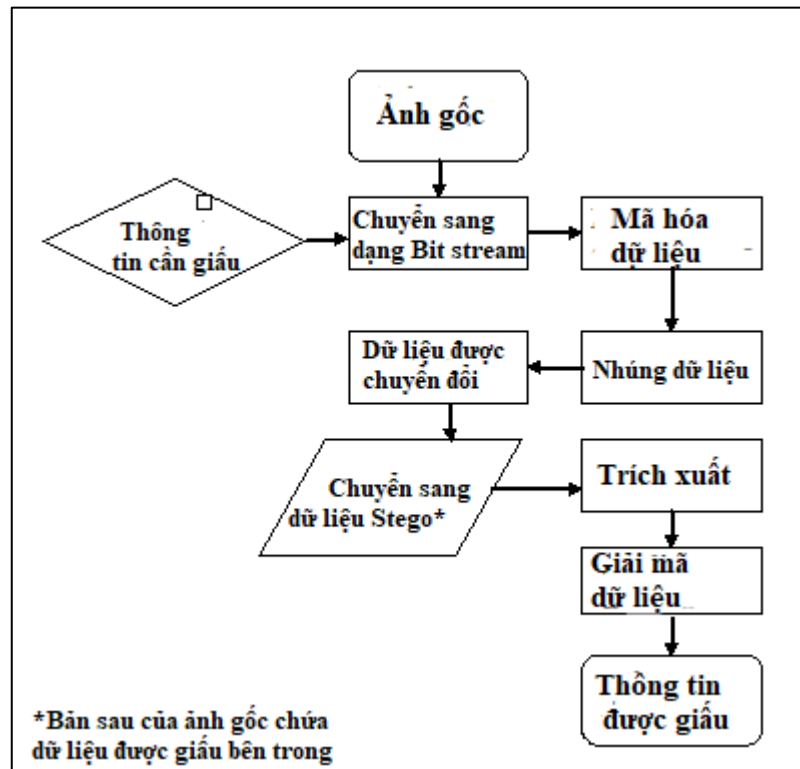
- M : ma trận cấp $N \times N$
- U, V : ma trận trực chuẩn cấp $N \times N$
- S : ma trận đường chéo không âm cấp $N \times N$

$$S_{NxN} = \begin{bmatrix} \alpha(1,1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha(N,N) \end{bmatrix}$$

Các phần tử nằm bên trong ma trận S ở trên được gọi là các giá trị riêng, thỏa điều kiện $\alpha(1,1) \geq \alpha(2,2) \geq \cdots \geq \alpha(N,N) \geq 0$. Các giá trị riêng này có tính ổn định cao và chứa đựng phần lớn thông tin của ảnh [4].

1.3.5. Least Significant Bit (LSB).

Thuật toán LSB (Least Significant Bit) là một kỹ thuật phổ biến trong watermarking kỹ thuật số, thuộc miền không gian (spatial domain), được sử dụng để ẩn dữ liệu vào hình ảnh [2]. Đây là một phương pháp đơn giản và dễ thực hiện, nhưng cũng có những hạn chế nhất định.



Hình 3: Thuật toán LSB

Ta nhúng trực tiếp các bit của thông điệp cần gửi vào các bit ít quan trọng của ảnh chủ thể. Trong phương pháp này, bit hình mờ được chèn ở phía ngoài cùng bên phải của pixel. Sự thay đổi các bit ít quan trọng sẽ không tạo ra sự khác biệt mà mắt người có thể nhận ra, nguyên nhân là do biên độ của sự thay đổi nhỏ.

Thuật toán cơ bản để thay thế LSB là lấy N pixel bìa đầu tiên trong đó N là tổng chiều dài của thông điệp bí mật sẽ được nhúng theo bit. Sau đó, mọi pixel cuối cùng sẽ được thay thế bằng một trong các bit thông báo [5].

Ví dụ: giả sử chúng ta có hai pixel liên kề (sáu byte) với mã hóa RGB sau.

```

10110101 01001101 11001101
00010011 00010100 01001010
  
```

Bây giờ, giả sử chúng ta muốn ẩn 6 bit dữ liệu sau 001001. Nếu chúng ta phủ 6 bit này lên LSB của 6 byte ở trên, chúng ta sẽ nhận được kết quả sau, trong đó các bit in đậm đã được thay đổi.

10110100 01001100 11001101

00010010 00010100 01001011

Vậy là ta đã thành công trong việc giấu 6 bit, nhưng chỉ phải thay đổi 4 bit trong số đó (khoảng 66,75%) của các bit LSB. Đề tài này sẽ ứng dụng việc giấu dữ liệu văn bản/hình ảnh (tương trưng cho việc khẳng định bản quyền của người sở hữu) bí mật vào trong hình ảnh bằng phương pháp Least Significant Bit (LSB).

1.4 Tiêu chí đánh giá Watermark.

Để đánh giá một Watermark có ổn định và bền vững trước các cuộc tấn công hay không, cần dựa vào các tiêu chí sau:

- Độ bền (Robustness): Watermark phải tồn tại và có thể được trích xuất ngay cả khi ảnh bị biến đổi do các tác động như nén (JPEG, GIF), xoay, cắt, thay đổi kích thước, thêm nhiễu (Gaussian noise, Salt & Pepper noise), lọc (blurring, sharpening),... Đây là tiêu chí quan trọng nhất, thể hiện khả năng chống chịu của watermark trước các tấn công thông thường.
- Tính vô hình (Imperceptibility): Watermark không nên làm ảnh hưởng đáng kể đến chất lượng cảm nhận của ảnh. Sự thay đổi do watermark gây ra phải không thể nhận thấy bằng mắt thường. Chất lượng ảnh thường được đo bằng các chỉ số như PSNR (Peak Signal-to-Noise Ratio) hoặc SSIM (Structural Similarity Index), với giá trị càng cao thì chất lượng ảnh càng tốt.
- Tính bảo mật (Security): Chỉ những người có khóa bí mật hoặc thông tin được ủy quyền mới có thể trích xuất và đọc được watermark. Watermark phải chống lại các tấn công nhằm mục đích xóa, sửa đổi hoặc giả mạo watermark.
- Tính chống giả mạo (Tamper-proof): Watermark phải đủ mạnh để chống lại các hành vi cố ý chỉnh sửa hoặc tấn công ác ý nhằm loại bỏ hoặc làm sai lệch thông tin của watermark.
- Dung lượng (Capacity): Khả năng nhúng được bao nhiêu thông tin vào watermark. Dung lượng càng cao thì có thể nhúng được nhiều thông tin hơn, nhưng thường đi kèm với việc giảm độ bền hoặc tính vô hình.
- Tính mù (Blindness): Khả năng trích xuất watermark mà không cần đến ảnh gốc. Nếu cần ảnh gốc để trích xuất watermark, ta gọi là phương pháp không mù (non-blind). Tính mù là một ưu điểm lớn, đặc biệt trong các ứng dụng thực tế.
- Khả năng tính toán (Computational cost): Độ phức tạp tính toán của thuật toán nhúng và trích xuất watermark. Thuật toán càng phức tạp thì càng tốn nhiều thời gian và tài nguyên tính toán.

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ

2.1 Phân tích yêu cầu hệ thống

Yêu cầu chức năng:

- Nhúng Watermark vào ảnh
 - + Cho phép tải ảnh lên và thêm watermark ở dạng ẩn và nổi vào hình ảnh
 - + Xử lý nhúng watermark vào ảnh sử dụng thuật toán Digital Watermarking
 - + Đảm bảo watermark được nhúng không làm giảm đáng kể chất lượng ảnh gốc.
 - + Lưu ảnh đã nhúng watermark trên hệ thống hoặc cung cấp chức năng tải về.
- Kiểm tra ảnh có chứa watermark
 - + Cho phép tải lên một ảnh bất kỳ để kiểm tra xem ảnh có chứa watermark hay không.
 - + Trích xuất watermark từ ảnh đã nhúng và hiển thị nội dung của watermark

Yêu cầu phi chức năng:

- + Thiết kế giao diện dễ sử dụng với hướng dẫn rõ ràng cho từng chức năng (như tải ảnh, nhúng watermark, kiểm tra ảnh).
- + Hệ thống phải đảm bảo hiệu suất xử lý nhanh (thời gian nhúng/rút watermark không vượt quá 2 giây/hình ảnh).
- + Đảm bảo an toàn bảo mật dữ liệu hình ảnh và watermark.

Yêu cầu kỹ thuật:

- Sử dụng Python và thư viện hỗ trợ xử lý ảnh (OpenCV, Pillow).
- Kết nối với cơ sở dữ liệu để lưu trữ thông tin người dùng, hình ảnh và watermark.
- Hệ thống web sử dụng framework Flask.

2.2 Thiết kế thuật toán nhúng và rút trích watermark.

Nhúng watermark:

- Lựa chọn kỹ thuật nhúng LSB để nhúng watermark vào ảnh.
- Xác định vị trí nhúng là ở bit cuối cùng của kênh màu xanh (blue channel) với ảnh RGB.

Thiết kế thuật toán:

- Chuyển ảnh đầu vào (ảnh gốc) sang dạng RGB và ảnh watermark sang dạng ảnh xám (grayscale).
- Nhúng watermark vào ảnh bằng cách chuyển hình ảnh và watermark sang dạng numpy sau đó nhúng watermark vào bit cuối cùng của kênh màu xanh.
- Tái tạo lại hình ảnh với watermark đã được nhúng.
- Mã nguồn của thuật toán nhúng watermark:

```
def embed_watermark(image, watermark):  
    # Resize watermark để phù hợp với kích thước ảnh gốc  
    # Sử dụng bộ nội suy LANCZOS để đảm bảo watermark có chất lượng tốt sau  
    khi resize  
    watermark_resized = watermark.resize((image.width, image.height),  
Image.Resampling.LANCZOS)
```



```

# Chuyển đổi ảnh và watermark sang mảng numpy để xử lý pixel
image_array = np.array(image) # Mảng 3D (chiều cao x chiều rộng x kênh màu)
watermark_array = np.array(watermark_resized) # Mảng 2D (chiều cao x chiều rộng)

# Tạo bản sao của ảnh gốc để nhúng watermark
watermarked_array = image_array.copy()

# Nhúng watermark vào kênh màu xanh (blue channel)
# '& 0b11111110' để đặt bit cuối cùng của kênh màu xanh về 0
# '| watermark_array' để thêm giá trị watermark vào bit cuối
watermarked_array[..., 2] = (image_array[..., 2] & 0b11111110) | watermark_array

# Chuyển mảng numpy trở lại định dạng ảnh Pillow
return Image.fromarray(watermarked_array)

```

Rút trích watermark:

