**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC VÀ KĨ THUẬT THÔNG TIN**

----------------------------------------------------



**Nhập môn bảo đảm và an ninh thông tin**

**ĐỒ ÁN CUỐI KỲ**

**Đề tài:**

**Tạo bộ dữ liệu tấn công web defaced**

**GVHD:** Nguyễn Tấn Cầm

A person in a white shirt

AI-generated content may be incorrect.

**Sinh viên thực hiện (Nhóm 16):**

Dương Vi Khắc Nhật: 22521012

A person with her arms crossed

AI-generated content may be incorrect.

Từ Lý Mai: 23520912

A person in a blue and white shirt

AI-generated content may be incorrect.

Lê Trần Đức Thiện: 23521478

**🙡🙢** Tp. Hồ Chí Minh, 05/2025 **🙡🙢**

LỜI MỞ ĐẦU

Lời đầu tiên, chúng tôi xin dành lời cảm ơn chân thành sâu sắc nhất đến với giáo viên hướng dẫn, là TS. Nguyễn Tấn Cầm vì đã giúp đỡ chúng tôi trong việc hoàn thành bài báo cáo này, giải đáp các vấn đề mà chúng tôi thắc mắc và hướng dẫn trình bày bài theo đúng hướng.

Xin cảm ơn đến các thành viên trong nhóm đã góp thời gian và công sức để cùng nhau hoàn thành dự án này. Chúng ta đã cùng nhau trải qua thời gian làm việc nhóm vui vẻ và hợp tác để hướng đến mục đích chung.

Bài báo cáo này có nội dung về việc “Tạo bộ dữ liệu tấn công web defaced”, giúp chúng tôi có cái nhìn rõ hơn về đề tài và giá trị mà SQLite được ứng dụng trong thực tiễn. Ngoài ra, giúp chúng tôi củng cố kiến thức và học hỏi thêm nhiều bài học mới.

Sau đây, nhóm chúng tôi sẽ trình bày về đề tài “Tạo bộ dữ liệu tấn công web defaced” qua các chương sau:

* Chương 1: Tổng quan về web defaced
* Chương 2: Thu thập dữ liệu
* Chương 3: Phân tích, xử lý và đánh giá dữ liệu
* Chương 4: Xây dựng và huấn luyện mô hình
* Chương 5: Tổng kết và hướng phát triển

MỤC LỤC

[LỜI MỞ ĐẦU ii](#_Toc200614285)

[MỤC LỤC iii](#_Toc200614286)

[DANH MỤC HÌNH ẢNH vi](#_Toc200614287)

[DANH MỤC BẢNG vii](#_Toc200614288)

[Bảng phân công và đánh giá hiệu suất viii](#_Toc200614289)

[CHƯƠNG 1: TỔNG QUAN VỀ TẤN CÔNG WEB DEFACED 1](#_Toc200614290)

[1.1 Khái niệm về tấn công deface 1](#_Toc200614291)

[1.2 Nguyên nhân và kỹ thuật tấn công defacement 1](#_Toc200614292)

[1.2.1 Nguyên nhân dẫn đến tấn công defacement 1](#_Toc200614293)

[1.2.1.1 Lỗ hổng kỹ thuật: 1](#_Toc200614294)

[1.2.1.2 Cấu hình sai 1](#_Toc200614295)

[1.2.1.3 Yếu tố con người 2](#_Toc200614296)

[1.2.1.4 Thiếu giám sát và bảo trì 2](#_Toc200614297)

[1.2.2 Một số kỹ thuật tấn công defacement 2](#_Toc200614298)

[1.2.2.1. SQL Injection 2](#_Toc200614299)

[1.2.2.2 Cross-Site Scripting (XSS) 2](#_Toc200614300)

[1.2.2.3 Remote File Inclusion (RFI) và Local File Inclusion (LFI) 3](#_Toc200614301)

[1.2.2.4 Brute-force tài khoản quản trị 3](#_Toc200614302)

[1.2.2.5 Upload file độc hại 3](#_Toc200614303)

[1.2.3 Phân tích kỹ thuật và công cụ hỗ trợ 3](#_Toc200614304)

[1.2.4 Ví dụ về các cuộc tấn công phá hoại trang web 3](#_Toc200614305)

[1.2.4.1. Tấn công phá hoại NHS 4](#_Toc200614306)

[1.2.4.2. Google.ro và PayPal.ro 4](#_Toc200614307)

[1.2.4.3. Cuộc tấn công mạng lớn nhất trong lịch sử Georgia 4](#_Toc200614308)

[1.3 Phân loại và đặc điểm nhận dạng defacement 4](#_Toc200614309)

[1.3.1 Phân loại 4](#_Toc200614310)

[1.3.1.1 Homepage Defacement 4](#_Toc200614311)

[1.3.1.2 Mass Defacement 5](#_Toc200614312)

[1.3.1.3 Redefacement 5](#_Toc200614313)

[1.3.1.4 IP Address Location Defacement 6](#_Toc200614314)

[1.3.1.5 Special Defacement 6](#_Toc200614315)

[1.3.2 Một số đặc điểm nhận dạng defacement 7](#_Toc200614316)

[1.3.2.1 Thay đổi giao diện bất thường 7](#_Toc200614317)

[1.3.2.2 Xuất hiện thông điệp hoặc chữ ký hacker 7](#_Toc200614318)

[1.3.2.3 Nội dung bị xóa hoặc thay thế 7](#_Toc200614319)

[1.3.2.4 Chuyển hướng người dùng đến trang không mong muốn 7](#_Toc200614320)

[1.3.2.5 Phát hiện mã độc hoặc đoạn mã lạ trong mã nguồn 8](#_Toc200614321)

[1.4 Mục tiêu của kẻ tấn công 8](#_Toc200614322)

[1.4.1 Tuyên truyền chính trị hoặc xã hội: 8](#_Toc200614323)

[1.4.2 Phá hoại uy tín hoặc gây rối: 8](#_Toc200614324)

[1.4.3 Chứng tỏ năng lực hoặc danh tiếng 8](#_Toc200614325)

[1.4.4 Cảnh báo lỗ hổng bảo mật 9](#_Toc200614326)

[1.4.5 Lợi ích tài chính 9](#_Toc200614327)

[1.5 Các công cụ theo dõi và thống kê (Zone-H) 9](#_Toc200614328)

[1.5.1 Khái niệm: 10](#_Toc200614329)

[1.5.2 Các tính năng của Zone-H 10](#_Toc200614330)

[1.5.2.1 Lưu trữ dữ liệu defacement: 10](#_Toc200614331)

[1.5.2.2 Thống kê và phân tích: 10](#_Toc200614332)

[1.5.2.3 Xếp hạng hacker. 10](#_Toc200614333)

[1.5.2.4 Công cụ tìm kiếm và phân loại 11](#_Toc200614334)

[1.5.2.5 Cộng đồng và báo cáo 11](#_Toc200614335)

[1.5.3 Cách sử dụng Zone-H 11](#_Toc200614336)

[1.5.4. Vai trò của Zone-H 11](#_Toc200614337)

[1.6 Phương pháp khai thác và thu thập dữ liệu 12](#_Toc200614338)

[1.6.1. Cào dữ liệu (Web Scraping) 12](#_Toc200614339)

[1.6.2. Dữ liệu mô phỏng (Simulated Data) 12](#_Toc200614340)

[1.6.3. Dữ liệu thực tế (Real-world Data) 12](#_Toc200614341)

[1.6.4. Công cụ giám sát và phát hiện 13](#_Toc200614342)

[1.6.5. Phân tích hình ảnh và nội dung 13](#_Toc200614343)

[1.6.6. Phân tích log và giám sát mạng 13](#_Toc200614344)

[Chương 2: Thu thập dữ liệu 14](#_Toc200614345)

[2.1. Thu thập các trang web nguyên bản và bị deface 14](#_Toc200614346)

[2.1.1. Các trang web nguyên bản 14](#_Toc200614347)

[2.1.2. Các trang web bị defaced 14](#_Toc200614348)

[2.1.2.1. Cách cào links đầu tiên 15](#_Toc200614349)

[2.1.2.2. Cách cào links thứ hai 20](#_Toc200614350)

[2.2. Thu thập hình ảnh của các trang web bị deface 29](#_Toc200614351)

[2.2.1. Thực hiện chương trình 29](#_Toc200614352)

[2.3 Thu thập dữ liệu HTML của các trang web bị defaced 35](#_Toc200614353)

[2.3.1 Mục tiêu 35](#_Toc200614354)

[2.3.2 Nguồn dữ liệu 35](#_Toc200614355)

[2.3.3 Kiến trúc và quy trình thu thập 36](#_Toc200614356)

[2.3.3.1. Truy cập vào từng website và thu thập nội dung HTML 36](#_Toc200614357)

[2.4 Thu thập dữ liệu bằng cách tự tạo Web và tự tấn công 39](#_Toc200614358)

[2.4.1. Mô phỏng tấn công 41](#_Toc200614359)

[2.4.1.1. Mô phỏng tấn công SQL Injection: 41](#_Toc200614360)

[2.4.1.2. Mô phỏng tấn công XSS (Cross-Site Scripting) 42](#_Toc200614361)

[2.4.1.3. Mô phỏng tấn công Upload Web Shell 44](#_Toc200614362)

[Chương 3: Phân tích, xử lý và đánh giá dữ liệu 47](#_Toc200614363)

[3.1. Dataset về hình ảnh 47](#_Toc200614364)

[3.1.1. Xử lý dữ liệu 47](#_Toc200614365)

[3.1.2. Làm sạch dữ liệu 53](#_Toc200614366)

[3.1.3. Đánh giá dữ liệu 53](#_Toc200614367)

[3.2. Dataset về HTML 56](#_Toc200614368)

[3.2.1. Xử lý dữ liệu 56](#_Toc200614369)

[3.2.2. Làm sạch dữ liệu 56](#_Toc200614370)

[3.2.3 Đánh giá dataset sau làm sạch 60](#_Toc200614371)

[Chương 4: Xây Dựng và Huấn Luyện Mô Hình 62](#_Toc200614372)

[4.1 Chuẩn bị dữ liệu 62](#_Toc200614373)

[4.2 Cài đặt môi trường 62](#_Toc200614374)

[4.3 Kiến trúc mô hình 62](#_Toc200614375)

[4.4 Huấn luyện mô hình 64](#_Toc200614376)

[4.5 Sử dụng mô hình đã huấn luyện 64](#_Toc200614377)

[4.6 Đánh giá mô hình 65](#_Toc200614378)

[Chương 5: Tổng Kết và Hướng Phát Triển 68](#_Toc200614379)

[5.1 Tổng Kết 68](#_Toc200614380)

[5.1.1 Kết Quả Thu Thập Dữ Liệu 68](#_Toc200614381)

[5.1.2 Kết Quả Xử Lý và Làm Sạch Dữ Liệu 69](#_Toc200614382)

[5.1.3 Kết Quả Huấn Luyện Mô Hình 69](#_Toc200614383)

[5.1.4 Đóng Góp của Đồ Án 70](#_Toc200614384)

[5.1.5 Hạn Chế 70](#_Toc200614385)

[5.2 Hướng Phát Triển 71](#_Toc200614386)

[5.2.1 Cải Thiện Bộ Dữ Liệu 71](#_Toc200614387)

[5.3 Kết Luận 73](#_Toc200614388)

[TÀI LIỆU THAM KHẢO 74](#_Toc200614389)

[TÀI NGUYÊN BÁO CÁO 76](#_Toc200614390)

DANH MỤC HÌNH ẢNH

[Hình 1.1. Giao diện trang web Zone-H 10](#_Toc198503768)

[Hình 2.1. Ví dụ về web daa của ĐH CNTT với giao diện trang web sạch 14](#_Toc198503769)

[Hình 2.2. Giao diện trang web Zone-H (nơi sẽ cào dữ liệu) 15](#_Toc198503770)

[Hình 2.3. File văn bản chứa các đường links đã cào 19](#_Toc198503771)

[Hình 2.5. Cài đặt thư viện resquest, beautifulsoup và colorama 20](#_Toc198503772)

[Hình 2.6. Chạy chương trình trong terminal 20](#_Toc198503773)

[Hình 2.7. Cách lấy thông tin cookie mà chương trình yêu cầu 20](#_Toc198503774)

[Hình 2.8. Chọn chế độ cào trang web 21](#_Toc198503775)

[Hình 2.9. Đoạn code cào links của web bị defaced (1) 22](#_Toc198503776)

[Hình 2.10. Đoạn code cào links của web bị defaced (2) 23](#_Toc198503777)

[Hình 2.11. Đoạn code cào links của web bị defaced (3) 24](#_Toc198503778)

[Hình 2.4. Folder image chứa các nội dung vừa trích xuất được (bao gồm ảnh và text) 35](#_Toc198503779)

[Hình 2.12. File văn bản chứa links của các trang web bị defaced 36](#_Toc198503780)

[Hình 2.13. Đoạn code thu thập file html và làm nó thành file csv 37](#_Toc198503781)

[Hình 2.14. Folder chứa các văn bản html của trang web 39](#_Toc198503782)

[Hình 2.15. Giao diện trang web mặc định 40](#_Toc198503783)

[Hình 2.16. Chức năng của trang web 40](#_Toc198503784)

[Hình 2.17. Chức năng upload tệp của web 41](#_Toc198503785)

[Hình 2.18. Giao diện xuất hiện mã độc SQL Injection 42](#_Toc198503786)

[Hình 2.19. Đoạn code chèn mã độc bằng XSS 43](#_Toc198503787)

[Hình 2.20. Giao diện trang web khi bị chèn mã độc XSS 43](#_Toc198503788)

[Hình 2.21. Đoạn mã PHP 44](#_Toc198503789)

[Hình 2.22. Giao diện thành công khi upload file 45](#_Toc198503790)

[Hình 2.23. Giao diện của web sau mở file php từ thư mục 45](#_Toc198503791)

[Hình 2.24. Giao diện của trang web khi mở 46](#_Toc198503792)

[Hình 2.25. Nhập lệnh whoami để hiện tên người dùng 46](#_Toc198503793)

[Hình 3.1. Chi tiết folder image 47](#_Toc198503794)

[Hình 3.2. Folder image sau khi lọc các hình ảnh thành 3 nhãn 50](#_Toc198503795)

[Hình 3.3. Dữ liệu trong folder defaced 51](#_Toc198503796)

[Hình 3.4. Dữ liệu trong folder not\_defaced 51](#_Toc198503797)

[Hình 3.5. Dữ liệu trong folder not\_found 52](#_Toc198503798)

[Hình 3.6. Biểu đồ thống kế dữ liệu hình ảnh (lần thứ nhất) 52](#_Toc198503799)

[Hình 3.7. Biểu đồ thống kê dữ liệu hình ảnh (lần thứ hai) 54](#_Toc198503800)

[Hình 3.8. Thông số chi tiết dataset defaced 55](#_Toc198503801)

[Hình 3.9. Biểu đồ thống kê số liệu của dataset hình ảnh (lần cuối cùng) 56](#_Toc198503802)

[Hình 3.10. Đoạn code lọc dữ liệu html 58](#_Toc198503803)

[Hình 3.11. Quá trình xử lý code và dữ liệu 58](#_Toc198503804)

[Hình 3.12. Các Folder dataset về html sau khi lọc 59](#_Toc198503805)

[Hình 4.1. Sơ đồ kiến trúc mô hình 63](#_Toc198503806)

[Hình 4.2. Nạp mô hình 64](#_Toc198503807)

[Hình 4.3. Kiểm thử lần 1 với kết quả phát hiện tấn công Defaced 65](#_Toc198503808)

[Hình 4.4. Kiểm thử lần 2 với kết quả bình thường 65](#_Toc198503809)

[Hình 4.5. Biểu đồ Loss và Accuracy 66](#_Toc198503810)

DANH MỤC BẢNG

[*Bảng 2.1. Giải thích chi tiết về dòng code cào links trên web Zone-H 18*](#_Toc198504198)

[*Bảng 2.2. Giải thích chi tiết code cào links thứ hai 29*](#_Toc198504199)

[*Bảng 2.3. Giải thích chi tiết về dòng code thu thập hình ảnh của các trang web và trích xuất văn bản về chúng 34*](#_Toc198504200)

[*Bảng 2.4. Giải thích code tải các trang html về và lưu vào file csv 38*](#_Toc198504201)

Bảng phân công và đánh giá hiệu suất

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và Tên** | **MSSV** | **Phân công công việc** | **Đánh giá của nhóm** |
| **Lê Trần Đức Thiện** | 23521478 | - Lời mở đầu và cảm ơn  - Chương 1: Các phương pháp xử lý dữ liệu  - Chương 2: Viết code, thu thập dữ liệu, dataset hình ảnh các trang web  - Chương 3: Phân tích, xử lý và đánh giá dataset hình ảnh  - Chương 5: Đánh giá bài báo cáo | Qua quá trình làm đồ án, hoàn thành các công việc được giao đầy đủ, hoàn thiện. Có sự kỹ lưỡng và cẩn thận. Cũng như giúp đỡ hoàn thành format của bài báo cáo  Mức độ đánh giá: 100% |
| **Từ Lý Mai** | 23520912 | - Chương 1: Giới thiệu về web defaced, tầm quan trọng của dataset trong an ninh mạng  - Chương 2: Viết code, thu thập dữ liệu, dataset html của các trang web  - Chương 3: Phân tích, xử lý và đánh giá dữ liệu dataset html  - Chương 5: Ghi kết luận của đề tài và hướng phát triển dataset | Qua quá trình làm đồ án, hoàn thành tốt các công việc được giao, có sự chuẩn bị kỹ càng, hợp tác tốt. Góp phần hoàn thiện chỉnh chu bài báo cáo  Mức độ đánh giá: 100% |
| **Dương Vi Khắc Nhật** | 22521012 | - Chương 1: Phân loại, đặc điểm nhận dạng, phương pháp khai thác và thu thập dữ liệu  - Chương 2: Bổ sung dữ liệu bằng cách tự tạo web và tự tấn công  - Chương 4: Xây dựng và huấn luyện mô hình. Đánh giá tỷ lệ  - Chương 5: Hướng phát triển của mô hình | Qua quá trình làm đồ án, hoàn thành các công việc được giao một cách chỉnh chu, giúp đỡ các thành viên khác trong nhóm với kiến thức mới. Khai thác và rèn luyện mô hình tốt  Mức độ đánh giá: 100% |

CHƯƠNG 1: TỔNG QUAN VỀ TẤN CÔNG WEB DEFACED

# 1.1 Khái niệm về tấn công deface

- **Defacement** là hành vi phá hoại hình ảnh của một ai, một thứ, một đơn vị nào đó. Trong tin học, ta có **Website Defacement** – là tấn công, phá hoại website, làm thay đổi giao diện hiển thị của một trang web. Nói cách khác, khi người dùng truy cập vào địa chỉ của trang web đó thì giao diện của một trang web khác sẽ được hiển thị. Thông thường nội dung hiện lên sẽ là các thông điệp mà hacker muốn truyền tải.

# 1.2 Nguyên nhân và kỹ thuật tấn công defacement

## 1.2.1 Nguyên nhân dẫn đến tấn công defacement

Các nguyên nhân khiến website trở thành **mục tiêu** của defacement có thể được chia thành ba nhóm chính: lỗ hổng kỹ thuật, cấu hình sai, và yếu tố con người. Dưới đây là phân tích chi tiết từng nhóm, kèm ví dụ minh họa:

### 1.2.1.1 Lỗ hổng kỹ thuật:

- Mô tả: Các lỗ hổng trong mã nguồn, CMS, hoặc plugin tạo điều kiện cho hacker xâm nhập và thay đổi website. Đây là nguyên nhân phổ biến nhất, chiếm khoảng 70% các vụ deface (theo OWASP, 2024).

- Các lỗ hổng tiêu biểu:

* CMS lỗi thời: WordPress, Joomla, Drupal thường bị tấn công nếu không cập nhật bản vá (ví dụ: CVE-2020-24186 trong WordPress).
* Plugin không an toàn: Plugin hoặc theme chứa lỗ hổng (như WP File Manager) dễ bị khai thác.
* Lỗ hổng ứng dụng web: SQL Injection, Cross-Site Scripting (XSS), Local/Remote File Inclusion (LFI/RFI).

### 1.2.1.2 Cấu hình sai

- Mô tả: Các sai lầm trong cấu hình máy chủ, cơ sở dữ liệu, hoặc dịch vụ web tạo ra điểm yếu để hacker khai thác. Theo SANS Institute (2024), cấu hình sai chiếm 20% nguyên nhân defacement.

- Các lỗi cấu hình phổ biến:

* Máy chủ để lộ thư mục: Directory listing bật (cho phép hacker thấy tệp cấu hình).
* Dịch vụ không mã hóa: FTP, SSH sử dụng mật khẩu yếu hoặc không bật 2FA.
* Quyền truy cập quá mức: Tệp hoặc thư mục có quyền 777 (cho phép ghi).
* Tệp cấu hình lộ thông tin: Tệp .htaccess hoặc wp-config.php không được bảo vệ.

### 1.2.1.3 Yếu tố con người

- Mô tả: Sai lầm của quản trị viên, nhân viên, hoặc người dùng cuối tạo cơ hội cho hacker. Đây là nguyên nhân gián tiếp nhưng đóng vai trò lớn, đặc biệt trong các vụ phishing hoặc social engineering.

- Các yếu tố phổ biến:

* Mật khẩu yếu: Sử dụng mật khẩu dễ đoán như “123456” hoặc “admin”.
* Thiếu nhận thức bảo mật: Quản trị viên không cập nhật phần mềm hoặc kiểm tra log.
* Phishing/Social engineering: Hacker lừa nhân viên cung cấp thông tin đăng nhập.
* Quản lý tài khoản lỏng lẻo: Không sử dụng xác thực hai yếu tố (2FA) hoặc để lộ thông tin qua mạng xã hội.

### 1.2.1.4 Thiếu giám sát và bảo trì

- Mô tả: Không triển khai các công cụ giám sát (WAF, IDS) hoặc không bảo trì định kỳ khiến website dễ bị tấn công mà không được phát hiện kịp thời.

- Các vấn đề:

* Không sử dụng tường lửa ứng dụng web (WAF) để chặn tấn công.
* Không phân tích log máy chủ để phát hiện truy cập trái phép.
* Không có bản sao lưu định kỳ, khiến việc khôi phục khó khăn.

## 1.2.2 Một số kỹ thuật tấn công defacement

### 1.2.2.1. SQL Injection

- Hacker chèn mã SQL độc hại vào các trường nhập liệu (như form đăng nhập, tìm kiếm) để thao túng cơ sở dữ liệu, thay đổi nội dung hiển thị hoặc xóa dữ liệu.

- SQL Injection cho phép thay đổi nội dung mà không cần quyền quản trị, đặc biệt nguy hiểm với website sử dụng CMS.

### 1.2.2.2 Cross-Site Scripting (XSS)

- Hacker chèn mã JavaScript độc hại vào website để hiển thị nội dung deface, chuyển hướng người dùng, hoặc đánh cắp thông tin.

- XSS dễ thực hiện, đặc biệt với các website không lọc đầu vào (input sanitization).

### 1.2.2.3 Remote File Inclusion (RFI) và Local File Inclusion (LFI)

- Hacker nhúng tệp độc hại từ máy chủ bên ngoài (RFI) hoặc truy cập tệp nội bộ (LFI) để thực thi mã hoặc thay đổi nội dung website.

- RFI/LFI cho phép hacker kiểm soát toàn bộ máy chủ nếu quyền truy cập không bị giới hạn.

### 1.2.2.4 Brute-force tài khoản quản trị

- Hacker sử dụng công cụ tự động để đoán mật khẩu tài khoản quản trị, từ đó đăng nhập và thay đổi nội dung website.

- Brute-force hiệu quả với các website không giới hạn số lần thử đăng nhập hoặc không dùng 2FA.

### 1.2.2.5 Upload file độc hại

- Hacker khai thác lỗ hổng trong form upload (như tải ảnh, tài liệu) để tải lên web shell hoặc tệp độc hại, từ đó thay đổi nội dung hoặc kiểm soát máy chủ.

- Upload file là kỹ thuật nguy hiểm, cho phép hacker duy trì quyền truy cập lâu dài.

## 1.2.3 Phân tích kỹ thuật và công cụ hỗ trợ

Hacker thường sử dụng các công cụ tự động hoặc bán tự động để tìm kiếm và khai thác lỗ hổng, tăng hiệu quả tấn công defacement. Dưới đây là các công cụ phổ biến:

* Nmap: Quét máy chủ để tìm dịch vụ mở (FTP, SSH) hoặc phiên bản phần mềm lỗi thời.
* Metasploit: Framework khai thác lỗ hổng CMS, SQL Injection, hoặc RFI.
* Burp Suite: Phân tích yêu cầu HTTP để tìm XSS, upload file, hoặc lỗ hổng form.
* SQLMap: Tự động hóa SQL Injection để truy cập cơ sở dữ liệu.
* Hydra/Medusa: Thực hiện brute-force trên form đăng nhập hoặc dịch vụ như FTP.
* SET (Social-Engineer Toolkit): Tạo email phishing hoặc trang giả mạo để đánh cắp thông tin.

## 1.2.4 Ví dụ về các cuộc tấn công phá hoại trang web

Một số trang web lớn nhất thế giới đã từng bị tấn công phá hoại tại một thời điểm nào đó. Một cuộc tấn công phá hoại là một chỉ báo công khai cho thấy một trang web đã bị xâm phạm và gây thiệt hại cho thương hiệu và danh tiếng, kéo dài rất lâu sau khi tin nhắn của kẻ tấn công đã bị xóa.

### 1.2.4.1. Tấn công phá hoại NHS

Năm 2018, BBC đưa tin rằng một trang web lưu trữ dữ liệu từ các cuộc khảo sát bệnh nhân do Dịch vụ Y tế Quốc gia Anh (NHS) điều hành đã bị tin tặc phá hoại. Tin nhắn phá hoại có nội dung "Bị AnoaGhost phá hoại". Tin nhắn đã bị xóa trong vòng vài giờ, nhưng trang web có thể đã bị phá hoại trong năm ngày. Cuộc tấn công đã làm dấy lên mối lo ngại về tính an toàn của dữ liệu y tế do NHS kiểm soát.

### 1.2.4.2. Google.ro và PayPal.ro

Vào năm 2012, người dùng không thể truy cập Google Romania và thay vào đó được đưa đến màn hình phá hoại do MCA-CRB, "Tin tặc Algeria" đăng tải. Việc phá hoại diễn ra trong ít nhất một giờ. Cuộc tấn công được thực hiện bằng cách chiếm quyền điều khiển DNS—những kẻ tấn công đã cố gắng làm giả phản hồi DNS và chuyển hướng người dùng đến máy chủ của riêng họ thay vì máy chủ của Google. Cuộc tấn công tương tự đã được thực hiện đối với tên miền paypal.ro. Nhóm tin tặc MCA-DRB chịu trách nhiệm cho 5.530 vụ phá hoại trang web trên khắp năm châu lục, nhiều vụ trong số đó nhắm vào các trang web của chính phủ.

### 1.2.4.3. Cuộc tấn công mạng lớn nhất trong lịch sử Georgia

Vào năm 2019, Georgia, một quốc gia nhỏ ở châu Âu, đã trải qua một cuộc tấn công mạng khiến 15.000 trang web bị xóa và sau đó ngừng hoạt động. Trong số các trang web bị ảnh hưởng có các trang web của chính phủ, ngân hàng, báo chí địa phương và các đài truyền hình lớn. Một nhà cung cấp dịch vụ lưu trữ web của Georgia có tên là Pro-Service đã chịu trách nhiệm về vụ tấn công, đưa ra tuyên bố rằng một tin tặc đã xâm phạm hệ thống nội bộ của họ và xâm nhập các trang web.

# 1.3 Phân loại và đặc điểm nhận dạng defacement

## 1.3.1 Phân loại

### 1.3.1.1 Homepage Defacement

- **Khái niệm**: Homepage Defacement là hình thức tấn công phổ biến nhất trong các loại tấn công, kẻ tấn công sẽ truy cập vào máy chủ web và thay đổi nội dung của trang chủ. Mục đích thường là thể hiện quan điểm (chính trị, xã hội), tuyên truyễn chính trị, phá hoại uy tín chủ sở hữu,...

- **Đặc điểm:**

* Tập trung vào trang chủ, làm thay đổi giao diện hoặc nội dung mà người dùng thấy đầu tiên.
* Nội dung thay đổi: thông điệp, hình ảnh, quảng cáo, hoặc thông tin không liên quan mà nhóm hacker muốn thay đổi.
* Tác động trực tiếp đến hình ảnh và uy tín tổ chức/doanh nghiệp.
* Gây ảnh hưởng lớn nếu website có lưu lượng truy cập cao.

- **Cách thức tấn công:**

* Khai thác lỗ hổng CMS.
* Brute-force tài khoản quản trị.
* Lỗi upload file, SQL Injection, lỗi cấu hình server…

### 1.3.1.2 Mass Defacement

- **Khái niệm**: Hình thức này xảy ra khi hacker tấn công hàng loạt các website trong thời gian ngắn, thường nhắm vào hàng trăm hoặc hàng nghìn website cùng lúc. Các website này thường có chung điểm yếu bảo mật.

- **Đặc điểm:**

* Tấn công tự động hóa bằng các công cụ quét và khai thác lỗ hổng.
* Các website bị tấn công thường bị thay đổi giống nhau về nội dung deface.
* Nhắm vào các mục tiêu không cụ thể, chủ yếu khoe khoang hoặc gây rối.
* Thường xảy ra trên các máy chủ chia sẻ hoặc các nền tảng CMS phổ biến.