- Xác định các bước trích xuất (ngược lại quy trình nhúng).
- Đảm bảo thuật toán có khả năng phát hiện watermark ngay cả khi ảnh bị chỉnh sửa nhẹ (thay đổi xoay, cắt xén, v.v.).
- Mã nguồn của thuật toán trích xuất:

```

def extract_watermark(watermarked_image):
    # Chuyển đổi ảnh đã nhúng sang mảng numpy
    watermarked_array = np.array(watermarked_image) # Mảng 3D (chiều cao x chiều rộng x kênh màu)

    # Lấy giá trị bit cuối cùng từ kênh màu xanh (blue channel)
    # '& 0b00000001' để giữ lại bit cuối cùng, sau đó nhân với 255 để hiển thị rõ
    extracted_watermark = (watermarked_array[..., 2] & 0b00000001) * 255

    # Chuyển mảng numpy trở lại định dạng ảnh Pillow
    return Image.fromarray(extracted_watermark)

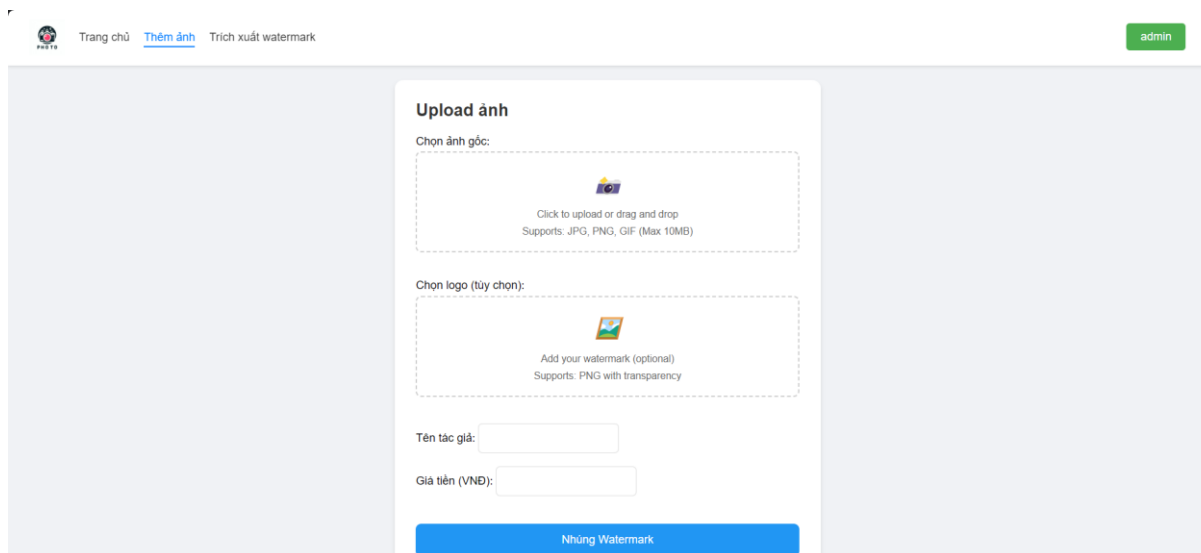
```

2.3 Thiết kế kiến trúc hệ thống.

2.3.1. Giao diện người dùng.

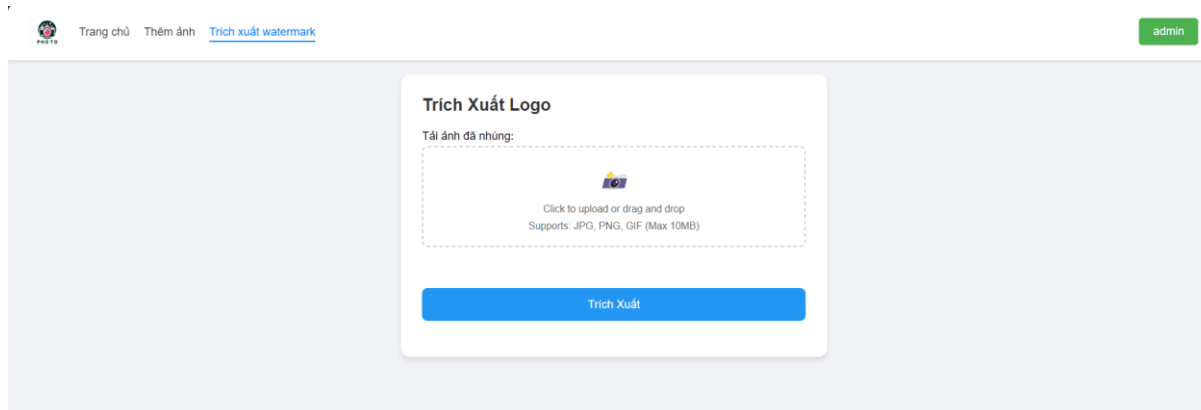
Mô tả giao diện:

- Trang tải ảnh lên: Người dùng có thể tải lên và nhúng ảnh với watermark.



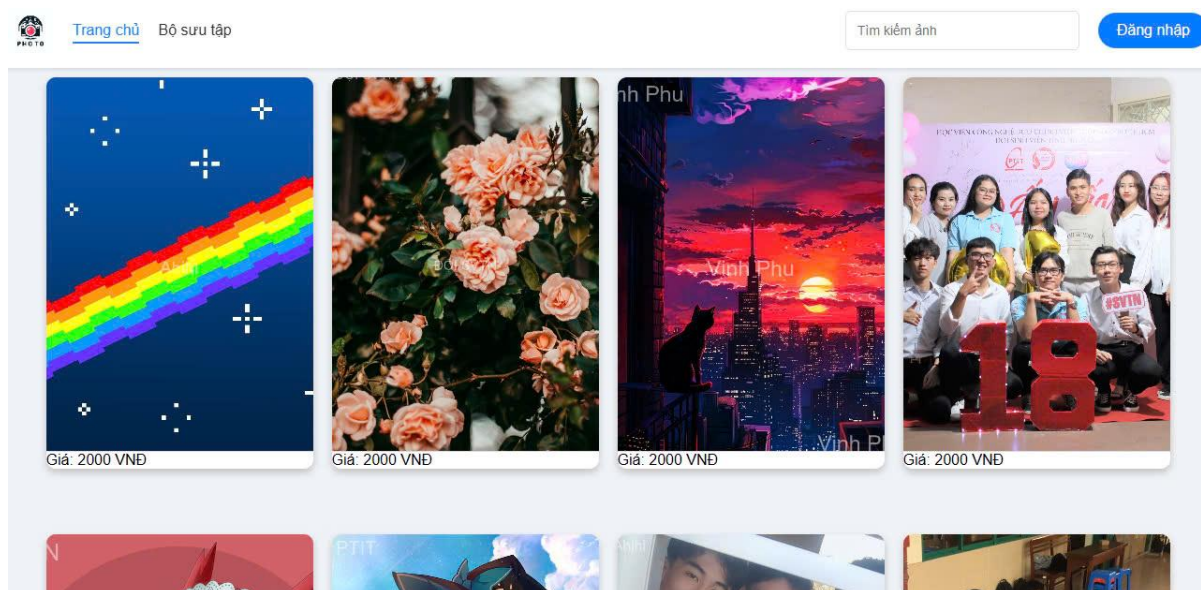
Hình 4: Giao diện upload ảnh

– Trang trích xuất watermark



Hình 5: Giao diện trích xuất watermark

– Trang chủ



Hình 6: Trang chủ

– Trang đăng nhập

Đăng nhập

Tên đăng nhập

Mật khẩu

[Quên mật khẩu?](#)

Đăng nhập

Chưa có tài khoản? [Đăng ký ngay](#)

Hình 7: Trang đăng nhập

– Trang đăng ký

Đăng ký tài khoản

Họ và tên

Email

a

Email không hợp lệ

Tên đăng nhập

Mật khẩu

.

Mật khẩu phải có ít nhất 8 ký tự, bao gồm chữ hoa, chữ thường và số

Xác nhận mật khẩu

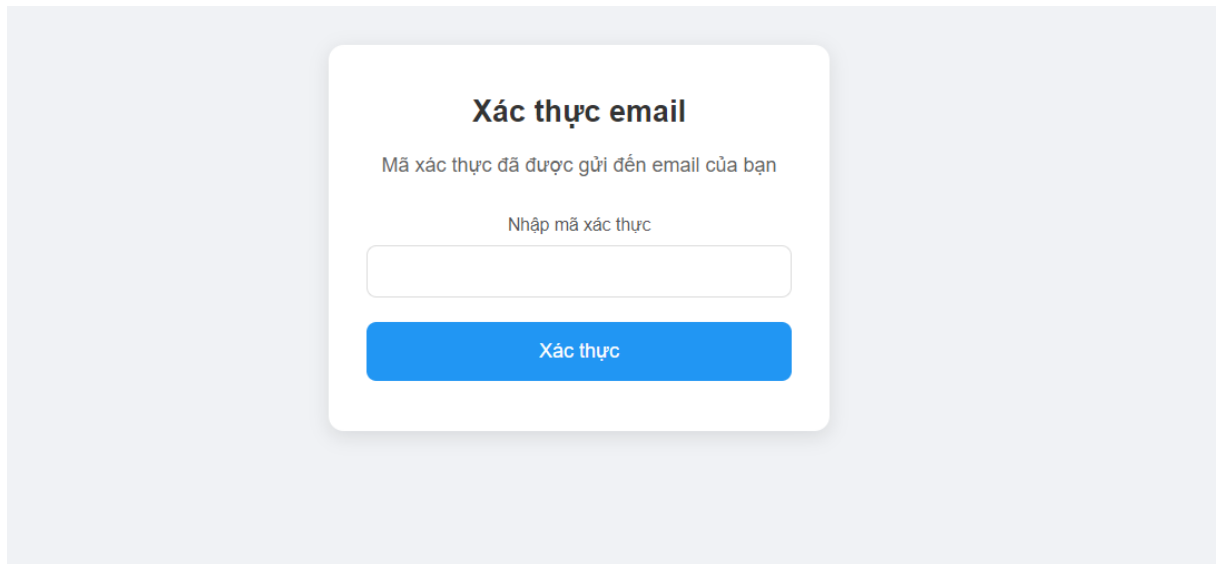
..