- **Cách tấn công**:

* Dò tìm lỗ hổng tự động trên diện rộng: sử dụng công cụ như Nmap, OpenVAS, hoặc script tự viết …
* Khai thác lỗ hổng hàng loạt: Tận dụng các lỗ hổng như CVE trong WordPress, Joomla, ...
* Tấn công máy chủ hosting: chiếm quyền kiểm soát máy chủ dể deface tất cả website trên cùng máy chủ.
* Tấn công qua API hoặc plugin: Khai thác các plugin hoặc giao diện API không an toàn.

### 1.3.1.3 Redefacement

**- Khái niệm**: Hiện tượng website bị tấn công lặp lại sau khi đã bị tấn công trước đó. Nguyên nhân là do các lỗ hổng bảo mật không được khắc phục triệt để hoặc không phát hiện backdoor go hacker để lại. Có thể xem Redefacement là cuộc chiến giữa các nhóm hacker để giành quyền kiểm soát website.

- **Đặc điểm**:

* Cùng nhắm vào một mục tiêu bị deface trước đó để tấn công.
* Nội dung deface có thể mang tính cạnh tranh, khiêu khích,...
* Những website dễ bị tấn công lại do thiếu bảo trì hoặc giám sát mật.
* Gây thiệt hại lâu dài về uy tín và giám sát mật.

- **Cách tấn công:**

* Sử dụng backdoor: Cài đặt các tệp độc hại trong lần tấn công đầu để truy cập lại lần sau.
* Khai thác lỗ hổng chưa vá: tái sử dụng các lỗ hổng như XSS, SQL Ịnection, …
* Brute-force tài khoản cũ: Hacker tiếp tục tấn công nếu mật khẩu quản trị không được thay đổi.
* Kiểm soát tài khoản quản trị: Sử dụng thông tin đăng nhập bị đánh cắp từ lần tấn công trước.

### 1.3.1.4 IP Address Location Defacement

- **Khái niệm**: IP Address Location Defacement là một dạng tấn công defacement kết hợp với thao túng địa chỉ IP hoặc cấu hình DNS để chuyển hướng lưu lượng truy cập đến một máy chủ giả mạo hoặc che giấu vị trí thực của website. Mục đích thường là đánh cắp dữ liệu, lừa đảo, hoặc gây nhầm lẫn cho người dùng.

- **Đặc điểm:**

* Không chỉ thay đổi về nội dung website mà còn can thiệp vào cơ chế định tuyến mạng.
* Người dùng có thể bị chuyển hướng đến một website giả mạo với giao diện giống hệt nhưng được host trên máy chủ của hacker.
* Tấn công này đòi hỏi kiến thức sâu về mạng và DNS.
* Thường khó phát hiện hơn các loại defacement khác.

- **Cách tấn công:**

* DNS Hijacking.
* ARP Spoofing.
* Khai thác máy chủ.
* Phishing kết hợp…

### 1.3.1.5 Special Defacement

**- Khái niệm:** Special Defacement là các cuộc tấn công defacement có tính chất đặc biệt, sử dụng kỹ thuật tiên tiến hoặc kết hợp với các hình thức tấn công khác (như mã độc, ransomware, hoặc gián điệp). Loại này thường nhắm vào các mục tiêu chiến lược (chính phủ, tài chính, công nghệ) và có mục đích vượt ngoài việc chỉ thay đổi giao diện.

- **Đặc điểm:**

* Nội dung thay đổi thường tinh vi, khó phát hiện (ví dụ: chỉ thay đổi một phần nhỏ của website).
* Kết hợp với các mục tiêu như lây nhiễm mã độc, đánh cắp dữ liệu, hoặc làm gián đoạn hệ thống.
* Có thể sử dụng các kỹ thuật chưa từng được công bố (zero-day).
* Thường nhắm vào các tổ chức có giá trị cao hoặc lượng truy cập lớn.

**- Cách tấn công:**

* Chèn mã độc: Thêm các đoạn mã JavaScript, iframe, hoặc tệp độc hại để lây nhiễm trình duyệt người dùng.
* Khai thác zero-day: Tận dụng lỗ hổng chưa được vá để truy cập hệ thống.
* Tấn công chuỗi cung ứng: Nhắm vào nhà cung cấp dịch vụ, plugin, hoặc CDN để deface gián tiếp.
* Social engineering: Lừa quản trị viên cung cấp quyền truy cập để thay đổi nội dung.

## 1.3.2 Một số đặc điểm nhận dạng defacement

### 1.3.2.1 Thay đổi giao diện bất thường

- Trang chủ hoặc các trang chính hiển thị nội dung lạ, như hình ảnh, logo, hoặc văn bản không thuộc thiết kế gốc.

- Bố cục, màu sắc, hoặc phông chữ thay đổi so với phiên bản gốc.

### 1.3.2.2 Xuất hiện thông điệp hoặc chữ ký hacker

- Các thông điệp như “Hacked by [Tên nhóm/hacker]”, “Owned by X”, hoặc khẩu hiệu chính trị xuất hiện trên giao diện.

- Biểu tượng hoặc hình ảnh đặc trưng của nhóm hacker (như Guy Fawkes của Anonymous).

### 1.3.2.3 Nội dung bị xóa hoặc thay thế

- Nội dung gốc (bài viết, sản phẩm, hình ảnh) bị xóa hoàn toàn hoặc thay bằng nội dung không liên quan.

- Website hiển thị lỗi 404, trang trắng, hoặc thông báo bất thường.

### 1.3.2.4 Chuyển hướng người dùng đến trang không mong muốn

- Người dùng bị chuyển hướng tự động đến trang phishing, quảng cáo, hoặc độc hại khi truy cập website.

- URL thay đổi bất thường hoặc dẫn đến tên miền lạ.

### 1.3.2.5 Phát hiện mã độc hoặc đoạn mã lạ trong mã nguồn

- Mã nguồn HTML, JavaScript, hoặc cơ sở dữ liệu chứa các đoạn mã bất thường, như iframe, script từ máy chủ lạ, hoặc web shell.

- Tệp như index.php hoặc wp-config.php bị chỉnh sửa với nội dung độc hại.

# 1.4 Mục tiêu của kẻ tấn công

## 1.4.1 Tuyên truyền chính trị hoặc xã hội:

- Mục đích: sử dụng defacement để truyền tải thông điệp chính trị, phản động, hoặc ủng hộ cho một phong trào xã hội. Nhằm thu hút sự chú ý của công chúng hoặc gây áp lực lên tổ chức nào đó.

- Đặc điểm:

* Nội dung deface thường bao gồm khẩu hiệu, biểu tượng, hoặc thông điệp mang tính biểu tượng.
* Nhắm vào các website có tầm ảnh hưởng lớn (chính phủ, tổ chức quốc tế).
* Thường công khai trên mạng xã hội hoặc Zone-H để lan tỏa thông điệp.

## 1.4.2 Phá hoại uy tín hoặc gây rối:

- Mục đích: Hacker nhằm làm tổn hại hình ảnh, uy tín của tổ chức, doanh nghiệp, hoặc cá nhân bằng cách thay đổi nội dung website thành các thông điệp xúc phạm, lăng mạ, hoặc gây tranh cãi. Đây thường là hành động phá hoại (vandalism) hoặc trả thù cá nhân.

- Đặc điểm:

* Nội dung deface mang tính khiêu khích, thường là hình ảnh tục tĩu, văn bản chế giễu, hoặc nội dung không liên quan.
* Nhắm vào các website có lượng truy cập cao để tối đa hóa tác động.
* Thường đi kèm chữ ký hacker để khoe khoang.

## 1.4.3 Chứng tỏ năng lực hoặc danh tiếng

- Mục đích: Hacker, đặc biệt là “script kiddies” hoặc hacker mới vào nghề, thực hiện defacement để chứng minh kỹ năng, đánh bóng tên tuổi cá nhân/nhóm, hoặc cạnh tranh trong cộng đồng hacker.

- Đặc điểm:

* Nội dung deface thường đơn giản, tập trung vào chữ ký hacker hoặc thông điệp khoe khoang.
* Nhắm vào các website dễ bị tấn công, không nhất thiết có giá trị cao.
* Thường gửi báo cáo đến Zone-H để được công nhận.

## 1.4.4 Cảnh báo lỗ hổng bảo mật

- Mục đích: Một số hacker (thường là “white hat” hoặc “gray hat”) deface website để chỉ ra các lỗ hổng bảo mật, khuyến khích quản trị viên khắc phục trước khi bị hacker độc hại khai thác.

- Đặc điểm:

* Nội dung deface thường thân thiện, kèm lời khuyên về bảo mật.
* Hacker có thể liên hệ trực tiếp với quản trị viên để báo cáo.
* Ít gây thiệt hại nghiêm trọng, tập trung vào cảnh báo.

## 1.4.5 Lợi ích tài chính

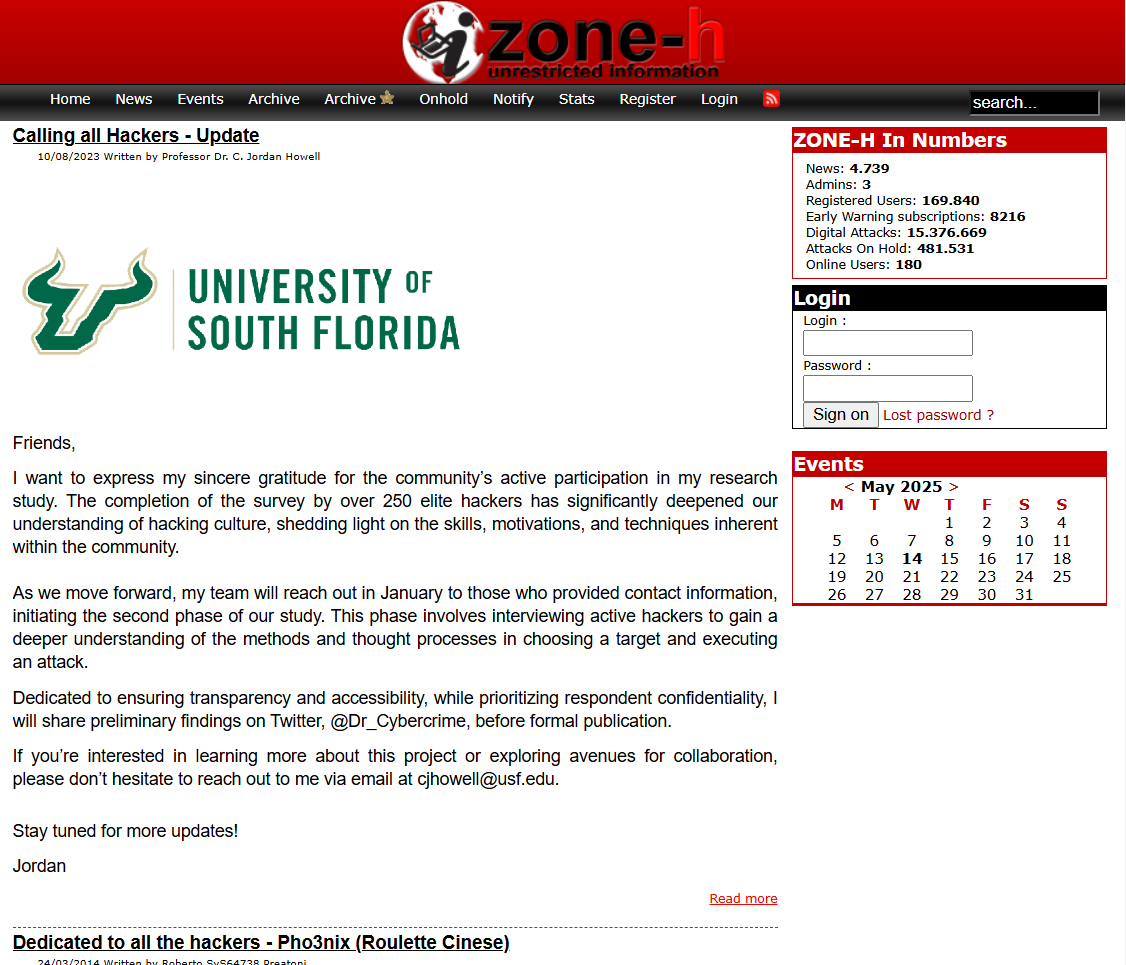
- Mục đích: Hacker deface website để chuyển hướng người dùng đến các trang lừa đảo, chèn quảng cáo trái phép, hoặc tống tiền bằng cách đe dọa tiếp tục tấn công.

- Đặc điểm:

* Nội dung deface thường liên quan đến chuyển hướng (redirect) hoặc quảng cáo.
* Có thể kết hợp với phishing hoặc ransomware.
* Hacker yêu cầu tiền chuộc để ngừng tấn công.

# 1.5 Các công cụ theo dõi và thống kê (Zone-H)

## 1.5.1 Khái niệm:



Hình 1.1. Giao diện trang web Zone-H

- Zone-H là một kho lưu trữ trực tuyến hàng đầu, được thành lập năm 2002, chuyên ghi nhận, lưu trữ, và phân tích các vụ tấn công defacement trên toàn cầu. Zone-H đóng vai trò như một “bảo tàng số” cho các website bị deface, cung cấp thông tin chi tiết về nội dung tấn công, hacker/nhóm hacker, và thời gian xảy ra.

## 1.5.2 Các tính năng của Zone-H

### 1.5.2.1 Lưu trữ dữ liệu defacement:

- Zone-H lưu giữ bản sao (mirror) của giao diện website bị deface, bao gồm ảnh chụp màn hình, nội dung văn bản, và mã nguồn (nếu có).

- Hacker hoặc quản trị viên có thể gửi báo cáo để lưu trữ trên Zone-H.

### 1.5.2.2 Thống kê và phân tích:

- Cung cấp số liệu chi tiết về số lượng deface theo quốc gia, loại website và nhóm hacker.

- Báo cáo xu hướng tấn công.

VD: Số lượng vụ Mass Defacement tăng đột biến.

### 1.5.2.3 Xếp hạng hacker.

- Dựa vào số lượng website đã deface, zone-H duy trì bảng xếp hạng dành cho hacker/nhóm hacker.

VD: Nhóm “DarkShadow” từng đứng đầu bảng xếp hạng năm 2020 với hơn 1.000 website bị deface.

### 1.5.2.4 Công cụ tìm kiếm và phân loại

- Cho phép người dùng tìm kiếm các vụ deface theo tên website, tên hacker, quốc gia, hoặc thời gian.

- Hỗ trợ phân loại theo loại tấn công (Homepage, Mass, Special Defacement).

### 1.5.2.5 Cộng đồng và báo cáo

- Zone-H có diễn đàn cho các nhà nghiên cứu và hacker chia sẻ thông tin.

- Hacker có thể gửi báo cáo deface để được công nhận, trong khi quản trị viên sử dụng dữ liệu để phân tích lỗ hổng.

## 1.5.3 Cách sử dụng Zone-H

- Truy cập: Vào http://www.zone-h.org/, đăng ký tài khoản (miễn phí hoặc trả phí cho tính năng nâng cao).

- Tìm kiếm: Sử dụng mục “Archive” hoặc “Search” để tra cứu các vụ deface.

- Nộp báo cáo: Hacker hoặc quản trị viên gửi thông tin về vụ deface qua biểu mẫu trực tuyến.

- Phân tích: Các nhà nghiên cứu sử dụng API của Zone-H (nếu có) để trích xuất dữ liệu cho nghiên cứu.

## 1.5.4. Vai trò của Zone-H

- Đối với quản trị viên: Giúp phát hiện các vụ deface, xác định lỗ hổng, và học hỏi từ các vụ tấn công tương tự.

- Đối với hacker: Là nơi để khoe khoang thành tích hoặc cảnh báo lỗ hổng (đặc biệt với hacker “white hat”).

- Đối với nhà nghiên cứu: Cung cấp dữ liệu thực tế để phân tích xu hướng, kỹ thuật tấn công, và phát triển công cụ bảo mật.

- Đối với cơ quan pháp luật: Hỗ trợ điều tra các vụ tấn công mạng bằng cách cung cấp thông tin về hacker và mẫu tấn công.

# 1.6 Phương pháp khai thác và thu thập dữ liệu

Thu thập dữ liệu là bước quan trọng để nghiên cứu, phát hiện, và phòng chống defacement, bao gồm thông tin về giao diện, mã nguồn, và hành vi hacker.

## 1.6.1. Cào dữ liệu (Web Scraping)

Web scraping là phương pháp sử dụng các công cụ tự động để thu thập nội dung từ website, bao gồm văn bản, hình ảnh, mã HTML, CSS, và JavaScript, nhằm phân tích giao diện và phát hiện dấu hiệu defacement như thay đổi bất thường hoặc nội dung độc hại. Quy trình bắt đầu bằng việc xác định danh sách website mục tiêu (chính phủ, doanh nghiệp), sau đó sử dụng thư viện như BeautifulSoup hoặc Scrapy để quét nội dung, lưu trữ dữ liệu dưới dạng JSON, CSV, hoặc cơ sở dữ liệu MongoDB, và cuối cùng so sánh với bản gốc để nhận diện Visual Defacement hoặc Hidden Content. Các công cụ như Selenium hỗ trợ quét website động (dùng JavaScript), trong khi Puppeteer (Node.js) hiệu quả với các trang Single Page Application (SPA). Ứng dụng chính của phương pháp này là phát hiện thay đổi giao diện (như logo hoặc thông điệp hacker) và phân tích nội dung độc hại. Ví dụ, nhóm nghiên cứu tại WhiteHat.vn (2023) sử dụng Scrapy để quét 500 website doanh nghiệp, phát hiện 10 website có thông điệp “Hacked by…” trong 24 giờ. Tuy nhiên, web scraping đối mặt với hạn chế như bị chặn bởi CAPTCHA, giới hạn tốc độ, hoặc Cloudflare, đặc biệt với các website động tải nội dung qua API. Để khắc phục, cần sử dụng proxy luân phiên, headless browser (Selenium, Puppeteer), và tối ưu hóa tần suất quét để tránh bị chặn, đảm bảo hiệu quả và tính liên tục của quá trình thu thập dữ liệu.

## 1.6.2. Dữ liệu mô phỏng (Simulated Data)

Simulated Data tạo website giả lập trên Docker để mô phỏng defacement (thay logo, chèn mã skimmer), thu thập dữ liệu phục vụ huấn luyện AI mà không cần dữ liệu thực. Quy trình gồm thiết lập CMS (WordPress), thực hiện deface, thu thập ảnh, mã nguồn, log, và huấn luyện CNN hoặc EfficientNet. Công cụ như Docker, Metasploit, Burp Suite hỗ trợ. Phương pháp kiểm tra công cụ giám sát hoặc nhận diện Visual Defacement. Ví dụ, Đại học Bách khoa Hà Nội (2023) mô phỏng website, huấn luyện CNN đạt 92% chính xác. Hạn chế là không phản ánh thực tế, tốn tài nguyên. Khắc phục bằng kết hợp dữ liệu Zone-H và tự động hóa script Python.

## 1.6.3. Dữ liệu thực tế (Real-world Data)

Real-world Data thu thập từ website bị deface, log máy chủ, hoặc Zone-H, cung cấp thông tin thực về mẫu tấn công, kỹ thuật hacker. Quy trình gồm tải ảnh, mã nguồn từ Zone-H, phân tích log Apache/Nginx, thu thập báo cáo Bkav, và lưu vào MongoDB/MySQL. Công cụ như Zone-H API, ELK Stack, Splunk hỗ trợ. Phương pháp nghiên cứu xu hướng, huấn luyện AI. Ví dụ, Viettel (2024) dùng 5.000 ảnh Zone-H, huấn luyện EfficientNet đạt 95%. Hạn chế là dữ liệu không đầy đủ, phân tích log cần kỹ năng. Khắc phục bằng kết hợp nhiều nguồn và AI tự động hóa.

## 1.6.4. Công cụ giám sát và phát hiện

Công cụ giám sát theo dõi website thời gian thực, phát hiện thay đổi bất thường, tích hợp AI hoặc quy tắc. Quy trình gồm cài đặt Site24x7, quét định kỳ, thu thập dữ liệu giao diện, mã nguồn, hiệu suất, và phân tích Visual Defacement, Page Redirection. Công cụ như VNCS Web Monitoring, WebOrion, Sucuri SiteCheck phát hiện trong 5-10 phút. Ví dụ, Site24x7 (2023) phát hiện redirect phishing trong 5 phút. Hạn chế là chi phí, false positives. Khắc phục bằng tinh chỉnh quy tắc, kết hợp công cụ.

## 1.6.5. Phân tích hình ảnh và nội dung

Phân tích hình ảnh/nội dung dùng học sâu để phát hiện thông điệp hacker, mã độc qua ảnh, văn bản, mã nguồn. Quy trình gồm thu thập ảnh Zone-H, huấn luyện CNN (EfficientNet), dùng Tesseract OCR, NLP (spaCy), và kiểm tra mã nguồn bằng YARA. Phương pháp nhận diện Visual Defacement, Hidden Content. Ví dụ, EfficientNet (2024) đạt 94% trên 10.000 ảnh Zone-H. Hạn chế là cần dữ liệu lớn, khó phát hiện Hidden Defacement. Khắc phục bằng kết hợp phân tích mã nguồn, transfer learning.

## 1.6.6. Phân tích log và giám sát mạng

Phân tích log/mạng phát hiện hành vi đáng ngờ (brute-force, upload file) qua log máy chủ, lưu lượng mạng. Quy trình gồm thu thập log Apache/Nginx, phân tích IP lạ, lệnh SQL bằng Splunk, ELK Stack, Wireshark, và lưu trữ để điều tra. Phương pháp xác định nguồn gốc hacker, kỹ thuật như SQL Injection. Ví dụ, ELK Stack (2024) phát hiện brute-force từ Nga. Hạn chế là cấu hình phức tạp, bỏ sót zero-day. Khắc phục bằng kết hợp giám sát giao diện, AI phân tích hành vi.

Chương 2: Thu thập dữ liệu

# 2.1. Thu thập các trang web nguyên bản và bị deface

Bản chất việc thu thập 2 loại trang web mang mục đích phục vụ tạo các dataset với các nhãn là “nguyên bản” và “bị defaced”, góp phần thuận lợi cho việc đánh giá, phân tích dữ liệu cũng như giúp người sử dụng dataset dễ dàng dùng nó cho các công cụ có ích ví dụ machine learning, phát hiện bị tấn công, v.v…

## 2.1.1. Các trang web nguyên bản

* Như ta đã biết, các trang web phổ biến hằng ngày mà chúng ta đang sử dụng như Facebook, Youtube... với các giao diện quen thuộc đang là những trang web sạch sẽ và nguyên bản nhất, tức là không có dấu hiệu bị tấn công

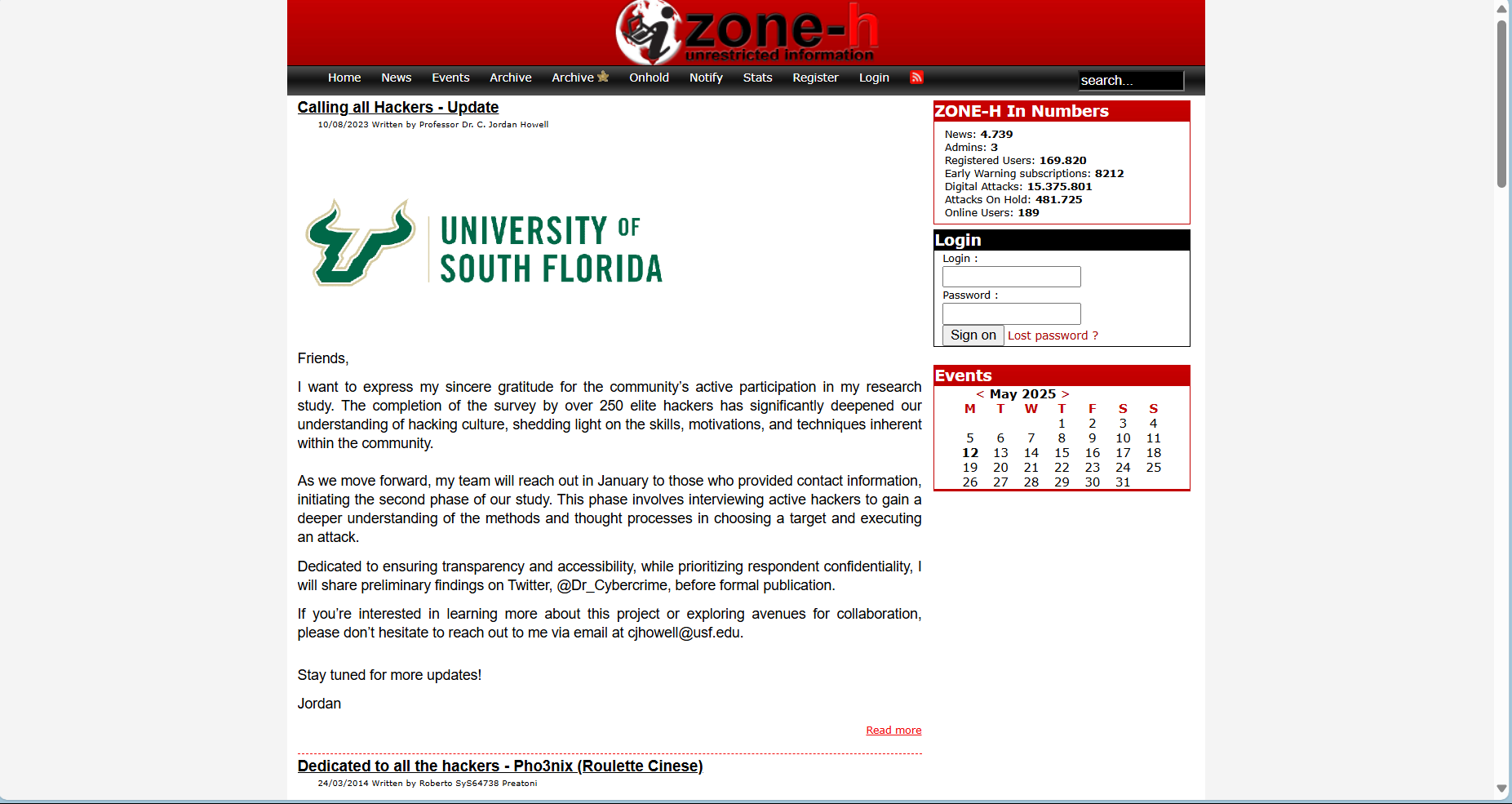


Hình 2.1. Ví dụ về web daa của ĐH CNTT với giao diện trang web sạch

* Việc thu thập các trang web này là vô cùng đơn giản vì nó hầu như ở xung quanh chúng ta, và yếu tố chính trong việc xây dựng dataset về web defaced vẫn chính là các trang web đang bị xâm nhập, nên không cần thiết các công cụ nâng cao để thu thập các đường link trang web này

## 2.1.2. Các trang web bị defaced

* Các trang web bị tấn công hầu như không dễ dàng tìm thấy ở trên internet, tuy nhiên đã có trang chủ đứng ra để giải quyết vấn đề này chính là [Zoneh.org](http://zoneh.org)



Hình 2.2. Giao diện trang web Zone-H (nơi sẽ cào dữ liệu)

* ZoneH: Tất cả thông tin chứa trong kho lưu trữ của Zone-H đều được thu thập trực tuyến từ các nguồn công khai hoặc được thông báo trực tiếp và ẩn danh cho trang chủ. Zone-H không chịu trách nhiệm về các tội phạm máy tính được báo cáo, cũng như không trực tiếp hay gián tiếp liên quan đến chúng. Bạn có thể tìm thấy một số nội dung gây khó chịu trong các trang web bị tấn công.
* Chúng ta sẽ trực tiếp thu thập các đường link được công bố trực tiếp trên ZoneH: [zone-h.org/archive/special=1](https://www.zone-h.org/archive/special=1) bằng cách sử dụng ngôn ngữ Python để cào các đường link ấy bằng 2 cách:

### 2.1.2.1. Cách cào links đầu tiên

import requests

import re

import sys

import os

import platform

# Nhập cookies

zhe\_cookie = input('\nEnter ZHE cookie: ').strip()

phpsessid\_cookie = input('Enter PHPSESSID cookie: ').strip()

cookie = {

"ZHE": zhe\_cookie,

"PHPSESSID": phpsessid\_cookie

}

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 6.1; rv:57.0) Gecko/20100101 Firefox/57.0"

}

def grab\_archive\_sites():

for i in range(1, 51):

url = f"http://www.zone-h.org/archive/special={i}"

res = requests.get(url, cookies=cookie, headers=headers)

html = res.content

print(f'\n📄 Đang lấy từ: {url}')

if b'captcha' in html:

print("⚠️ CAPTCHA detected. Hãy xác minh thủ công trên trình duyệt.")

sys.exit()

matches = re.findall(b'<td>([^<\n]+)\n\s+</td>', html)

with open("onhold\_zone.txt", 'a') as f:

for m in matches:

domain = m.split(b'/')[0].decode().strip()

print(f"[+] {domain}")

f.write(f"http://{domain}\n")

def main():

grab\_onhold\_sites()

if \_\_name\_\_ == "\_\_main\_\_":

main()