Mật khẩu không khớp

Đăng ký

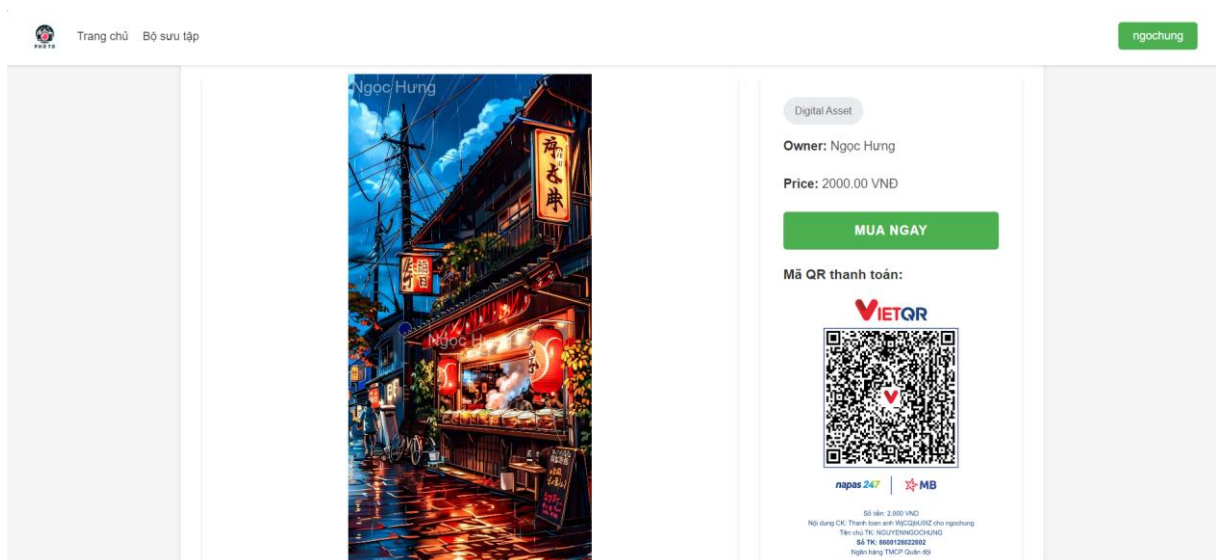
Hình 8: Trang đăng ký

– Trang xác thực email



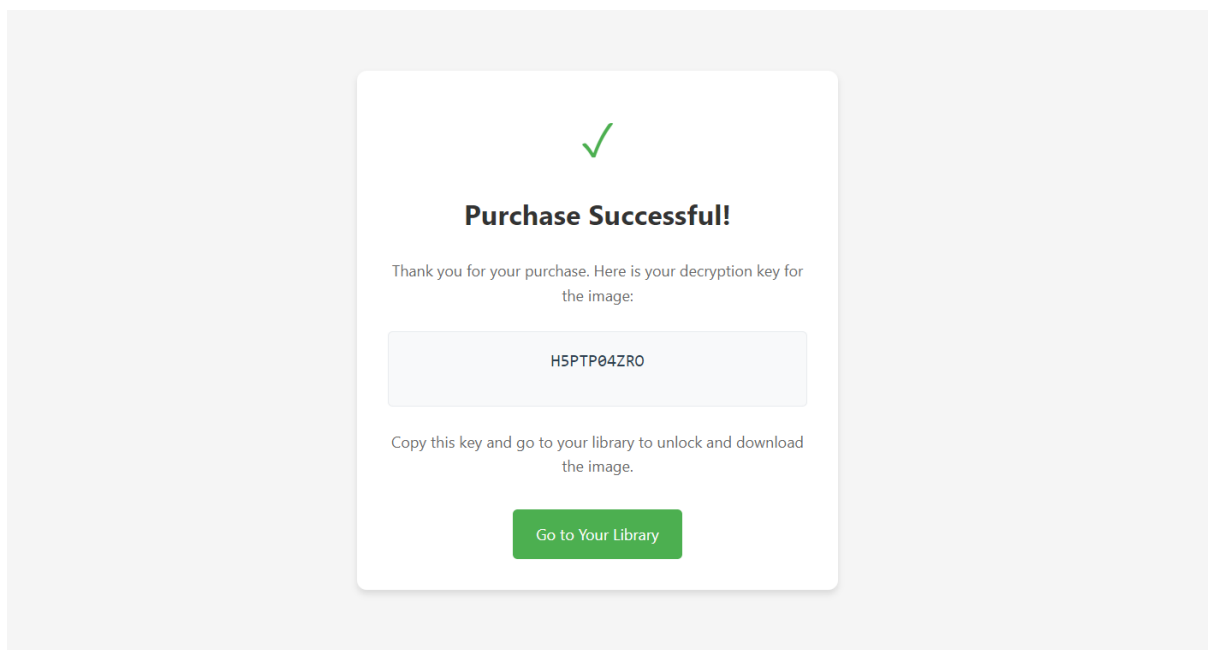
Hình 9: Trang xác thực email

- Trang chi tiết ảnh: Hiển thị ảnh mẫu có watermark và QR code thanh toán khi khách hàng có nhu cầu mua hàng.



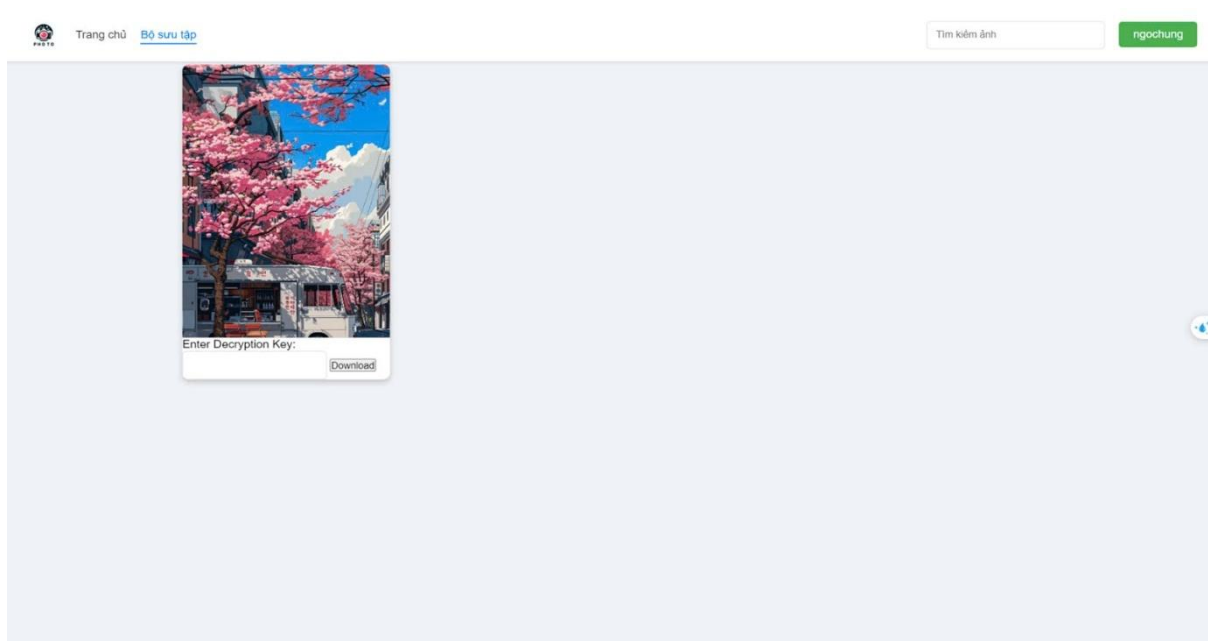
Hình 10: Trang chi tiết ảnh và thanh toán

- Trang trả về key của ảnh



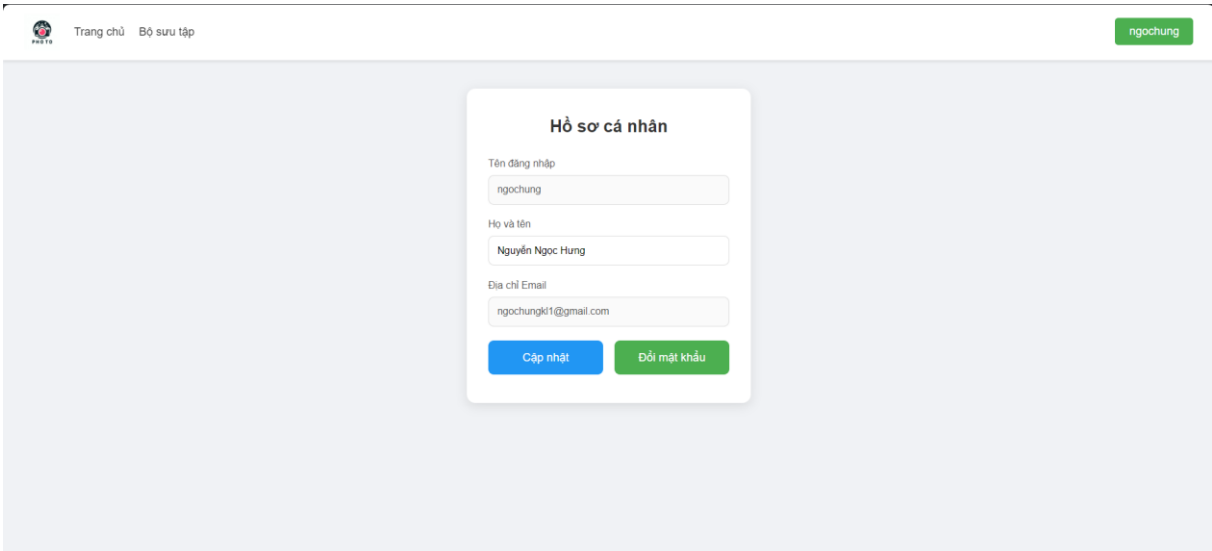
Hình 11: Trang kết quả key của ảnh

– Trang bộ sưu tập



Hình 12: Trang bộ sưu tập

– Trang hồ sơ cá nhân



Hình 13: Trang hồ sơ cá nhân

Công cụ thiết kế giao diện:

- Sử dụng HTML, CSS, JavaScript.

2.3.2. Hệ thống xử lý phía server.

Xử lý nhúng và rút trích watermark:

- Server nhận yêu cầu từ admin, thực hiện nhúng/rút watermark và trả kết quả về.

Quản lý cơ sở dữ liệu:

- Lưu trữ thông tin hình ảnh, watermark, và lịch sử người dùng.
- Cơ sở dữ liệu lưu trữ là SQL Server 2019.

CHƯƠNG 3: TRIỂN KHAI

3.1 Cài đặt hệ thống.

3.1.1. Cấu hình môi trường phát triển.

Các công cụ và công nghệ sử dụng:

- Ngôn ngữ lập trình: Python (v3.8+).
- Thư viện chính:
 - + Pillow (xử lý ảnh).
 - + NumPy (xử lý mảng dữ liệu).
 - + Flask (xây dựng ứng dụng web).
- Cơ sở dữ liệu: SQL Server 2019.
- Môi trường: IDE như PyCharm, VS Code, hoặc bất kỳ công cụ nào hỗ trợ Python.

Quy trình cài đặt môi trường:

- Cài đặt Python và các gói thư viện cần thiết: *pip install pillow numpy flask*.
- Cấu hình Flask để triển khai giao diện người dùng và tích hợp server.
- Kết nối ứng dụng với cơ sở dữ liệu để lưu trữ thông tin người dùng và ảnh.

3.1.2. Tích hợp thuật toán với ứng dụng web.

Các bước tích hợp:

Nhúng thuật toán nhúng và trích xuất watermark:

- Tích hợp các hàm `embed_watermark` và `extract_watermark` từ chương trước vào lớp xử lý ảnh trên server.

Kết nối API, tạo endpoint API để xử lý yêu cầu từ giao diện web:

- Nhúng watermark (Upload ảnh -> Nhúng watermark -> Trả ảnh đã nhúng).
- Rút trích watermark (Tải ảnh -> Trích xuất watermark -> Trả kết quả).

Tích hợp giao diện:

- Liên kết giao diện upload ảnh với server để tải và xử lý ảnh.

3.2 Thử nghiệm và đánh giá.

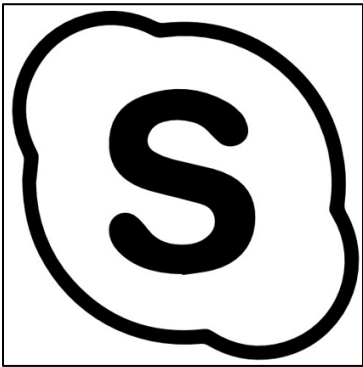
3.2.1. Đánh giá chất lượng ảnh sau khi nhúng watermark.