* Giải thích:

|  |  |
| --- | --- |
| **Đoạn code** | **Giải thích chi tiết** |
| import requests  import re  import sys  import os  import platform | * Thư viện **request**: dùng để gửi yêu cầu HTTP (như GET, POST) đến trang web Zone-H và nhận phản hồi. Ở đây, ta dùng nó để tải nội dung HTML từ trang * Thư viện **re**: xử lý biểu thức chính quy (regular expressions), giúp tìm kiếm và trích xuất dữ liệu (như tên domain) từ HTML * Thư viện **sys:** cung cấp các công cụ tương tác với hệ thống * Thư viện **os:** dùng để làm việc với hệ điều hành, như xóa màn hình hoặc tạo file * Thư viện **platform:** giúp xác định hệ điều hành đang chạy (Windows, Linux, macOS) để thực hiện lệnh phù hợp |
| zhe\_cookie = input('\nEnter ZHE cookie: ').strip()  phpsessid\_cookie = input('Enter PHPSESSID cookie: ').strip() | * Hiển thị dòng chữ "Enter ZHE cookie: " và chờ người dùng nhập cookie ZHE, đây sẽ là cookie trên máy tính người dùng sau khi người dùng đã vượt qua mã captcha trên trang ZoneH thì trang web này sẽ tự động lưu mã này để có thể xác thực đúng người dùng cho lần sau mà không cần phải vượt captcha lần nữa * **.strip():** xóa khoảng trắng ở đầu và cuối chuỗi mà người dùng nhập, tránh lỗi do ký tự thừa * Tương tự, yêu cầu nhập cookie PHPSESSID, cũng là một phần của thông tin đăng nhập |
| cookie = {  "ZHE": zhe\_cookie,  "PHPSESSID": phpsessid\_cookie  }  headers = {  "User-Agent": "Mozilla/5.0 (Windows NT 6.1; rv:57.0) Gecko/20100101 Firefox/57.0"  } | * Tạo một dictionary chứa hai cookie vừa nhập. Khi gửi yêu cầu đến Zone-H, cookie này sẽ được gửi kèm để xác thực (như đăng nhập vậy) * Tương tự vậy, tạo một dictionary chứa thông tin "User-Agent" (giả lập trình duyệt Firefox). Điều này giúp trang web nghĩ rằng yêu cầu đến từ trình duyệt thật, tránh bị chặn |
| def grab\_archive\_sites():  for i in range(1, 51):  url = f"http://www.zone-h.org/archive/special={i}"  res = requests.get(url, cookies=cookie, headers=headers)  html = res.content  print(f'\n📄 Đang lấy từ: {url}')  if b'captcha' in html:  print("⚠️ CAPTCHA detected. Hãy xác minh thủ công trên trình duyệt.")  sys.exit()  matches = re.findall(b'<td>([^<\n]+)\n\s+</td>', html)  with open("onhold\_zone.txt", 'a') as f:  for m in matches:  domain = m.split(b'/')[0].decode().strip()  print(f"[+] {domain}")  f.write(f"http://{domain}\n") | * Mục đích: lấy danh sách các trang web đang chờ duyệt (on hold) từ Zone-H * **Url:** cho đường link ZoneH (nơi lưu trữ các đường link trang web bị defaced) lặp lại, chạy từ trang 1 đến trang 50 * **Res:** gửi yêu cầu GET đến URL, kèm cookie và headers đã khai báo ở trên * **Html = res.content:** lấy nội dung HTML dưới dạng bytes (chưa decode thành text) * **print(f’\ Đang lấy từ…**: in URL đang xử lý * **If b’captcha’ in html:** kiểm tra xem HTML có chứa từ "captcha" không (dấu hiệu bị yêu cầu xác minh CAPTCHA). Nếu có in dòng chữ cảnh báo và lập tức thoát chương trình * **Matches:** dùng biểu thức chính quy để tìm tất cả các domain trong thẻ <td> của HTML * **open(“onhold\_zone.txt”,...)**: mở file “onhold\_zone.txt” ở chế độ append ('a'), nếu file chưa tồn tại thì tạo mới * **For m in matches:** lặp qua các domain tìm được   + **Domain:** lấy phần tên domain (trước dấu /), chuyển từ bytes sang text (decode()), và xóa khoảng trắng   + In domain ra màn hình với dấu với **print**   + Ghi domain vào file, thêm http:// và xuống dòng bằng lệnh write |
| def main():    grab\_onhold\_sites()  if \_\_name\_\_ == "\_\_main\_\_":  main() | * Chạy hàm **grab\_archive\_sites** đã khai báo ở trên * Kiểm tra xem script có được chạy trực tiếp không (không phải import từ file khác) bằng xem điều kiện **if \_\_name\_\_** * Gọi hàm **main()** để bắt đầu chương trình |

Bảng 2.1. Giải thích chi tiết về dòng code cào links trên web Zone-H

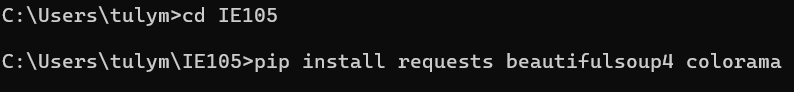
* Kết quả file onhold\_zone.txt lên đến 1250 đường link:



Hình 2.3. File văn bản chứa các đường links đã cào

### 2.1.2.2. Cách cào links thứ hai

* Mục tiêu: Truy xuất danh sách các domain(tên miền) đã bị tấn công và được khai báo trên Zone-H.
* Có 2 chế độ thu thập:
* Theo Notifier: Cho phép người dùng nhập vào tên Hacker để truy cập danh sách bị tấn công.
* Theo OnHold: Thu thập các website đã được báo cáo nhưng chưa được xác minh.
* Quy trình thực hiện:
* Yêu cầu thư viện: cần cài thư viện sau trước khi chạy chương trình:



Hình 2.5. Cài đặt thư viện resquest, beautifulsoup và colorama

* Chạy chương trình

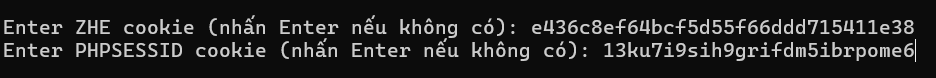


Hình 2.6. Chạy chương trình trong terminal

* Sau khi chạy, chương trình sẽ yêu cầu nhập:
* ZHE Cookie và PHPSESSID Cookie: dùng để vượt qua CAPTCHA của Zone-H khi cần.

A screenshot of a computer

AI-generated content may be incorrect.



Hình 2.7. Cách lấy thông tin cookie mà chương trình yêu cầu

* Lựa chọn chế độ thu thập:
  + 1: Theo Notifier (người báo cáo vụ deface).
  + 2: Theo trạng thái "OnHold" (chưa duyệt).

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 2.8. Chọn chế độ cào trang web

* Chương trình:

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 2.9. Đoạn code cào links của web bị defaced (1)

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 2.10. Đoạn code cào links của web bị defaced (2)

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 2.11. Đoạn code cào links của web bị defaced (3)

* Giải thích mã nguồn:

|  |  |
| --- | --- |
| Code | Giải thích |
| import requests  import re  import os  import csv  import sys  import platform  from bs4 import BeautifulSoup  from colorama import Fore, Style, init | //Khai báo thư viện.   * requests: Gửi HTTP request để lấy nội dung website. * re: Xử lý biểu thức chính quy để trích xuất thông tin. * os: Tạo thư mục, xử lý đường dẫn. * csv: Ghi dữ liệu đầu ra vào file CSV. * platform: Xác định hệ điều hành, hỗ trợ xóa màn hình. * bs4 (BeautifulSoup): Phân tích và trích xuất dữ liệu HTML. * colorama: Tô màu văn bản đầu ra để dễ nhận biết trạng thái chương trình. |
| init(autoreset=True)  os.makedirs("html\_pages", exist\_ok=True)  os.makedirs("data", exist\_ok=True) | //Tạo thư mục html\_pages để lưu HTML các site bị defaced và data để lưu file CSV.   * init(autoreset=True): Giúp văn bản màu tự động reset về mặc định sau mỗi lần in. * os.makedirs: Tạo thư mục html\_pages/ để lưu HTML và data/ để lưu CSV. |
| headers = {  "User-Agent": "Mozilla/5.0"  } | Tạo header giả lập trình duyệt (tránh bị chặn bởi server). |
| zhe\_cookie = input('\nEnter ZHE cookie (nhấn Enter nếu không có): ').strip()  phpsessid\_cookie = input('Enter PHPSESSID cookie (nhấn Enter nếu không có): ').strip() | Một số trang Zone-H yêu cầu đăng nhập hoặc xác thực CAPTCHA, nên nếu có cookie sẵn thì nhập vào. |
| cookie = {}  if zhe\_cookie and phpsessid\_cookie:  cookie = {"ZHE": zhe\_cookie, "PHPSESSID": phpsessid\_cookie} | Nếu người dùng nhập cookie, tạo dictionary cookie để gắn vào mỗi HTTP request. |
| ef clean\_text(s):  return s.replace("\n", " ").replace(",", ";").strip() if s else "" | Hàm phụ: loại bỏ ký tự xuống dòng, thay dấu phẩy bằng dấu chấm phẩy để tránh lỗi CSV, và loại bỏ khoảng trắng đầu-cuối. |
| def save\_html(domain, html\_text):  fname = domain.replace("/", "\_") + ".html"  path = os.path.join("html\_pages", fname)  with open(path, "w", encoding="utf-8") as f:  f.write(html\_text)  return fname | //Lưu HTML về máy  Đặt tên file theo domain, lưu HTML vào thư mục html\_pages. |
| def extract\_info(url):  try:  res = requests.get(url, headers=headers, cookies=cookie, timeout=10)  html = res.text  soup = BeautifulSoup(html, "html.parser")  text = soup.get\_text()  html\_file = save\_html(url.replace("http://", "").replace("https://", ""), html) | //Hàm extract\_info(): Lấy nội dung HTML từ URL, phân tích và trích xuất các thông tin như:   * Truy cập website, lấy mã HTML. * Phân tích HTML với BeautifulSoup. * Lấy toàn bộ text (dùng cho regex). * Lưu file HTML về máy. |
| # Hacked by  hacked\_by = re.search(r"Hacked By ([^\n<]+)", text, re.IGNORECASE)  message = soup.find("h1") | * Tìm cụm "Hacked By <tên>" trong text. * Tìm thông điệp đầu tiên được in đậm (giả định đặt trong thẻ <h1>). |
| # Contact info  emails = re.findall(r"[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+", text)  telegram = re.search(r"(https?://t\.me/\S+)", text)  phones = re.findall(r"\+?\d[\d\s\-\(\)]{7,}", text)  zip\_files = re.findall(r"\S+\.zip", text) | Dò tìm thông tin liên hệ:   * Email (qua regex). * Link Telegram. * Số điện thoại. * Link file .zip (nhiều hacker nhúng tools vào zip). |
| # Status  status\_code = ""  if "404 Not Found" in text:  status\_code = "404 Not Found"  elif "406 Not Acceptable" in text:  status\_code = "406 Not Acceptable" | Xác định trạng thái HTTP trả về bằng cách kiểm tra nội dung HTML (trong trường hợp server không phản hồi mã HTTP rõ ràng). |
| blocked\_by = ""  if "cloudflare" in text.lower():  blocked\_by = "Cloudflare"  elif "mod\_security" in text.lower():  blocked\_by = "ModSecurity" | Nếu trang bị chặn bởi firewall như Cloudflare hoặc ModSecurity, lưu lại thông tin này. |
| short\_text = text[:200].replace("\n", " ").strip() | Lấy 200 ký tự đầu tiên từ nội dung trang – để phân tích sơ bộ nội dung. |
| return [  url,  html\_file,  clean\_text(hacked\_by.group(1) if hacked\_by else ""),  clean\_text(message.text if message else ""),  ", ".join(set(emails)),  telegram.group(1) if telegram else "",  ", ".join(set(phones)),  ", ".join(zip\_files),  short\_text,  status\_code,  blocked\_by  ] | Trả về danh sách các thông tin đã trích xuất để ghi vào CSV. |
| def grab\_domains(mode, notifier=None): | Hàm thu thập danh sách domain từ Zone-H. |
| for i in range(1, 51): | Lặp qua 50 trang đầu tiên để thu thập dữ liệu. |
| url = f"http://www.zone-h.org/archive/notifier={notifier}/page={i}" if mode == "notifier" else f"http://www.zone-h.org/archive/published=0/page={i}" | * Nếu người dùng chọn chế độ "notifier", thu thập theo người đăng (notifier). * Nếu chọn "OnHold", lấy danh sách site chưa duyệt (có thể là site mới bị hack). |
| res = requests.get(url, cookies=cookie, headers=headers)  html = res.content | Gửi request và lấy nội dung HTML trang kết quả. |
| if b'captcha' in html:  print(Fore.YELLOW + "⚠️ CAPTCHA phát hiện! Xác minh thủ công trong trình duyệt.")  break | Nếu gặp CAPTCHA, ngừng việc cào tự động (vì cần xác minh thủ công). |
| matches = re.findall(b'<td>([^<\n]+)\n\s+</td>', html) | Tìm danh sách domain nằm trong thẻ <td> |
| for m in matches:  domain = m.decode(errors='ignore').strip().split("/")[0]  if "." in domain:  domains.append(f"http://{domain}") | Lấy phần domain từ mỗi kết quả và chuẩn hóa thành URL. |
| def main(): | Hàm chính để điều phối toàn bộ chương trình. |
| os.system('cls' if platform.system() == 'Windows' else 'clear') | Dọn màn hình terminal trước khi chạy (hỗ trợ đa nền tảng). |
| print("\n1. Cào site theo Notifier")  print("2. Cào site OnHold (chưa duyệt)")  choice = input("Chọn chế độ (1-2): ").strip() | Giao diện tương tác: người dùng chọn cách thu thập domain. |
| if choice == "1":  notifier = input("Nhập tên Notifier: ").strip()  domains = grab\_domains("notifier", notifier)  csv\_name = f"data/defaced\_by\_{notifier}.csv"  elif choice == "2":  domains = grab\_domains("onhold")  csv\_name = "data/defaced\_onhold.csv" | Lưu domain theo chế độ đã chọn, đồng thời tạo tên file CSV để lưu kết quả. |
| with open(csv\_name, "w", newline="", encoding="utf-8") as f:  writer = csv.writer(f)  writer.writerow([  "URL", "HTML\_File", "Hacked\_By", "Message", "Emails",  "Telegram", "Phones", "Zip\_Files", "Short\_Text",  "Status\_Code", "Blocked\_By"  ]) | Tạo file CSV mới, ghi dòng tiêu đề gồm các cột thông tin. |
| for url in domains:  result = extract\_info(url)  if result:  writer.writerow(result)  print(Fore.GREEN + f"✔ Ghi dữ liệu: {url}") | * Với mỗi domain, gọi extract\_info để trích xuất dữ liệu. * Nếu thành công, ghi vào CSV và thông báo đã hoàn tất. |
| if \_\_name\_\_ == "\_\_main\_\_":  main() | Nếu file đang được chạy trực tiếp (không phải import từ nơi khác), gọi hàm main() để bắt đầu chương trình. |

Bảng 2.2. Giải thích chi tiết code cào links thứ hai

# 2.2. Thu thập hình ảnh của các trang web bị deface

Dựa vào các đường link mà ta đã cào từ phần trên, ta sử dụng nó để chụp ảnh các trang web này