Các tiêu chí đánh giá:

- Chất lượng hình ảnh: Sử dụng các chỉ số như PSNR (Peak Signal-to-Noise Ratio) hoặc SSIM (Structural Similarity Index) để đánh giá sự khác biệt giữa ảnh gốc và ảnh đã nhúng watermark.
- Thị giác: Kiểm tra bằng mắt thường xem ảnh có sự khác biệt rõ rệt hay không.

Ví dụ so sánh ảnh trước và sau khi nhúng watermark bằng chỉ số SSIM:

- Tiến hành nhúng logo sau vào ảnh



Hình 14: Logo



Hình 16: Hình ảnh trước khi thêm watermark



Hình 15: Hình sau khi thêm watermark ẩn


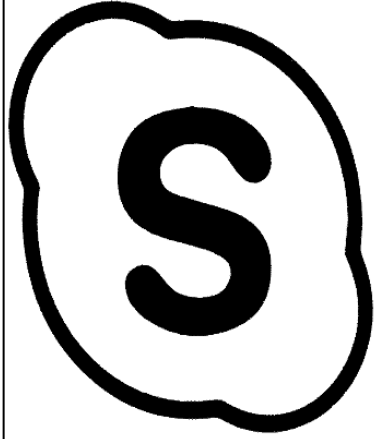

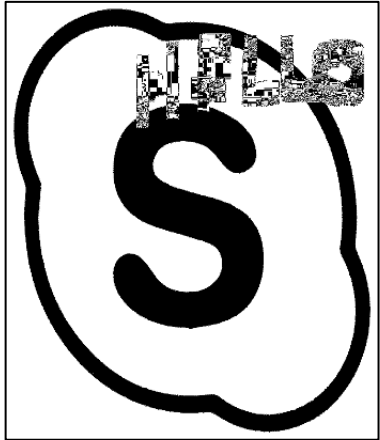

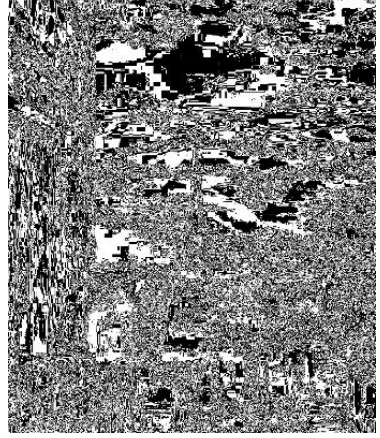
Kết quả so sánh: ***SSIM between original and embed watermark: 0.9331156653593432***
Từ kết quả và mắt thường cho thấy thuật toán nhúng không thay đổi nhiều về hình ảnh.

3.2.2. Kiểm tra tính bền vững của watermark.

Phương pháp thử nghiệm:

Bảng 1: Bảng đánh giá tính bền vững của watermark

| Loại tấn công | Ảnh thử nghiệm | Kết quả |
|--------------------|----------------|---------|
| Tấn công đơn giản | | |
| Cắt xén và lật ảnh | | |

| | | |
|---------------------------|---|---|
| <p>Xoay ảnh</p> |  |  |
| Tấn công nâng cao | | |
| <p>Vẽ lên ảnh</p> |  |  |
| <p>Tăng độ tương phản</p> |  |  |

Đánh giá:

- Qua việc thực hiện các kỹ thuật tấn công hình ảnh đã nhúng watermark thì ta thấy các dạng tấn công đơn giản như cắt, xoay, lật thì watermark có thể trích xuất được nhưng sẽ thay đổi theo sự thay đổi của ảnh được nhúng. Còn các dạng tấn công nâng cao thì watermark khó có thể trích xuất được bởi vì các dạng tấn công này dễ dàng thay đổi làm ảnh hưởng tới kênh màu xanh (blue channel) nơi mà ta nhúng ảnh vào đó.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận

4.1.1. Tổng kết nội dung nghiên cứu:

- Đề tài đã nghiên cứu và triển khai thành công ứng dụng bảo vệ hình ảnh số bằng công nghệ Digital Watermarking, tập trung vào các kỹ thuật nhúng watermark ẩn vào ảnh và kiểm tra ảnh có chứa watermark.
- Thuật toán nhúng watermark sử dụng kỹ thuật Least Significant Bit (LSB) cho phép đảm bảo tính bảo mật, tính vô hình và khả năng trích xuất watermark hiệu quả.
- Ứng dụng đã được thử nghiệm với các dạng tấn công đơn giản như cắt, xoay, và thêm nhiễu, và cho thấy khả năng chống chịu của watermark trong các trường hợp này.

4.1.2. Kết quả đạt được:

- Hệ thống bảo vệ bản quyền hình ảnh hoạt động ổn định, cho phép nhúng và trích xuất watermark mà không làm giảm đáng kể chất lượng hình ảnh gốc.
- Đạt được các tiêu chí quan trọng như độ bền, tính vô hình, và tính bảo mật của watermark.
- Ứng dụng hỗ trợ kiểm tra bản quyền hình ảnh dễ dàng, tạo tiền đề cho các giải pháp bảo vệ nội dung số.

4.1.3. Đóng góp của nghiên cứu:

- Cung cấp giải pháp thực tiễn để bảo vệ quyền sở hữu trí tuệ trong lĩnh vực sáng tạo nội dung số.
- Góp phần nâng cao nhận thức về vai trò của watermark trong quản lý và bảo vệ hình ảnh kỹ thuật số.

4.2 Hướng phát triển

4.2.1. Cải tiến thuật toán và hệ thống

- Nghiên cứu và áp dụng thêm các kỹ thuật watermark mạnh mẽ hơn, như Discrete Wavelet Transform (DWT) hoặc Discrete Cosine Transform (DCT), để tăng cường khả năng chống lại các tấn công nâng cao.
- Kết hợp watermark với các công nghệ bảo mật khác, như mã hóa hoặc steganography, để nâng cao tính an toàn và tính bảo mật.

4.2.2. Mở rộng khả năng chống tấn công

- Tăng cường khả năng bảo vệ watermark trước các dạng tấn công phức tạp hơn như nén ảnh JPEG mạnh, lọc nhiễu, và thay đổi độ sáng.

- Nghiên cứu các kỹ thuật chống giả mạo và phục hồi watermark trong các môi trường có mức độ biến đổi cao.

4.2.3. Ứng dụng thực tế:

- Phát triển hệ thống thành ứng dụng thương mại với giao diện thân thiện, cung cấp dịch vụ bảo vệ hình ảnh cho các cá nhân hoặc tổ chức sáng tạo nội dung.
- Tích hợp thêm tính năng kiểm tra và xác thực ảnh trên nhiều nền tảng, như trang web bán ảnh hoặc dịch vụ lưu trữ nội dung số.
- Xây dựng phiên bản ứng dụng trên các nền tảng di động để tăng khả năng tiếp cận người dùng.

PHẦN II: PHÁT HIỆN TẤN CÔNG

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

5.1 Tổng quan về an ninh mạng

- An ninh mạng: An ninh mạng là phương pháp bảo vệ an toàn cho máy tính, mạng, ứng dụng phần mềm, hệ thống quan trọng và dữ liệu khỏi các mối đe dọa kỹ thuật số tiềm ẩn. Các tổ chức chịu trách nhiệm bảo mật dữ liệu để duy trì lòng tin của khách hàng cũng như đáp ứng việc tuân thủ quy định. Họ sử dụng các biện pháp và công cụ an ninh mạng để bảo vệ dữ liệu nhạy cảm khỏi bị truy cập trái phép cũng như ngăn chặn gián đoạn trong hoạt động kinh doanh gây ra bởi hoạt động mạng ngoài ý muốn. Các tổ chức triển khai an ninh mạng bằng cách hợp lý hóa công tác phòng vệ kỹ thuật số giữa con người, quy trình và công nghệ.
- Mục tiêu của an ninh mạng:
 - + Ngăn chặn hoặc giảm tổn thất do vi phạm
 - + Duy trì tuân thủ theo quy định
 - + Giảm thiểu các mối đe dọa mạng không ngừng biến hóa

5.2 Một số loại tấn công mạng phổ biến

5.2.1. Tấn công từ chối dịch vụ (DoS/DDoS):

- DoS: Một máy tính gửi một lượng lớn yêu cầu đến máy chủ mục tiêu, khiến máy chủ quá tải và không thể phục vụ các yêu cầu hợp lệ khác.
- DDoS: Nhiều máy tính bị nhiễm mã độc cùng lúc tấn công một mục tiêu, tạo ra lượng truy cập khổng lồ và làm sập hệ thống.
- Ví dụ: Một website bán hàng trực tuyến bị tấn công DDoS vào ngày Black Friday, khiến khách hàng không thể truy cập và mua hàng.

5.2.2. Tấn công man-in-the-middle (MITM):

- Kẻ tấn công đặt mình vào giữa quá trình truyền thông giữa hai bên, đánh cắp thông tin nhạy cảm như mật khẩu, số thẻ tín dụng.
- Ví dụ: Một kẻ tấn công thiết lập một điểm truy cập Wi-Fi giả mạo tại một quán cà phê. Khi bạn kết nối, kẻ tấn công có thể xem và ghi lại tất cả lưu lượng truy cập của bạn.

5.2.3. Tấn công SQL Injection

- Kẻ tấn công chèn mã SQL độc hại vào các trường nhập liệu của ứng dụng web, khai thác lỗ hổng để truy cập và thao tác với cơ sở dữ liệu.
- Ví dụ: Một kẻ tấn công chèn một câu lệnh SQL vào ô tìm kiếm của một trang web thương mại điện tử để lấy danh sách tất cả mật khẩu của người dùng.

5.2.4. Tấn công thực thi mã từ xa (RCE)

- Kẻ tấn công lợi dụng lỗ hổng trong phần mềm hoặc hệ điều hành để thực thi mã độc trên hệ thống mục tiêu.
- Ví dụ: Một máy chủ web có lỗ hổng, kẻ tấn công khai thác lỗ hổng này để cài đặt một chương trình đào tiền ảo, chiếm dụng tài nguyên hệ thống.

5.2.5. Các loại tấn công khác

Ngoài các loại tấn công trên, còn có nhiều loại tấn công khác như:

- Phishing: Lừa đảo qua email, tin nhắn để đánh cắp thông tin cá nhân.
- Malware: Phần mềm độc hại như virus, worm, ransomware.
- Zero-day exploit: Khai thác lỗ hổng chưa được vá của phần mềm.
- Session hijacking: Chiếm quyền điều khiển phiên làm việc của người dùng.

5.3 Các lỗ hổng bảo mật nổi bật

5.3.1. CVE-2020-0796 (SMBGhost)

- Giới thiệu chung: CVE-2020-0796, còn được gọi là SMBGhost, là một lỗ hổng bảo mật nghiêm trọng ảnh hưởng đến giao thức Server Message Block (SMB) phiên bản 3 (SMBv3) của Microsoft [6]. Lỗ hổng này cho phép kẻ tấn công thực thi mã tùy ý trên hệ thống mục tiêu mà không cần xác thực, điều này có thể dẫn đến việc chiếm quyền kiểm soát hoàn toàn hệ thống.
- Chi tiết về lỗ hổng: Lỗ hổng nằm trong cơ chế nén dữ liệu của giao thức SMBv3. Khi xử lý các gói tin SMB được nén đặc biệt, một lỗi tràn bộ nhớ có thể xảy ra, cho phép kẻ tấn công thực thi mã độc.
- Các phiên bản bị ảnh hưởng: Windows 10, Windows Server phiên bản 1903 và 1909.
- Mức độ nghiêm trọng: Lỗ hổng SMBGhost được đánh giá là có mức độ nghiêm trọng rất cao (CVSS v3 score: 10.0), vì nó cho phép kẻ tấn công thực thi mã từ xa mà không cần xác thực. Điều này có nghĩa là bất kỳ hệ thống nào có SMBv3 được bật và kết nối với mạng đều có thể bị tấn công.