## 2.2.1. Thực hiện chương trình

* Ý tưởng:
  + Ta sẽ tiếp tục sử dụng ngôn ngữ Python vì thư viện mà ngôn ngữ này hỗ trợ mang lại nhiều lợi ích trong việc chụp ảnh màn hình tự động cũng như truy cập các trang web với số lượng lớn
  + Ta sẽ tạo driver một trình duyệt ảo nằm sẵn trên máy tính (ở bài báo cáo này sẽ dùng trình duyệt Edge)
  + Sau đó ta dùng trình duyệt ảo này để truy cập đến từng trang web trong file “Onhold\_Zone.txt” để chụp ảnh màn hình các trang web này lại vào lưu chúng vào một file
  + Để thuận tiện cho việc sàng lọc ảnh sau này, ta có thể dùng các thư viện để hỗ trợ phân tích chữ trong các hình ảnh rồi xuất ra các file txt nội dung của ảnh đó và kẹp chúng vào 1 folder đặt tên là đường dẫn của web đó
* Đoạn code thực hiện chương trình:

from selenium import webdriver

from selenium.webdriver.edge.service import Service

from webdriver\_manager.microsoft import EdgeChromiumDriverManager

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.common.exceptions import WebDriverException

import os

import re

import pytesseract

from PIL import Image

# Cấu hình đường dẫn Tesseract (CẦN CÀI ĐẶT TRƯỚC)

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Đọc các URL từ file onhold\_zone.txt

file\_path = "D:/IE105/onhold\_zone.txt"

with open(file\_path, "r") as file:

urls = [line.strip() for line in file.readlines() if line.strip()]

# Cấu hình trình duyệt edge ở chế độ headless

edge\_options = webdriver.EdgeOptions()

edge\_options.add\_argument("--headless")

edge\_options.add\_argument("--disable-gpu")

edge\_options.add\_argument("--no-sandbox")

# Khởi tạo trình duyệt

try:

driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()), options=edge\_options)

except WebDriverException as e:

print(f"Lỗi khi khởi tạo WebDriver: {e}")

exit()

# Lặp qua từng URL và xử lý

for index, url in enumerate(urls, start=1):

try:

print(f"\nĐang xử lý URL {index}/{len(urls)}: {url}")

# Truy cập URL và chụp màn hình

driver.get(url)

folder\_name = re.sub(r'[^\w\-]', '\_', url)

folder\_path = os.path.join("d:/IE105/image", folder\_name)

os.makedirs(folder\_path, exist\_ok=True)

# Lưu ảnh chụp màn hình

screenshot\_path = os.path.join(folder\_path, f"screenshot\_{index}.png")

driver.save\_screenshot(screenshot\_path)

print(f"Đã lưu ảnh tại: {os.path.abspath(screenshot\_path)}")

# Phân tích hình ảnh bằng OCR

try:

# Mở và xử lý ảnh

image = Image.open(screenshot\_path)

# Chuyển đổi ảnh sang grayscale để tăng độ chính xác

image = image.convert('L')

# Trích xuất văn bản

extracted\_text = pytesseract.image\_to\_string(image, lang='eng+vie') # Hỗ trợ cả tiếng Anh và Việt

# Lưu kết quả ra file txt

text\_file\_path = os.path.join(folder\_path, f"text\_analysis\_{index}.txt")

with open(text\_file\_path, 'w', encoding='utf-8') as f:

f.write(extracted\_text)

print(f"Đã phân tích và lưu văn bản tại: {text\_file\_path}")

except Exception as ocr\_error:

print(f"Lỗi OCR: {ocr\_error}")

# Đóng trình duyệt

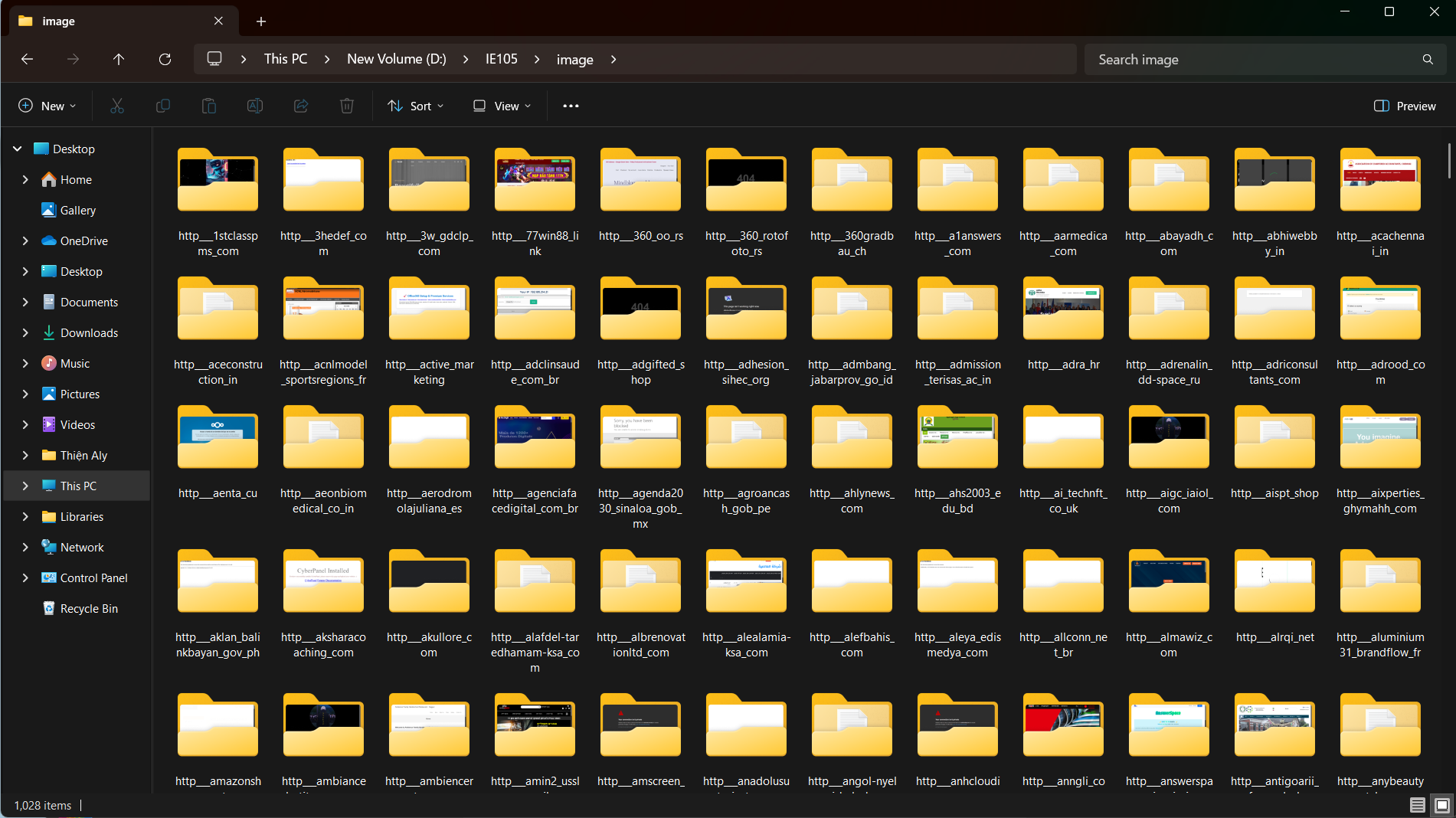
driver.quit()

* Giải thích:
  + Thuật ngữ **Tesseract**: là một công cụ OCR (Optical Character Recognition) mạnh mẽ để trích xuất văn bản từ ảnh, và việc cấu hình nó đúng rất quan trọng để script hoạt động mượt mà
  + **OCR (Optical Character Recognition - Nhận dạng ký tự quang học)** là một công nghệ cho phép máy tính hoặc thiết bị điện tử nhận diện và chuyển đổi các ký tự, chữ viết trên hình ảnh thành văn bản có thể chỉnh sửa được

|  |  |
| --- | --- |
| **Đoạn code** | **Giải thích chi tiết** |
| from selenium import webdriver  from selenium.webdriver.edge.service import Service  from webdriver\_manager.microsoft import EdgeChromiumDriverManager  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  from selenium.common.exceptions import WebDriverException  import os  import re  import pytesseract  from PIL import Image | * Thư viện **selenium với package webdriver**: tự động hóa trình duyệt, giúp mở website, chụp ảnh, tương tác trang web * Thư viện **selenium.webdriver.edge.service với package Service**: cung cấp cách khởi tạo trình duyệt Edge * Thư viện **webdriver… với package EdgeChromium…:** tự động tải và cài driver cho Edge, không cần em cài thủ công * Thư viện **selenium…ui với package WebDriverWait:** công cụ chờ cho đến khi trang tải xong * Thư viện **selenium…support với package EC:** các điều kiện chờ (như chờ phần tử xuất hiện) * Thư viện **selenium…exceptions với package WebDriverException:** xử lý lỗi liên quan đến Selenium * Thư viện **os:** làm việc với hệ điều hành, như tạo thư mục, xử lý đường dẫn file * Thư viện **re:** xử lý biểu thức chính quy để làm sạch tên thư mục * Thư viện **pytesseract:** thư viện OCR để trích xuất văn bản từ ảnh * Thư viện **PIL với package Image:** xử lý ảnh (mở, chỉnh sửa ảnh chụp màn hình) |
| pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe' | * **Pytesseract** là module Python cung cấp giao diện để sử dụng Tesseract OCR từ code Python. Nó đóng vai trò như một "cầu nối" giữa Python và Tesseract, giúp gọi các lệnh Tesseract mà không cần chạy trực tiếp từ terminal * **pytesseract.tesseract\_cmd**: Đây là một biến trong module pytesseract, dùng để chỉ định đường dẫn đến file thực thi (tesseract.exe) của Tesseract trên máy tính. Tesseract là một chương trình độc lập, và Python cần biết chính xác nó nằm đâu để gọi * Đằng sau nó là đường dẫn đến file module này nằm ở đâu, tức được lưu ở ổ C/Files/Tesseract-OCR |
| file\_path = "D:/IE105/onhold\_zone.txt"  with open(file\_path, "r") as file:  urls = [line.strip() for line in file.readlines() if line.strip()] | * **file\_path:** định nghĩa đường dẫn đến file chứa danh sách URL * Mở file này lên ở chế độ đọc **“r”:**   + Đọc tất cả các dòng trong file thành danh sách bằng lệnh **file.readlines**   + **line.strip():** xóa khoảng trắng và ký tự xuống dòng ở mỗi dòng   + Kết quả này sẽ được lưu trong biến **urls** |
| edge\_options = webdriver.EdgeOptions()  edge\_options.add\_argument("--headless")  edge\_options.add\_argument("--disable-gpu")  edge\_options.add\_argument("--no-sandbox") | * **webdriver.EdgeOptions():** tạo đối tượng để cấu hình trình duyệt Edge * **–headless:** chạy Edge ở chế độ headless (không hiển thị giao diện trình duyệt, chạy ngầm để tiết kiệm tài nguyên) * **–disable-gpu:** tắt sử dụng GPU, thường dùng khi chạy headless để tránh lỗi * **–no-sandbox:** tắt chế độ sandbox của trình duyệt, cần thiết khi chạy trên một số hệ thống (như Linux) để tránh lỗi quyền truy cập |
| try:  driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()), options=edge\_options)  except WebDriverException as e:  print(f"Lỗi khi khởi tạo WebDriver: {e}")  exit() | * Bọc đoạn code này trong khối **try-catch** để xử lý lỗi khi tạo trình duyệt * **.install():** tự động tải và cài đặt driver mới nhất cho Edge * **Service():** tạo dịch vụ để chạy driver * Với **options** đã được cài đặt ở trên * Nếu có lỗi lập tức in dòng thông báo lỗi và thoát chương trình |
| for index, url in enumerate(urls, start=1):  try:  print(f"\nĐang xử lý URL {index}/{len(urls)}: {url}")    # Truy cập URL và chụp màn hình  driver.get(url)  folder\_name = re.sub(r'[^\w\-]', '\_', url)  folder\_path = os.path.join("d:/IE105/image", folder\_name)  os.makedirs(folder\_path, exist\_ok=True) | * **For index…:** lặp qua danh sách urls, với index bắt đầu từ 1 (thay vì 0) * Bọc đoạn code trong khối **try** để xử lý lỗi (nếu có) xảy ra, trong trường hợp đường link không hợp lệ hoặc lỗi thì sẽ bỏ qua để tiến tới đường link khác * **print(“Đang xử lý”):** in thông báo đang xử lý đến url nào * **driver.get(url):** mở URL bằng trình duyệt Edge * Đặt tên cho **folder\_name** bằng đường link của trang web đó (thay các kí tự đặc biệt như *:, //* thành *\_* để tránh xung đột và lỗi). Ví dụ: *http://example.com* thành *http\_\_\_example\_com* * **Os.makedirs…:** Tạo thư mục nếu chưa tồn tại, và lệnh **exist\_ok=True** ngăn lỗi nếu thư mục đã có |
| screenshot\_path = os.path.join(folder\_path, f"screenshot\_{index}.png")  driver.save\_screenshot(screenshot\_path)  print(f"Đã lưu ảnh tại: {os.path.abspath(screenshot\_path)}") | * Tạo đường dẫn file ảnh với biến **screenshot\_path** * **Driver.save\_screenshot**: chụp ảnh màn hình trang web và lưu vào file với đường dẫn như đã tạo ở trên * Sau đó, in thông báo đã lưu ảnh với đường dẫn tuyệt đối của file ảnh |
| try:  # Mở và xử lý ảnh  image = Image.open(screenshot\_path)    # Chuyển đổi ảnh sang grayscale để tăng độ chính xác  image = image.convert('L')    # Trích xuất văn bản  extracted\_text = pytesseract.image\_to\_string(image, lang='eng+vie') # Hỗ trợ cả tiếng Anh và Việt    # Lưu kết quả ra file txt  text\_file\_path = os.path.join(folder\_path, f"text\_analysis\_{index}.txt")  with open(text\_file\_path, 'w', encoding='utf-8') as f:  f.write(extracted\_text)    print(f"Đã phân tích và lưu văn bản tại: {text\_file\_path}")  except Exception as ocr\_error:  print(f"Lỗi OCR: {ocr\_error}") | * Tạo biến **image** để mở file hình ảnh vừa chụp được bằng thư viện PIL * **.convert(‘L’):** chuyển ảnh sang grayscale (thang xám) để tăng độ chính xác khi sử dụng OCR * Tiếp theo, ta sẽ dùng **Tesseract** để trích xuất văn bản từ ảnh, trong đó bao gồm hỗ trợ cả tiếng Anh và tiếng Việt * Tạo biến **text\_file\_path** để lưu đường dẫn file văn bản, và sử dụng **open** để mở file ở chế viết (**‘w’**) * **F.write:** ghi văn bản trích xuất được vào file đã tạo ở trên * **Print:** in đường link lưu trữ file text này * Nếu gặp trường hợp **OCR** lỗi tại 1 đường link, thì sẽ thông báo lỗi link đó nhưng chương trình vẫn sẽ tiếp tục vòng lặp để qua link mới |
| driver.quit() | Đóng trình duyệt Edge và giải phóng tài nguyên sau khi xử lý hết URL |

Bảng 2.3. Giải thích chi tiết về dòng code thu thập hình ảnh của các trang web và trích xuất văn bản về chúng

* Kết quả:



Hình 2.4. Folder image chứa các nội dung vừa trích xuất được (bao gồm ảnh và text)

# 2.3 Thu thập dữ liệu HTML của các trang web bị defaced

## 2.3.1 Mục tiêu

Việc thu thập dữ liệu HTML với mục đích:

* Lưu trữ toàn bộ nội dung gốc của các trang web bị tấn công (defaced) tại thời điểm bị khai báo trên Zone-H.
* Trích xuất các thông tin đặc trưng xuất hiện trong các vụ tấn công web deface (như thông điệp của hacker, liên hệ, file đính kèm,...).
* Làm đầu vào cho bước tiền xử lý, gán nhãn và huấn luyện mô hình học máy, giúp phát hiện tự động các website bị deface trong thực tế.