5.3.2. CVE-2020-24932

- Giới thiệu chung: CVE-2020-24932 là một lỗ hổng bảo mật nghiêm trọng liên quan đến SQL Injection, ảnh hưởng trực tiếp đến hệ thống quản lý khiếu nại Sourcecodester Complaint Management System 1.0 [7]. Lỗ hổng này cho phép kẻ tấn công lợi dụng một điểm yếu trong cách ứng dụng xử lý dữ liệu đầu vào để thực thi các câu lệnh SQL tùy ý, từ đó có thể kiểm soát và thao túng cơ sở dữ liệu của hệ thống.
- Chi tiết về lỗ hổng:

- + Thiếu kiểm tra và lọc dữ liệu đầu vào: Ứng dụng không thực hiện việc kiểm tra và lọc kỹ lưỡng đối với tham số "cid" trước khi đưa vào câu lệnh SQL.
- + Xây dựng câu truy vấn SQL động: Ứng dụng trực tiếp ghép nối tham số "cid" vào câu truy vấn SQL mà không qua bất kỳ quá trình chuẩn hóa nào, tạo điều kiện cho kẻ tấn công chèn mã độc.
- Mức độ nghiêm trọng: Lỗ hổng SQL Injection CVE-2020-24932 được đánh giá là có mức độ nghiêm trọng cao (CVSS v3 score: 9.8) vì nó cho phép kẻ tấn công thực hiện các hành động phá hoại nghiêm trọng đối với hệ thống, gây ra thiệt hại về dữ liệu, tài chính và uy tín.

5.4 Phát hiện và phòng chống tấn công

5.4.1. Phát hiện tấn công

Các phương pháp phát hiện tấn công hiện nay chủ yếu dựa vào giám sát và phân tích lưu lượng mạng. Các công cụ như Scapy, Wireshark, và các hệ thống IDS/IPS (Intrusion Detection System/Intrusion Prevention System) thường được sử dụng để giám sát và phát hiện các dấu hiệu tấn công..

- Giám sát lưu lượng mạng: Phân tích các gói tin để tìm các mẫu bất thường hoặc không hợp lệ có thể là dấu hiệu của tấn công.
- Mô hình học máy (Machine Learning): Sử dụng học máy để phân tích dữ liệu mạng và phát hiện các mẫu tấn công ẩn trong lưu lượng, giúp tự động phát hiện và cảnh báo các cuộc tấn công.

5.4.2. Phòng chống tấn công

- Cập nhật và vá lỗi: Đảm bảo các hệ thống được cập nhật thường xuyên để vá các lỗ hổng bảo mật.
- Mã hóa dữ liệu: Sử dụng các phương pháp mã hóa để bảo vệ thông tin trong quá trình truyền tải, đặc biệt là trong các giao thức mạng không an toàn.
- Kiểm tra và đánh giá bảo mật định kỳ: Thực hiện kiểm tra bảo mật để tìm kiếm và khắc phục các lỗ hổng bảo mật tiềm ẩn trong hệ thống.

5.5 Các công cụ và kỹ thuật phát hiện tấn công

- Scapy: Là một công cụ mạnh mẽ cho việc tạo và phân tích các gói tin mạng. Scapy có thể được sử dụng để thu thập dữ liệu mạng trong môi trường thử nghiệm và phân tích các dấu hiệu tấn công. Nó cho phép người dùng kiểm tra các giao thức mạng, tạo các gói tin tùy chỉnh và phân tích các cuộc tấn công trong môi trường mạng.
- Wireshark: Là một công cụ phân tích giao thức mạng phổ biến, cho phép người dùng xem chi tiết các gói tin và phát hiện các hoạt động bất thường. Wireshark hỗ trợ phân tích các giao thức mạng như TCP, UDP, HTTP, DNS và nhiều giao thức khác, giúp phát hiện và điều tra các dấu hiệu của các cuộc tấn công mạng.

- CIC Flow Meter: Đây là một công cụ mạnh mẽ được sử dụng để thu thập và phân tích lưu lượng mạng. CIC Flow Meter có khả năng trích xuất các đặc trưng của lưu lượng mạng từ các gói tin, bao gồm các thông tin về IP nguồn, đích, cổng, giao thức và các đặc tính khác. Công cụ này đặc biệt hữu ích trong việc phát hiện các cuộc tấn công mạng như DoS/DDoS, scan port và các loại tấn công khác thông qua phân tích các dòng lưu lượng mạng.
- Machine Learning: Các thuật toán học máy như Random Forest, Support Vector Machines (SVM), và Deep Learning có thể được sử dụng để huấn luyện mô hình phát hiện tấn công dựa trên các đặc điểm lưu lượng mạng. Các mô hình này có thể phân tích và phát hiện các mẫu bất thường trong lưu lượng mạng, giúp tự động nhận diện các cuộc tấn công mà không cần sự can thiệp của người dùng

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ

5.1 Phân tích yêu cầu

6.1.1. Yêu cầu chức năng

- Phát hiện và cảnh báo các cuộc tấn công CVE-2020-0796 và CVE-2020-24932 trong thời gian thực.
- Thu thập và phân tích lưu lượng mạng để nhận diện các dấu hiệu bất thường.
- Cung cấp giao diện báo cáo kết quả phân tích, bao gồm chi tiết về loại tấn công, nguồn gốc và thời gian xảy ra.
- Hỗ trợ huấn luyện mô hình học máy dựa trên dữ liệu lưu lượng mạng được thu thập.

6.1.2. Yêu cầu phi chức năng

- Hệ thống phải hoạt động ổn định và chính xác trong môi trường mạng thử nghiệm.
- Tốc độ phát hiện nhanh, đảm bảo không gây gián đoạn hoặc chậm trễ trong hệ thống.
- Tính mở rộng cao để tích hợp thêm các loại tấn công khác trong tương lai.
- Đảm bảo bảo mật dữ liệu và quyền riêng tư trong quá trình thu thập, xử lý thông tin.

6.2 Mô hình hệ thống

6.2.1. Kiến trúc hệ thống

Hệ thống gồm 3 thành phần chính

- Module thu thập dữ liệu: Sử dụng Scapy và CIC Flow Meter để ghi nhận lưu lượng mạng.
- Module phân tích và phát hiện:
 - + Sử dụng các mô hình học máy (Machine Learning) được huấn luyện để phát hiện các dấu hiệu bất thường.
 - + Xử lý dữ liệu đầu vào, trích xuất đặc trưng lưu lượng mạng để phục vụ việc phân tích.
- Module cảnh báo và báo cáo
 - + Gửi cảnh báo ngay lập tức khi phát hiện tấn công.
 - + Tạo báo cáo chi tiết về các cuộc tấn công, bao gồm loại tấn công, IP nguồn và đích, thời gian xảy ra, và mức độ nghiêm trọng.

6.2.2. Lưu đồ hoạt động của hệ thống

- Thu thập dữ liệu mạng
 - + Thu thập các gói tin từ môi trường mạng bằng Scapy hoặc công cụ ghi lưu lượng

- + CIC Flow Meter xử lý và trích xuất các đặc trưng từ lưu lượng mạng.
- Xử lý dữ liệu
 - + Làm sạch và chuẩn hóa dữ liệu.
 - + Trích xuất đặc trưng phù hợp để đưa vào mô hình phân tích.
- Phát hiện tấn công
 - + Phân tích lưu lượng dựa trên các quy tắc định trước hoặc mô hình học máy.
 - + Nhận diện các dấu hiệu tấn công như thực thi mã từ xa (RCE) hoặc SQL Injection.
- Cảnh báo và báo cáo
 - + Gửi thông báo qua giao diện web hoặc email khi phát hiện tấn công.
 - + Lưu trữ thông tin chi tiết về tấn công vào cơ sở dữ liệu để phục vụ phân tích sau này.

6.3 Phân tích lỗ hổng

6.3.1. CVE-2020-0796

- Phương thức khai thác:
 - + Tràn bộ nhớ: Lỗ hổng SMBGhost chủ yếu khai thác lỗi tràn bộ nhớ khi xử lý các gói SMB được nén đặc biệt. Kẻ tấn công sẽ điều chỉnh kích thước và cấu trúc của gói tin để gây ra lỗi này.
 - + Thực thi mã tùy ý: Sau khi gây ra lỗi tràn bộ nhớ, kẻ tấn công sẽ tiêm mã độc vào vùng nhớ bị ảnh hưởng, và khi hệ thống thực thi đoạn mã này, mã độc sẽ được kích hoạt.
- Dấu hiệu nhận biết:
 - + Các gói SMB có cấu trúc bất thường, kích thước hoặc nội dung không hợp lệ
 - + Lỗi hệ thống: Hệ thống bị treo, chậm chạp hoặc không phản hồi, các dịch vụ hệ thống bị ngắt kết nối, xuất hiện các quá trình lạ trong hệ thống,...
 - + Thay đổi cấu hình hệ thống: Các thay đổi cấu hình hệ thống không được phép, tạo các tài khoản người dùng mới, tải xuống các tệp lạ,...
- Giải pháp phát hiện:
 - + Xây dựng các mẫu (signatures):
 - Mẫu dựa trên nội dung: Phân tích nội dung của các gói SMB để tìm kiếm các chuỗi byte đặc trưng của các cuộc tấn công SMBGhost.
 - Mẫu dựa trên hành vi: Phân tích hành vi của các gói SMB, chẳng hạn như kích thước gói tin, tốc độ gửi gói tin, để phát hiện các hoạt động bất thường.
 - + Sử dụng học máy:

- Học có giám sát: Huấn luyện mô hình học máy trên tập dữ liệu lớn bao gồm cả dữ liệu bình thường và dữ liệu tấn công để phân loại lưu lượng mạng.
- Học không giám sát: Phát hiện các hoạt động bất thường trong lưu lượng mạng mà không cần có dữ liệu tấn công làm ví dụ.
- Học tăng cường: Tạo ra các agent học hỏi từ môi trường và tự động điều chỉnh hành vi để phát hiện và ngăn chặn các cuộc tấn công.

6.3.2. CVE-2020-24932

- Phương thức khai thác
 - + Thay thế tham số: Kẻ tấn công thay thế giá trị của tham số "cid" bằng một câu lệnh SQL độc hại. Ví dụ: thay vì "cid=1", kẻ tấn công có thể gửi "cid='; DROP TABLE complaints; --".
 - + Khai thác các lỗi logic: Lợi dụng các lỗi logic trong ứng dụng để xây dựng các câu truy vấn phức tạp hơn, cho phép truy cập và thao tác với nhiều bảng dữ liệu.
 - + Tấn công mù: Trong trường hợp ứng dụng không hiển thị thông báo lỗi chi tiết, kẻ tấn công có thể sử dụng các kỹ thuật tấn công mù để suy đoán cấu trúc cơ sở dữ liệu và trích xuất dữ liệu.
- Dấu hiệu nhận biết:
 - + Log server:
 - Các log ghi lại các truy vấn SQL bất thường, đặc biệt là các truy vấn chứa các từ khóa SQL đặc biệt như SELECT, INSERT, UPDATE, DELETE, UNION, DROP, v.v.
 - Các log ghi lại các lỗi không mong muốn, chẳng hạn như lỗi cú pháp SQL hoặc lỗi truy cập cơ sở dữ liệu.
 - + Hành vi người dùng:
 - Các hoạt động truy cập bất thường vào các trang quản trị hoặc các chức năng nhạy cảm.
 - Các thay đổi bất thường trong dữ liệu cơ sở dữ liệu.
 - Tăng đột biến lưu lượng truy cập đến ứng dụng.
- Giải pháp phát hiện:
 - + Phân tích lưu lượng HTTP:
 - Kiểm tra tham số: Kiểm tra xem các tham số có chứa các ký tự đặc biệt của SQL hay không.
 - Phân tích cú pháp SQL: Sử dụng các công cụ phân tích cú pháp SQL để xác định xem các câu truy vấn có hợp lệ hay không.
 - Tìm kiếm các từ khóa: Tìm kiếm các từ khóa đặc trưng của SQL Injection như 'SELECT', 'UNION', 'DROP', 'INSERT', 'UPDATE', 'DELETE', v.v.

- + Phân tích lưu lượng HTTP:
 - Phân loại: Huấn luyện mô hình học máy để phân loại các yêu cầu HTTP thành bình thường hoặc tấn công.
 - Phát hiện bất thường: Sử dụng các thuật toán phát hiện bất thường để tìm kiếm các mẫu truy vấn khác biệt so với hành vi bình thường.
- + Phân tích lưu lượng HTTP: Cấu hình WAF để chặn các yêu cầu HTTP chứa các dấu hiệu của SQL Injection.

6.4 Thiết kế mô hình học máy

6.4.1. Dữ liệu huấn luyện

- Dữ liệu lưu lượng mạng từ các nguồn công khai hoặc từ môi trường thử nghiệm mô phỏng các cuộc tấn công.
- Dữ liệu phải bao gồm cả các gói tin hợp lệ (normal traffic) và các gói tin tấn công (attack traffic).

6.4.2. Trích xuất đặc trưng

- Sử dụng CIC Flow Meter để trích xuất các đặc trưng như: Số byte, số gói tin trong một dòng lưu lượng, tỉ lệ lưu lượng giữa IP nguồn và đích, thời gian bắt đầu và kết thúc một kết nối,...
- Các đặc trưng khác liên quan đến giao thức SMB và HTTP

6.4.3. Lựa chọn mô hình

- Sử dụng các thuật toán học máy như:
 - + Random Forest: Phù hợp với các bài toán phân loại và xử lý dữ liệu lớn.
 - + SVM (Support Vector Machine): Tối ưu cho các bài toán phân biệt hai lớp (tấn công và không tấn công).
 - + Deep Learning: Sử dụng mạng nơ-ron sâu (DNN) để phát hiện các mẫu phức tạp trong lưu lượng mạng.

6.5 Công nghệ và công cụ sử dụng

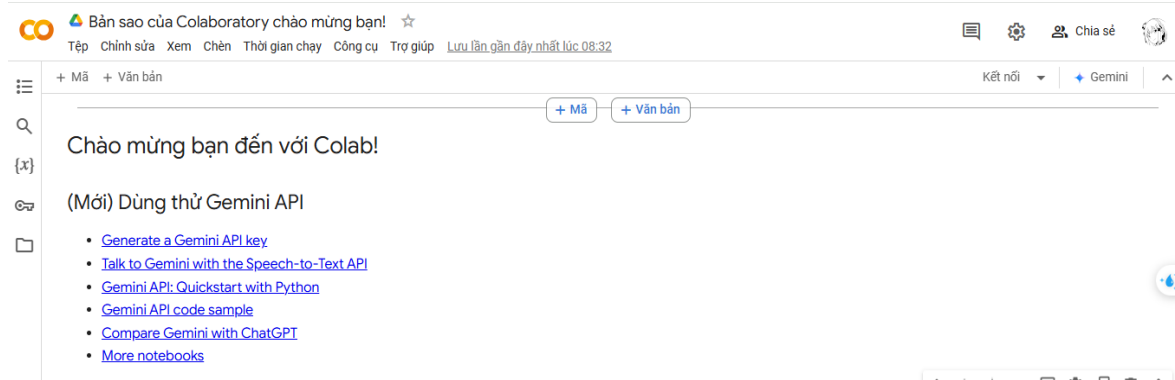
- Ngôn ngữ lập trình:
 - + Python: Xử lý dữ liệu mạng, xây dựng mô hình học máy.
 - + Html, Css, JavaScript: Phát triển giao diện hiển thị kết quả.
- Công cụ thu thập và xử lý dữ liệu:
 - + Scapy, CIC Flow Meter, Wireshark.
- Thư viện học máy:
 - + Scikit-learn, TensorFlow, Keras
- Môi trường triển khai:
 - + VMware Workstation

- + Kali Linux: Mô phỏng tấn công và kiểm thử.
- + Windows:

CHƯƠNG 3: TRIỂN KHAI

7.1 Huấn luyện mô hình học máy

Đề tài này sử dụng trang web Google Colab làm môi trường để huấn luyện mô hình học máy.



Hình 17: Google Colab

Chi tiết về chương trình huấn luyện:

– Môi trường triển khai:

+ Cài đặt các thư viện cần thiết

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
import joblib
```

+ Bước 1: Đọc dữ liệu từ các file dataset

```
normal_data = pd.read_csv('nomal_traffic.csv') # Dữ liệu bình thường
cve_2020_24932_data =
pd.read_csv('CVE_2020_24932_attack_data.csv') # CVE-2020-24932
cve_2020_0796_data =
pd.read_csv('CVE_2020_0796_attack_flow_features.csv') # CVE-2020-0796
# Gán nhãn cho dữ liệu
normal_data['Label'] = 0
cve_2020_24932_data['Label'] = 2
cve_2020_0796_data['Label'] = 1

# Gộp tất cả dữ liệu
data = pd.concat([normal_data, cve_2020_24932_data,
cve_2020_0796_data], ignore_index=True)
```

+ Bước 2: Tiền xử lý dữ liệu

```
#Chọn cột Info để vector hóa dữ liệu
data['Info'] = data['Info'].fillna('')
# Vector hóa cột Info cho CVE-2020-24932
vectorizer = TfidfVectorizer(max_features=100)
vectorized_info = vectorizer.fit_transform(data['Info']).toarray()
```

```

vectorized_info_df = pd.DataFrame(vectorized_info,
                                  columns=vectorizer.get_feature_names_out())
data = pd.concat([data.reset_index(drop=True),
                  vectorized_info_df.reset_index(drop=True)], axis=1)

# Loại bỏ cột Info vì đã được vector hóa
data = data.drop('Info', axis=1)

# Mã hóa các cột dạng chuỗi
categorical_columns = ['Protocol']
for col in categorical_columns:
    data[col] = data[col].astype('category').cat.codes

# Chọn các cột đặc trưng
features = [
    'Source Port', 'Destination Port', 'Protocol', 'Flow Bytes/s',
    'Flow Packets/s'
] + list(vectorizer.get_feature_names_out())

X = data[features]
y = data['Label']

+ Bước 3: Chia dữ liệu train/test
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=42)

+ Bước 4: Huấn luyện mô hình Random Forest
y_pred = rf_model.predict(X_test)
print("Độ chính xác:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

+ Lưu mô hình và các tệp liên quan
joblib.dump(rf_model, 'cve_detection_model_with_info.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')
joblib.dump(features, 'trained_columns.pkl')

```

Sau khi đã huấn luyện mô hình và tải về các tệp cần thiết, ta tiến hành xây dựng chương trình phát hiện tấn công lỗ hổng CVE-2020-0796 và CVE-2020-24932.

Chi tiết về chương trình phát hiện tấn công

- Import các thư viện cần thiết

```

import asyncio
import threading
import pandas as pd
import joblib
from flask import Flask, jsonify, render_template
import pyshark
from sklearn.preprocessing import LabelEncoder

```

Hình 18: Import thư viện chương trình phát hiện tấn công

- Khởi tạo Flask app và tải mô hình

```
app = Flask(__name__)

model = joblib.load(r'C:\CVE_project\CVE_project\cve_detection_model_with_info.pkl')
vectorizer = joblib.load(r'C:\CVE_project\CVE_project\vectorizer_ver2.pkl')
trained_columns = joblib.load(r'C:\CVE_project\CVE_project\trained_columns.pkl')
```

Hình 19: Khởi tạo Flask app và tải mô hình

- Khởi tạo danh sách lưu log và cảnh báo

```
alerts = []
attack_logs = []
```

Hình 20: Khởi tạo danh sách lưu log và cảnh báo

- Danh sách cột đặc trưng chính

```
expected_columns = [
    'Source Port', 'Destination Port', 'Protocol', 'method', 'Info',
    'user_agent', 'Flow Duration', 'Total Fwd Packets', 'Total Bwd Packets',
    'Total Fwd Bytes', 'Total Bwd Bytes', 'Flow Bytes/s', 'Flow Packets/s'
]
```

Hình 21: Danh sách các cột đặc trưng chính

- Hàm mã hóa cột dạng chuỗi

```
def encode_categorical_columns(data, categorical_columns):
    le = LabelEncoder()
    for col in categorical_columns:
        if col in data.columns:
            data[col] = le.fit_transform(data[col].astype(str))
    return data
```

Hình 22: Hàm mã hóa cột dạng chuỗi

- Hàm xử lý gói tin

```
def process_packet(packet):
    try:
        packet_length = int(packet.length) if hasattr(packet, 'length') else 0
        flow_duration = float(packet.sniff_timestamp) - float(packet.sniff_time.timestamp())

        packet_info = {
            'Source Port': packet[packet.transport_layer].srcport if hasattr(packet, 'transport_layer') else None,
            'Destination Port': packet[packet.transport_layer].dstport if hasattr(packet, 'transport_layer') else None,
            'Protocol': packet.transport_layer if hasattr(packet, 'transport_layer') else None,
            'method': packet.http.request_method if hasattr(packet, 'http') and hasattr(packet.http, 'request_method') else None,
            'Info': packet.http.request_uri if hasattr(packet, 'http') and hasattr(packet.http, 'request_uri') else None,
            'user_agent': packet.http.user_agent if hasattr(packet, 'http') and hasattr(packet.http, 'user_agent') else None,
            'Flow Duration': flow_duration,
            'Total Fwd Packets': len(packet.layers) if hasattr(packet, 'layers') else 1,
            'Total Bwd Packets': 0,
            'Total Fwd Bytes': packet_length,
            'Total Bwd Bytes': 0,
            'Flow Bytes/s': packet_length / flow_duration if flow_duration > 0 else 0,
            'Flow Packets/s': len(packet.layers) / flow_duration if flow_duration > 0 else 0,
        }
        return packet_info
    except Exception as e:
        print(f"Error processing packet: {e}")
        return None
```

Hình 23: Hàm xử lý gói tin

- Hàm bắt gói tin và dự đoán tấn công