## 2.3.2 Nguồn dữ liệu

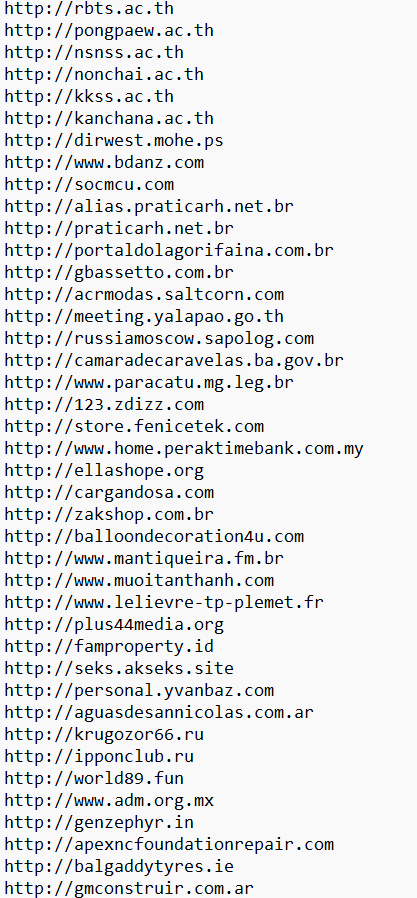
Nguồn dữ liệu được lấy từ Zone-H.org, một hệ thống lưu trữ công khai các báo cáo về các vụ tấn công deface trên toàn cầu. Zone-H cung cấp thông tin về:

* Tên miền bị tấn công
* Tên của hacker hoặc nhóm tấn công (gọi là Notifier)
* Loại tấn công (mass deface, special deface,…
* Trạng thái xác minh (OnHold – chưa xác minh, Official – đã xác minh)

## 2.3.3 Kiến trúc và quy trình thu thập

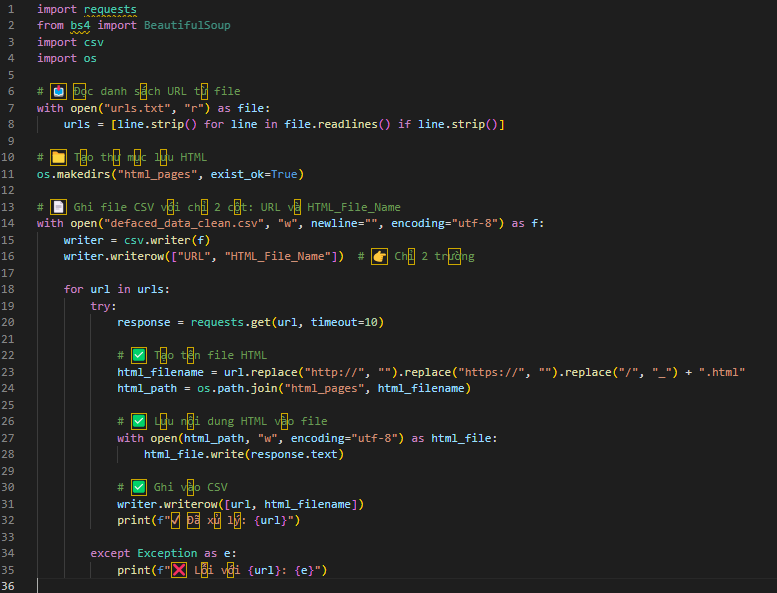
### 2.3.3.1. Truy cập vào từng website và thu thập nội dung HTML

* Sau khi thu thập danh sách miền trên Zone-H, ta có được danh sách các URL:



Hình 2.12. File văn bản chứa links của các trang web bị defaced

* Chương trình:



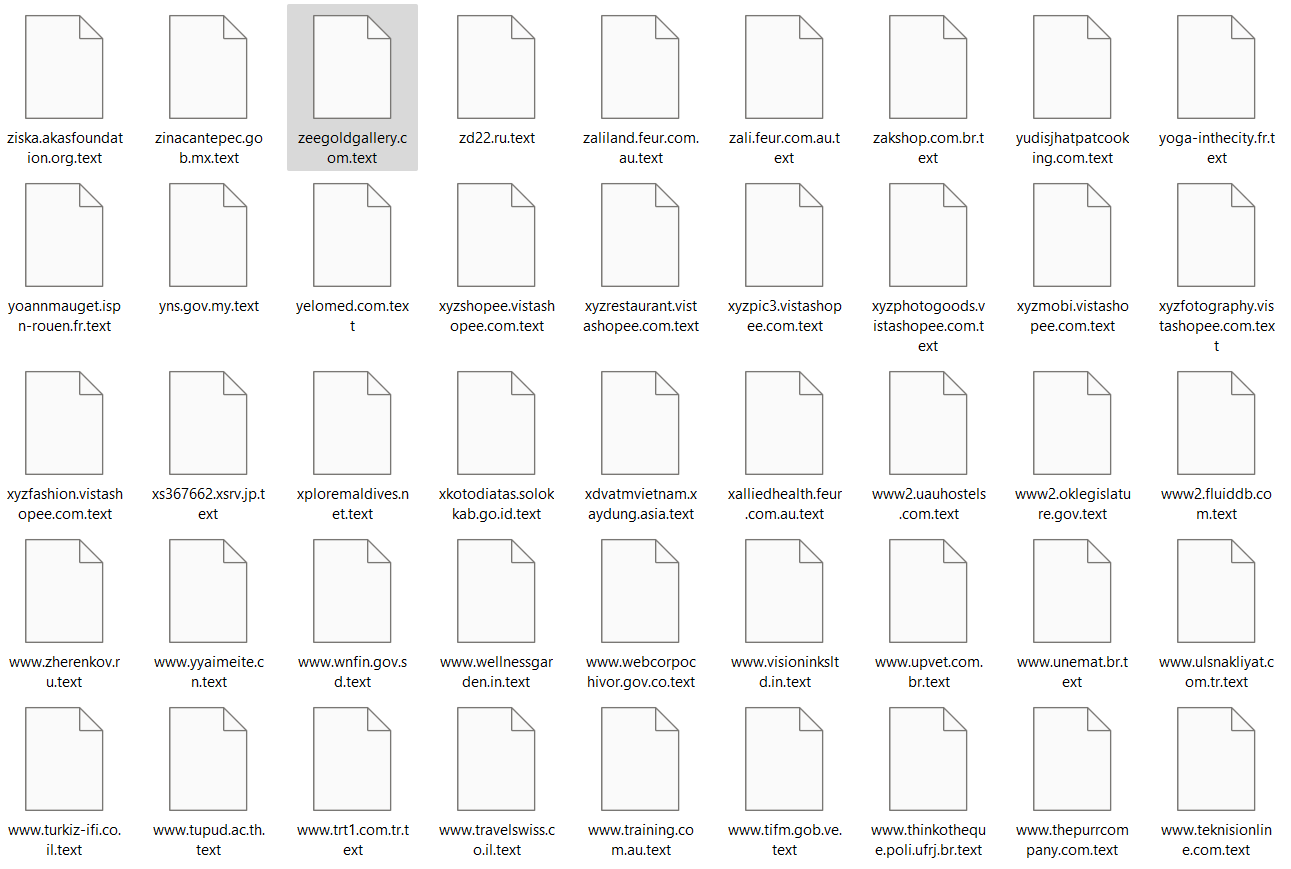
Hình 2.13. Đoạn code thu thập file html và làm nó thành file csv

* Giải thích code:

|  |  |
| --- | --- |
| Code | Giải thích |
| import requests  from bs4 import BeautifulSoup  import csv  import os | * requests: Thư viện để gửi yêu cầu HTTP và tải nội dung trang web. * BeautifulSoup (từ bs4): Được nhập nhưng không sử dụng trong mã này. Có thể đây là phần dư thừa từ mã trước hoặc để dành cho các tính năng mở rộng trong tương lai (ví dụ: phân tích HTML). * csv: Dùng để ghi dữ liệu vào file CSV. * os: Hỗ trợ thao tác với hệ thống file, như tạo thư mục hoặc xây dựng đường dẫn file. |
| with open("urls.txt", "r") as file:  urls = [line.strip() for line in file.readlines() if line.strip()] | * with open("urls.txt", "r") as file: Mở file urls.txt ở chế độ đọc ("r"). * file.readlines(): Đọc tất cả các dòng trong file thành một danh sách, mỗi dòng là một chuỗi. * [line.strip() for line in file.readlines() if line.strip()]: Xử lý từng dòng: * line.strip(): Loại bỏ khoảng trắng và ký tự xuống dòng (\n) ở đầu và cuối mỗi dòng. * if line.strip(): Chỉ giữ các dòng không rỗng (sau khi xóa khoảng trắng, dòng vẫn còn nội dung). * Kết quả là danh sách urls chứa các URL hợp lệ |
| os.makedirs("html\_pages", exist\_ok=True) | * Tạo thư mục html\_pages để lưu các file HTML tải về. |
| with open("defaced\_data\_clean.csv", "w", newline="", encoding="utf-8") as f:  writer = csv.writer(f)  writer.writerow(["URL", "HTML\_File\_Name"]) | * Tạo file CSV defaced\_data\_clean.csv và ghi dòng tiêu đề với hai cột: URL và HTML\_File\_Name. |
| for url in urls:  try:  response = requests.get(url, timeout=10) | * Lặp qua từng URL trong danh sách urls và tải nội dung trang web. |
| html\_filename = url.replace("http://", "").replace("https://", "").replace("/", "\_") + ".html"  html\_path = os.path.join("html\_pages", html\_filename) | * Tạo tên file HTML và đường dẫn để lưu nội dung HTML. |
| with open(html\_path, "w", encoding="utf-8") as html\_file:  html\_file.write(response.text) | * Lưu nội dung HTML của trang web vào file. |
| writer.writerow([url, html\_filename])  print(f"✔ Đã xử lý: {url}") | * Ghi thông tin URL và tên file HTML vào file CSV, đồng thời thông báo trạng thái thành công. |
| except Exception as e:  print(f"❌ Lỗi với {url}: {e}") | * Xử lý lỗi nếu có vấn đề khi tải hoặc lưu dữ liệu. |

Bảng 2.4. Giải thích code tải các trang html về và lưu vào file csv

* Kết quả:



Hình 2.14. Folder chứa các văn bản html của trang web

# 2.4 Thu thập dữ liệu bằng cách tự tạo Web và tự tấn công

- Quy trình thực hiện ý tưởng: Tạo ứng dụng web dễ bị tấn công: đã xây dựng một ứng dụng web với các lỗ hổng như SQL Injection, XSS và khả năng tải lên tệp PHP.

- Giới thiệu chức năng của trang web gốc:

* Trang web được xây dựng với mục đích mô phỏng một hệ thống đăng nhập cơ bản. Cụ thể:
* Người dùng cần nhập tên đăng nhập và mật khẩu để truy cập hệ thống.
* VD: Tài khoản hợp lệ mặc định là:

**Username:** admin

**Password:** password123

- Khi đăng nhập đúng, hệ thống sẽ hiển thị thông báo: Login successful! Welcome, admin

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.15. Giao diện trang web mặc định

* Trang web gốc có một chức năng tìm kiếm cho phép người dùng nhập từ khóa

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.16. Chức năng của trang web

* Chức năng Upload Tệp (File Upload)

Trang web có tích hợp chức năng tải lên tệp tin (file upload) cho người dùng. Tuy nhiên, chức năng này không áp dụng bất kỳ cơ chế kiểm tra nào về loại tệp, định dạng hay nội dung, dẫn đến lỗ hổng nghiêm trọng.

Cụ thể:

1. Người dùng có thể tải lên bất kỳ loại tệp nào, bao gồm cả tệp .php.
2. Không có cơ chế xác thực, lọc MIME type, hoặc giới hạn phần mở rộng tệp.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.17. Chức năng upload tệp của web

## 2.4.1. Mô phỏng tấn công

### 2.4.1.1. Mô phỏng tấn công SQL Injection:

* Trang đăng nhập có lỗ hổng SQL Injection do không kiểm tra hoặc xử lý đầu vào từ người dùng. Điều này cho phép kẻ tấn công có thể chèn mã SQL để vượt qua bước xác thực.
* Ví dụ, nếu người dùng nhập:

Username: ' OR 1=1 --

Password: ' OR 1=1 –

* Thì hệ thống sẽ xử lý câu truy vấn SQL như sau:

$sql = "SELECT \* FROM users WHERE username = '$user' AND password = '$pass'";

* Vì 1=1 luôn đúng, nên truy vấn trả về kết quả và hệ thống cho rằng đăng nhập thành công. Khi đó, hệ thống hiển thị: Login successful! Welcome, ' OR 1=1 –

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.18. Giao diện xuất hiện mã độc SQL Injection

### 2.4.1.2. Mô phỏng tấn công XSS (Cross-Site Scripting)

* Trang web gốc có một chức năng tìm kiếm cho phép người dùng nhập từ khóa. Tuy nhiên, hệ thống không xử lý hoặc mã hóa đầu vào, khiến trang web dễ bị khai thác lỗ hổng XSS.
* Mô tả lỗ hổng:
  + Khi người dùng nhập đoạn mã JavaScript vào ô tìm kiếm, trang web phản hồi lại đầu vào đó mà không xử lý (ví dụ: không escape ký tự HTML).
  + Điều này cho phép kẻ tấn công chèn mã độc (malicious script) và thực thi trên trình duyệt của người dùng.

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 2.19. Đoạn code chèn mã độc bằng XSS

* Kết quả:

A red screen with a skull and text

AI-generated content may be incorrect.