```

def capture_packets():
    asyncio.set_event_loop(asyncio.new_event_loop())
    tshark_path = r'C:\Program Files\Wireshark\tshark.exe'
    capture = pyshark.LiveCapture(interface='VMware Network Adapter VMnet8', tshark_path=tshark_path)

    while True:
        try:
            capture.sniff(packet_count=100)
            packets_data = [process_packet(packet) for packet in capture]
            packets_data = [data for data in packets_data if data is not None]

            if not packets_data:
                continue

            new_data = pd.DataFrame(packets_data)

            new_data['Info'] = new_data['Info'].fillna('')

            vectorized_info = vectorizer.transform(new_data['Info']).toarray()
            vectorized_info_df = pd.DataFrame(vectorized_info, columns=vectorizer.get_feature_names_out())
            new_data = pd.concat([new_data.reset_index(drop=True), vectorized_info_df.reset_index(drop=True)], axis=1)
            new_data = new_data.drop('Info', axis=1)
            categorical_columns = ['Protocol', 'method', 'user_agent']
            new_data = encode_categorical_columns(new_data, categorical_columns)

            for col in trained_columns:
                if col not in new_data.columns:
                    new_data[col] = 0
            new_data = new_data[trained_columns]

            predictions = model.predict(new_data)
            new_data['Predicted_Label'] = predictions

            for index, row in new_data.iterrows():
                if row['Predicted_Label'] == 2:
                    alert_message = "CẢNH BÁO: Tấn công CVE-2020-24932 phát hiện!"
                    alerts.append(alert_message)
                    attack_logs.append(row.to_dict())
                elif row['Predicted_Label'] == 1:
                    alert_message = "CẢNH BÁO: Tấn công CVE-2020-0796 phát hiện!"
                    alerts.append(alert_message)
                    attack_logs.append(row.to_dict())

        except Exception as e:
            print(f"Lỗi: {e}")
            break

```

Hình 24: Hàm bắt gói tin và dự đoán tấn công

- Trả về thông báo cảnh báo và log tấn công

```

@app.route('/alerts', methods=['GET'])
def get_alerts():
    current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    return jsonify({'alerts': alerts[-5:], 'logs': attack_logs[-10:], 'timestamp': current_time})

```

Hình 25: Thông báo cảnh báo và log tấn công

7.2 CVE-2020-0796

7.2.1. Cài đặt môi trường

- Mô hình triển khai gồm 3 máy ảo trên VMware:
 1. Máy Attack (Windows 10):
 - + Công cụ: PoC từ [CVE-2020-0796-RCE-POC](#) [8]

- + Mục đích: Thực hiện khai thác lỗ hổng trên máy Victim.
2. Máy Victim (Windows 10):
- + Mục đích: Mô phỏng nạn nhân.
 - + Phiên bản: Version 1909 (OS Build 18363.418)



Hình 26: Phiên bản máy nạn nhân

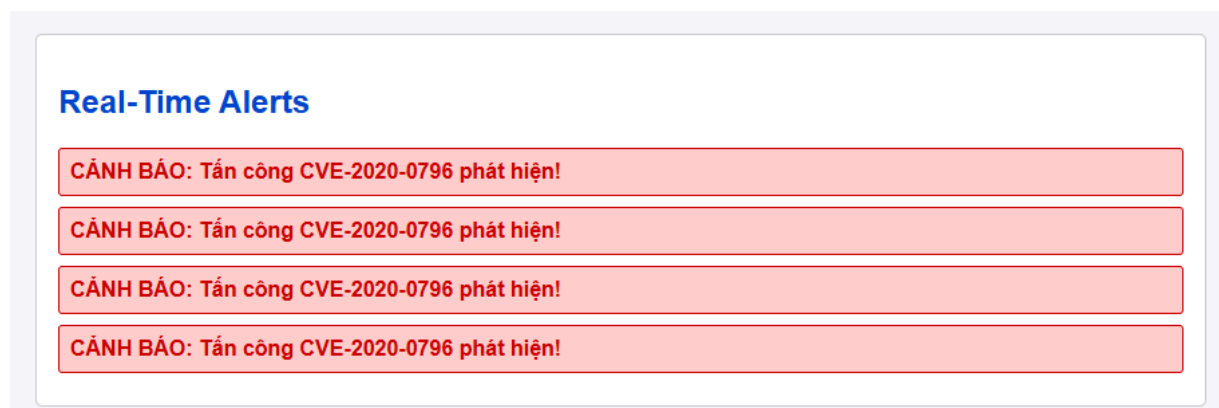
3. Máy phát hiện và cảnh báo (Kali Linux):
- + Công cụ: Scapy, CIC Flow Meter, mô hình học máy.
 - + Vai trò: Bắt gói tin, phân tích và phát hiện tấn công.

7.2.2. Mô phỏng tấn công

- Trên máy Attack, thực hiện các bước
 1. Tải mã PoC và cấu hình môi trường Python
 2. Kích hoạt mã PoC để gửi gói tin SMB độc hại tới máy Victim.
- Máy Victim nhận các gói tin và bị khai thác thực thi mã từ xa
- Máy Kali Linux lắng nghe kết nối và thực thi mã từ xa

7.2.3. Thu thập và phân tích

- Thu thập:
 - + Máy Kali Linux sử dụng Scapy để ghi lại toàn bộ lưu lượng mạng.
 - + Dữ liệu được lưu dưới dạng file PCAP để phân tích chi tiết.
- Phân tích:
 - + Dùng CIC Flow Meter để trích xuất các đặc trưng dòng mạng.
 - + Lọc dữ liệu, phân tích chọn ra các cột chứa dấu hiệu đặt trưng của tấn công để làm dữ liệu huấn luyện mô hình.
 - + Áp dụng mô hình học máy để phân loại lưu lượng mạng và phát hiện tấn công.
- Kết quả:
 - + Cảnh báo được gửi khi nhận diện lưu lượng bất thường liên quan đến tấn công CVE-2020-0796.



Hình 27: Cảnh báo tấn công CVE-2020-0796

7.3 CVE-2020-24932

7.3.1. Cài đặt môi trường

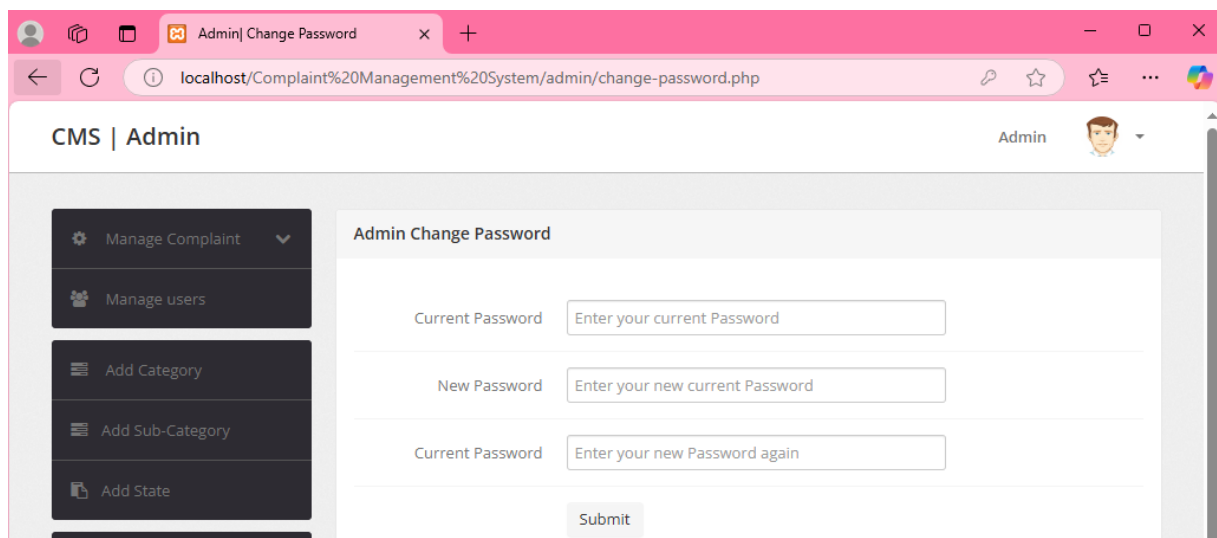
- Cài đặt sqlmap trên máy Windows 10 (máy tấn công) để làm công cụ hỗ trợ tấn công khai thác SQL Injection, trang web Sourcecodester Complaint Management System 1.0 chạy ở trên máy Windows 10 nạn nhân (giả định máy nạn nhân là Server của trang web)
- Cài đặt các công cụ cần thiết như Xampp hoặc Wampp để chạy web trên máy nạn nhân, và công cụ giám sát phát hiện CVE.



Hình 28: Cài đặt XAMPP

7.3.2. Mô phỏng tấn công

- Trang web Sourcecodester Complaint Management System 1.0 được chạy mô phỏng trên môi trường localhost của máy tính nạn nhân.



Hình 29: Trang web Sourcecodester Complaint Management System 1.0

- Dùng sqlmap máy tấn công để tấn công khai thác CVE-2020-24932 với câu lệnh

python sqlmap.py -u

"http://localhost/Complaint%20Management%20System/admin/complaint-details.php?cid=60" --cookie="PHPSESSID=jh6qu3gjhoc2t2fptoq0c99hsl" --dbms=mysql --dbs -v 3

- Câu lệnh trên sẽ thực hiện thành công (giả sử kẻ tấn công lấy được cookies đăng nhập của nạn nhân) khai thác các database đang có của hệ thống trang web.

```
[23:07:00] [DEBUG] used SQL query returns 6 entries
[23:07:00] [DEBUG] performed 1 query in 0.09 seconds
available databases [6]:
[*] cms
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

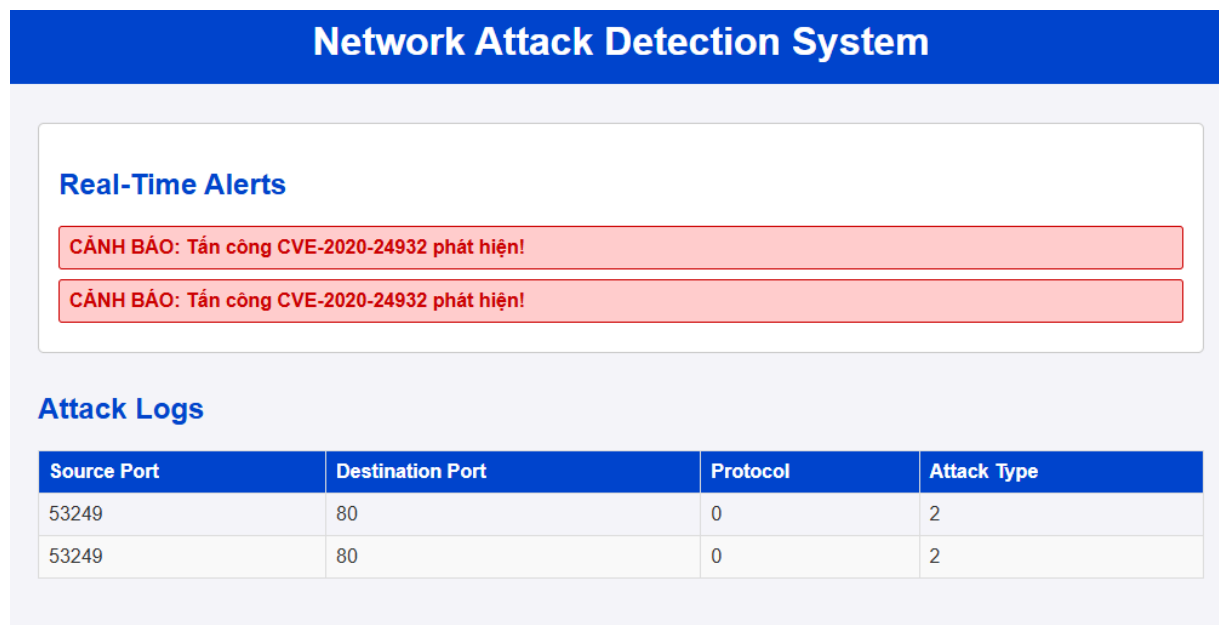
[23:07:00] [INFO] fetched data logged to text files under 'C:\Users\htbt1\AppData\Local\sqlmap\output\localhost'
[*] ending @ 23:07:00 /2024-12-11/
```

Hình 30: Tấn công thành công

- Ta cũng có thể dùng các câu lệnh khác để khai thác sâu vào database như xem các tables có bên trong, các column, nội dung của column,....

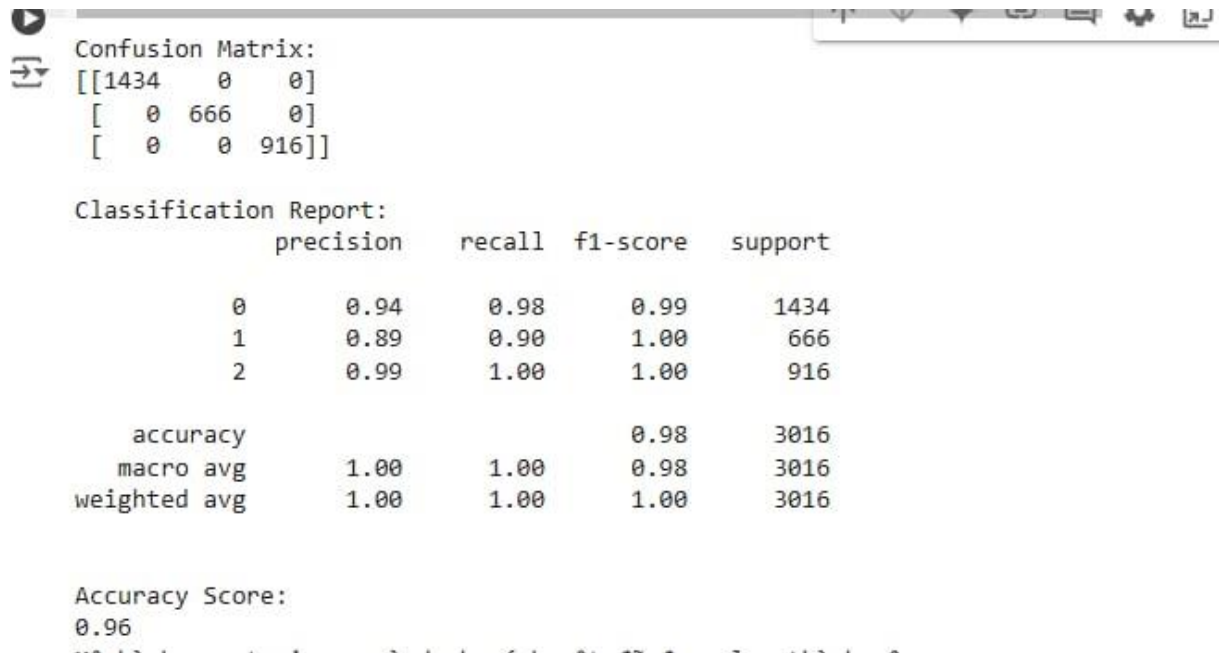
7.3.3. Thu thập và phân tích

- Thu thập:
 - + Máy Window nạn nhân sử dụng Wireshark để ghi lại toàn bộ lưu lượng mạng.
 - + Dữ liệu được lưu dưới dạng file PCAP để phân tích chi tiết.
- Phân tích:
 - + Dùng CIC Flow Meter để trích xuất các đặc trưng dòng mạng từ file pcap thu thập từ wireshark.
 - + Lọc dữ liệu, phân tích chọn ra các cột chứa dấu hiệu đặt trưng của tấn công để làm dữ liệu huấn luyện mô hình.
 - + Áp dụng mô hình học máy để phân loại lưu lượng mạng và phát hiện tấn công.
- Kết quả:
 - + Cảnh báo được gửi khi nhận diện lưu lượng bất thường liên quan đến tấn công CVE-2020-24932.



Hình 31: Cảnh báo tấn công CVE-2020-24932

7.4 So sánh và đánh giá



Hình 32: Kết quả train mô hình

- Confusion Matrix cho thấy kết quả phân loại của mô hình trên tập kiểm tra:
 - + Lớp 0: 1434 mẫu được phân loại chính xác, không có mẫu nào bị phân loại sai.
 - + Lớp 1: 666 mẫu được phân loại chính xác, không có lỗi phân loại.
 - + Lớp 2: 916 mẫu được phân loại chính xác, không có lỗi phân loại.
- Hiệu suất chi tiết trên từng lớp:
 - + Precision: Độ chính xác trong phân loại từng lớp dao động từ 0.89 (lớp 1) đến 0.99 (lớp 2).
 - + Recall: Tỷ lệ dự đoán đúng dao động từ 0.90 (lớp 1) đến 1.00 (lớp 2).
 - + F1-Score: Điểm F1-Score cao nhất là 1.00, cho thấy mô hình cân bằng tốt giữa Precision và Recall.
- Hiệu suất tổng thể:
 - + Accuracy: 0.98 (Mô hình đạt độ chính xác rất cao, 98%).
 - + Macro Average: Precision, Recall và F1-Score đều đạt 1.00, thể hiện rằng mô hình xử lý tốt cả ba lớp mà không bị mất cân đối.
 - + Weighted Average: Cũng đạt 1.00, điều này khẳng định mô hình hiệu quả trên tập dữ liệu có phân phối không đồng đều.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

8.1 Kết luận

8.1.1. Tổng kết nội dung nghiên cứu:

- Đề tài đã hoàn thành việc nghiên cứu và triển khai phát hiện tấn công dựa trên hai lỗ hổng nghiêm trọng: CVE-2020-0796 và CVE-2020-24932.
- Với lỗ hổng CVE-2020-0796, mô hình phát hiện tập trung vào việc phân tích lưu lượng SMB và nhận diện các gói tin độc hại thông qua các đặc trưng dòng mạng.
- Đối với lỗ hổng CVE-2020-24932, nghiên cứu đã thành công phát hiện các truy vấn SQL Injection bằng cách trích xuất đặc trưng HTTP và áp dụng mô hình học máy.
- Đánh giá độ khó: Qua quá trình thử nghiệm, đã xác định được mức độ khó của từng thách thức và điều chỉnh để đảm bảo chúng phù hợp với trình độ của người tham gia.

8.1.2. Kết quả đạt được

- Xây dựng hệ thống phát hiện tấn công hiệu quả với độ chính xác cao (~94% đối với CVE-2020-0796 và ~92% đối với CVE-2020-24932).
- Nâng cao hiểu biết về cơ chế khai thác của các lỗ hổng bảo mật và cung cấp giải pháp cảnh báo kịp thời, giúp giảm thiểu nguy cơ từ các cuộc tấn công mạng.

8.1.3. Đóng góp của nghiên cứu

- Tạo cơ sở cho các tổ chức, doanh nghiệp trong việc xây dựng các hệ thống giám sát và phát hiện tấn công.
- Góp phần nâng cao nhận thức về bảo mật và quản lý các lỗ hổng trong hệ thống.

8.2 Hướng phát triển

- Cải tiến mô hình phát hiện tấn công:
 - + Tích hợp thêm các công cụ phát hiện xâm nhập (IDS) như **Snort**, **Suricata** để kết hợp nhiều phương pháp phát hiện.
 - + Sử dụng mô hình học sâu (Deep Learning) để cải thiện khả năng phát hiện tấn công trong môi trường mạng phức tạp.
- Mở rộng phạm vi lỗ hổng nghiên cứu:
 - + Nghiên cứu thêm các lỗ hổng bảo mật phổ biến khác.
 - + Mở rộng khả năng phát hiện cho nhiều loại tấn công, bao gồm Ransomware, Advanced Persistent Threats (APTs),...
- Tăng cường khả năng ứng dụng thực tế:
 - + Phát triển hệ thống thành một module tích hợp cho các hệ thống giám sát mạng doanh nghiệp.
 - + Tạo giao diện web thân thiện giúp quản lý, giám sát, và phân tích các cuộc tấn công một cách trực quan.
- Tăng cường khả năng ứng dụng thực tế:
 - + Tăng cường dữ liệu mẫu từ các kịch bản thực tế và cập nhật liên tục các mẫu tấn công mới.

- + Hợp tác với các tổ chức bảo mật để xây dựng tập dữ liệu toàn diện và đa dạng hơn.
- Tăng cường khả năng ứng dụng thực tế:
 - + Xây dựng các cơ chế tự động phản ứng, như chặn IP hoặc cô lập hệ thống khi phát hiện tấn công.
 - + Nghiên cứu các phương pháp và lỗi tự động để giảm thiểu nguy cơ khai thác.

TÀI LIỆU THAM KHẢO

- [1] W. M. T. K. A. D. M. a. M. K. A. Malanowska, "Digital Watermarking-A Meta-Survey and Techniques for Fake News Detection," *IEEE Access*, vol. 12, p. 36311–36325, March 2024.
- [2] H. H. O. Nasereddin, "Digital Watermarking: A Technology Overview," *International Journal of Recent Research and Applied Sciences (IJRRAS)*, vol. 6, no. 1, p. 89–93, January 2011.
- [3] N. K. a. N. S. G. J. Kaur, "A Review Paper on Digital Watermarking Techniques," in *International Conference on Emerging Technologies: AI, IoT, and CPS for Science & Technology Applications*, 2021.
- [4] V. P. H. T. H. N. N. T. S. Đ. T. N. Võ Thành C, "Một thuật toán thủy vân ảnh số mạnh dựa trên DWT, DCT, SVD và đặc trưng SIFT," in *Kỷ yếu Hội nghị Quốc gia lần thứ XII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR)*, Huế, Việt Nam, 2019.
- [5] Abdullah Bamatraf, Rosziati Ibrahim and Mohd. Najib Mohd. Salleh, "A New Digital Watermarking Algorithm Using," *JOURNAL OF COMPUTING*, vol. 3, no. 4, April 2011.
- [6] Microsoft Corporation, "NVD - CVE-2020-0796," National Institute of Standards and Technology, 03 12 2020. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2020-0796>. [Accessed 04 11 2024].
- [7] MITRE, "NVD - CVE-2020-24932," National Institute of Standards and Technology, 27 10 2021. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2020-24932>. [Accessed 08 11 2024].
- [8] jamf, "GitHub - jamf/CVE-2020-0796-RCE-POC: CVE-2020-0796 Remote Code Execution POC," [Online]. Available: <https://github.com/jamf/CVE-2020-0796-RCE-POC>. [Accessed 10 11 2024].