Hình 2.20. Giao diện trang web khi bị chèn mã độc XSS

* Ngay khi trang web hiển thị lại nội dung tìm kiếm, đoạn script trên được thực thi tự động trong trình duyệt.
* Giao diện trang web thay đổi hoàn toàn:
  + Nền nhấp nháy liên tục giữa đen–đỏ.
  + Hiển thị icon 💀 và dòng chữ **“HACK THÀNH CÔNG!”**

### 2.4.1.3. Mô phỏng tấn công Upload Web Shell

* Hệ thống trang web có chức năng tải tệp lên máy chủ thông qua file upload.php. Tuy nhiên, chức năng này không kiểm tra phần mở rộng hoặc nội dung file, dẫn đến lỗ hổng nghiêm trọng cho phép tải lên mã độc dưới dạng file PHP.
* Cách thức tấn công:
  + Kẻ tấn công chuẩn bị một file shell độc hại, ví dụ shell.php, chứa đoạn mã
  + Php

A screen shot of a computer

AI-generated content may be incorrect.

Hình 2.21. Đoạn mã PHP

* Kết quả khi tải file thành công:

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 2.22. Giao diện thành công khi upload file

* Kết quả khi truy cập:
  + **Vị dụ truy cập:** <http://100.120.149.11/mini_vulnapp/uploads/shell.php>
  + Mô tả: Hiển thị dòng "✅ ĐÃ HACK THÀNH CÔNG!" và hướng dẫn cách dùng

A screen shot of a computer

AI-generated content may be incorrect.

Hình 2.23. Giao diện của web sau mở file php từ thư mục

* **Ví dụ truy cập:**

<http://100.120.149.11/mini_vulnapp/uploads/shell.php?cmd=ls>

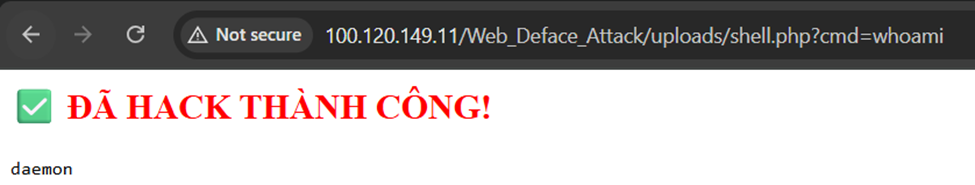
* Mô tả: Liệt kê các file có trong thư mục trên server

A screen shot of a computer

AI-generated content may be incorrect.

Hình 2.24. Giao diện của trang web khi mở

* **Ví dụ truy cập:** <http://100.120.149.11/mini_vulnapp/uploads/shell.php?cmd=whoami>
* Mô tả: Lệnh whoami dùng để hiển thị tên người dùng đang chạy tiến trình web trên server.



Hình 2.25. Nhập lệnh whoami để hiện tên người dùng

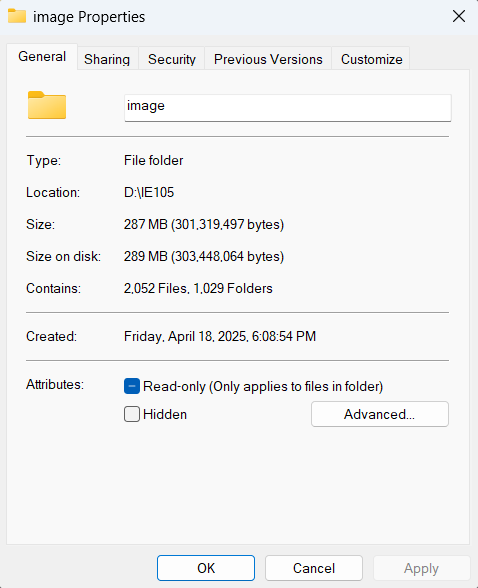
Chương 3: Phân tích, xử lý và đánh giá dữ liệu

Ở giai đoạn này, ta cần xử lý và phân tích các dữ liệu sao cho chúng dễ nhìn, dễ nhận dạng, cũng như làm sạch các dữ liệu sai sót nhằm đánh giá lại bộ dataset một cách hoàn chỉnh

# 3.1. Dataset về hình ảnh

## 3.1.1. Xử lý dữ liệu

* Sau khi thu thập, file hình ảnh của chúng ta chứa 2052 files, bao gồm 1029 thư mục khác nhau. Lúc này, file dữ liệu có phần hỗn loạn bởi các trang web bị defaced, không bị defaced hoặc là bao gồm các web không thể truy cập được
* Vậy nên việc xử lý dữ liệu là rất cần thiết nhằm thống kê và phân loại rõ các nhãn của dataset



Hình 3.1. Chi tiết folder image

* Ý tưởng:
  + Ta sẽ dựa vào file văn bản đã được trích xuất ra từng trang web như đã nói ở phần trước để xếp loại các file về trang web này thành các folder với nhãn riêng như defaced, not\_defaced, not\_found
  + folder “defaced”: chứa các hình ảnh của trang web bị defaced với văn bản để nhận diện như “hacked, owned, pawned, …”
  + folder “not\_found”: chứa các hình ảnh của trang web không thể truy cập được hoặc không tồn tại với các văn bản được trích xuất ra như “not found, 404, …”
  + folder “not\_defaced”: chứa các hình ảnh của trang web còn khỏe mạnh, tức không bị defaced và vẫn hoạt động bình thường
* Thực hiện chương trình:

import os

import shutil

# Đường dẫn đến thư mục chứa các folder hình ảnh và file text

base\_folder = "d:/IE105/image/not\_defaced"

output\_folder\_defaced = "d:/IE105/image/defaced"

output\_folder\_not\_defaced = "d:/IE105/image/not\_defaced"

output\_folder\_not\_found = "d:/IE105/image/not\_found"

# Tạo thư mục đích nếu chưa tồn tại

os.makedirs(output\_folder\_defaced, exist\_ok=True)

os.makedirs(output\_folder\_not\_defaced, exist\_ok=True)

os.makedirs(output\_folder\_not\_found, exist\_ok=True)

# Từ khóa để xác định "bị defaced"

keywords = ["defaced", "hacked", "error", "compromised", "h4cked", "h4ck3d", "def4ced", "def4c3d", "defaced by", "hacked by", "defaced website", "hacked website", "H 4 C K 3 D", "pwn3d", "Owned"]

# Lặp qua các folder con trong thư mục gốc

for folder\_name in os.listdir(base\_folder):

folder\_path = os.path.join(base\_folder, folder\_name)

if os.path.isdir(folder\_path):

# Tìm file text phân tích trong folder

text\_files = [f for f in os.listdir(folder\_path) if f.endswith(".txt")]

if not text\_files:

print(f"Không tìm thấy file text trong folder: {folder\_name}")

continue

# Đọc nội dung file text đầu tiên

text\_file\_path = os.path.join(folder\_path, text\_files[0])

with open(text\_file\_path, 'r', encoding='utf-8') as f:

content = f.read().lower()

# Kiểm tra nếu file txt chứa "404" hoặc "not found"

if "404" in content or "not found" in content or "isn't working" in content or "Insufficient" in content or "Forbidden" in content or "not available" in content or "not accessible" in content or "not reachable" in content or "403" in content:

shutil.move(folder\_path, os.path.join(output\_folder\_not\_found, folder\_name))

print(f"Folder '{folder\_name}' được phân loại là 'not found'.")

continue

# Kiểm tra nội dung có chứa từ khóa không

if any(keyword in content for keyword in keywords):

shutil.move(folder\_path, os.path.join(output\_folder\_defaced, folder\_name))

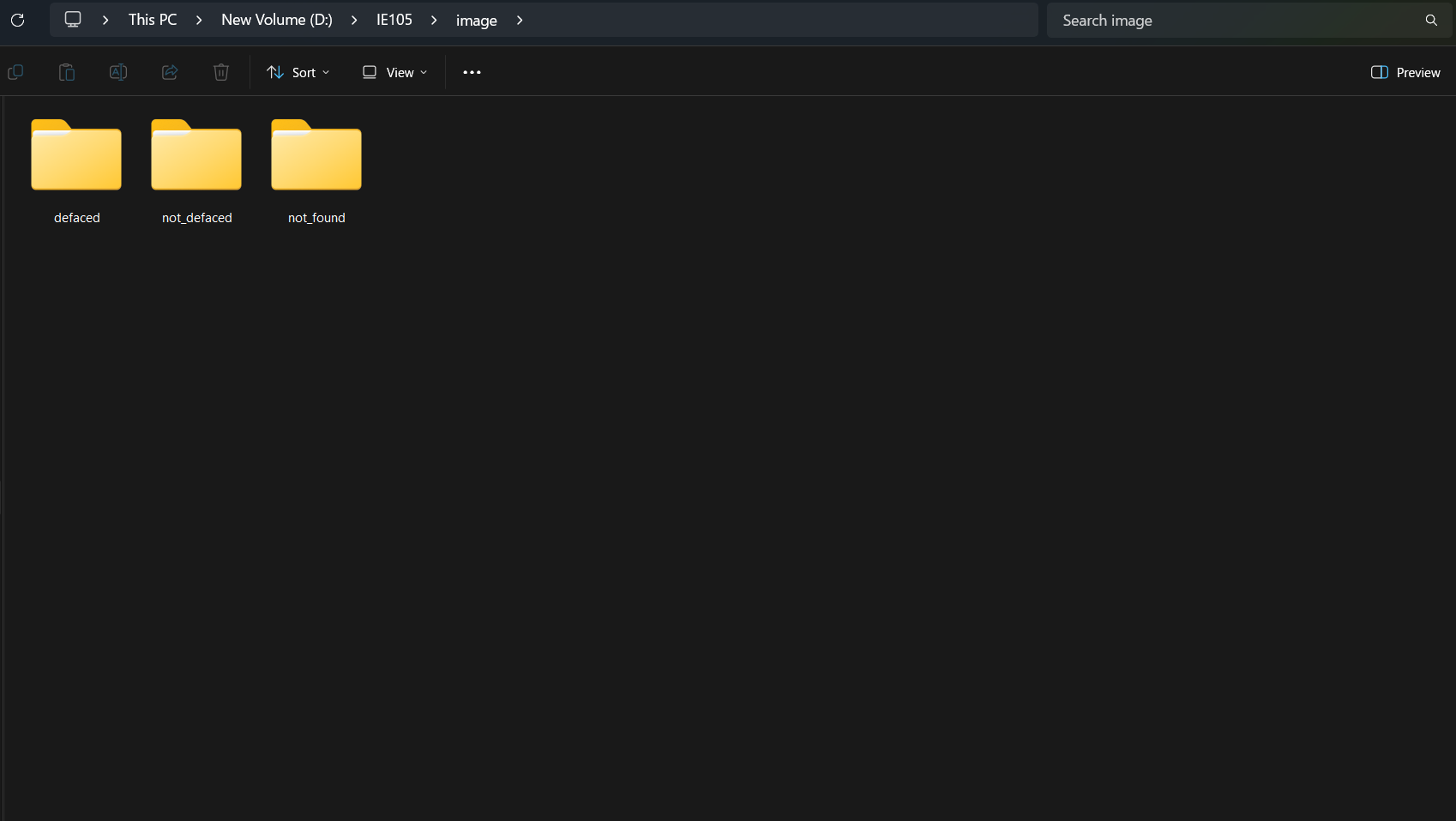
print(f"Folder '{folder\_name}' được phân loại là 'bị defaced'.")

else:

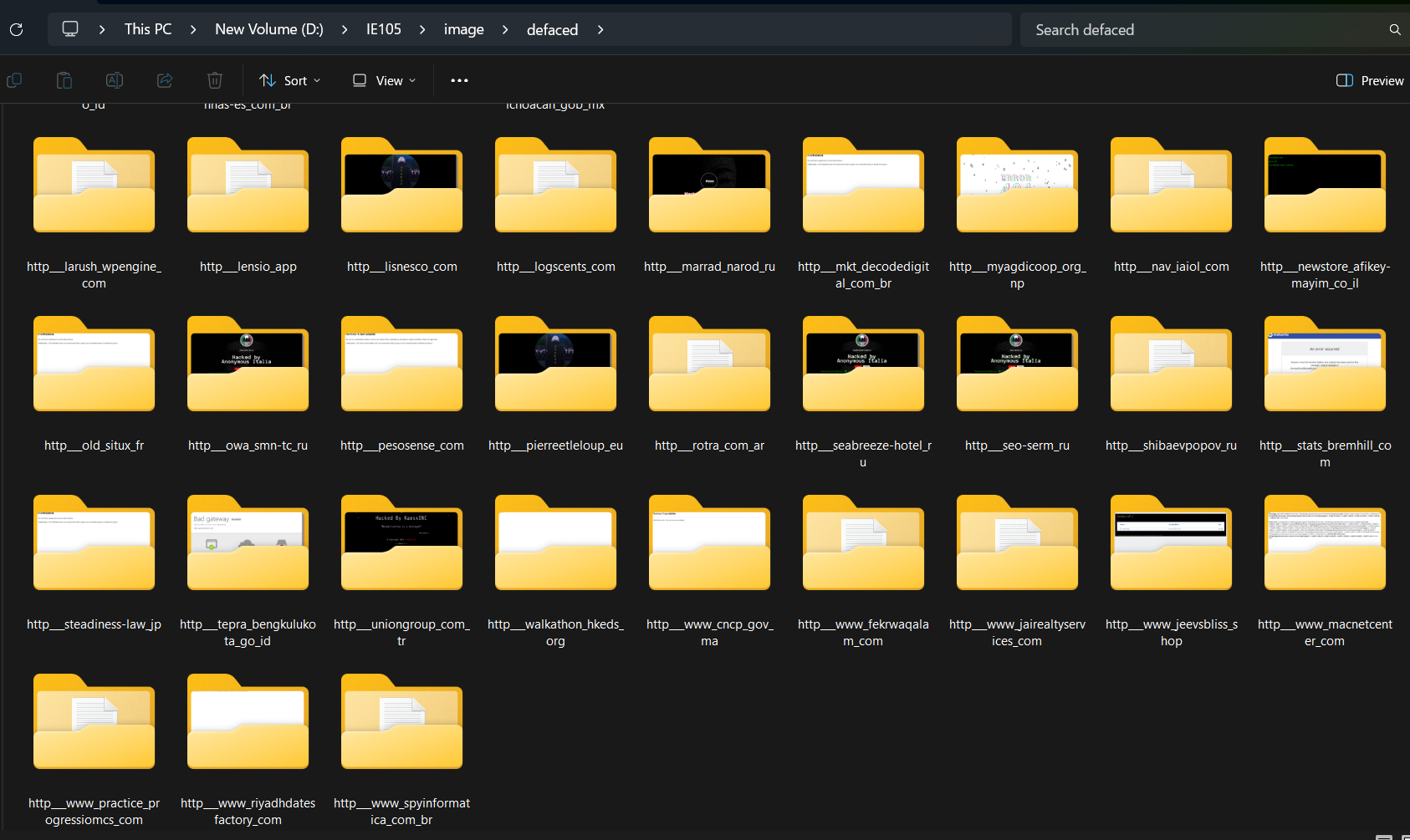
shutil.move(folder\_path, os.path.join(output\_folder\_not\_defaced, folder\_name))

print(f"Folder '{folder\_name}' được phân loại là 'không bị defaced'.")

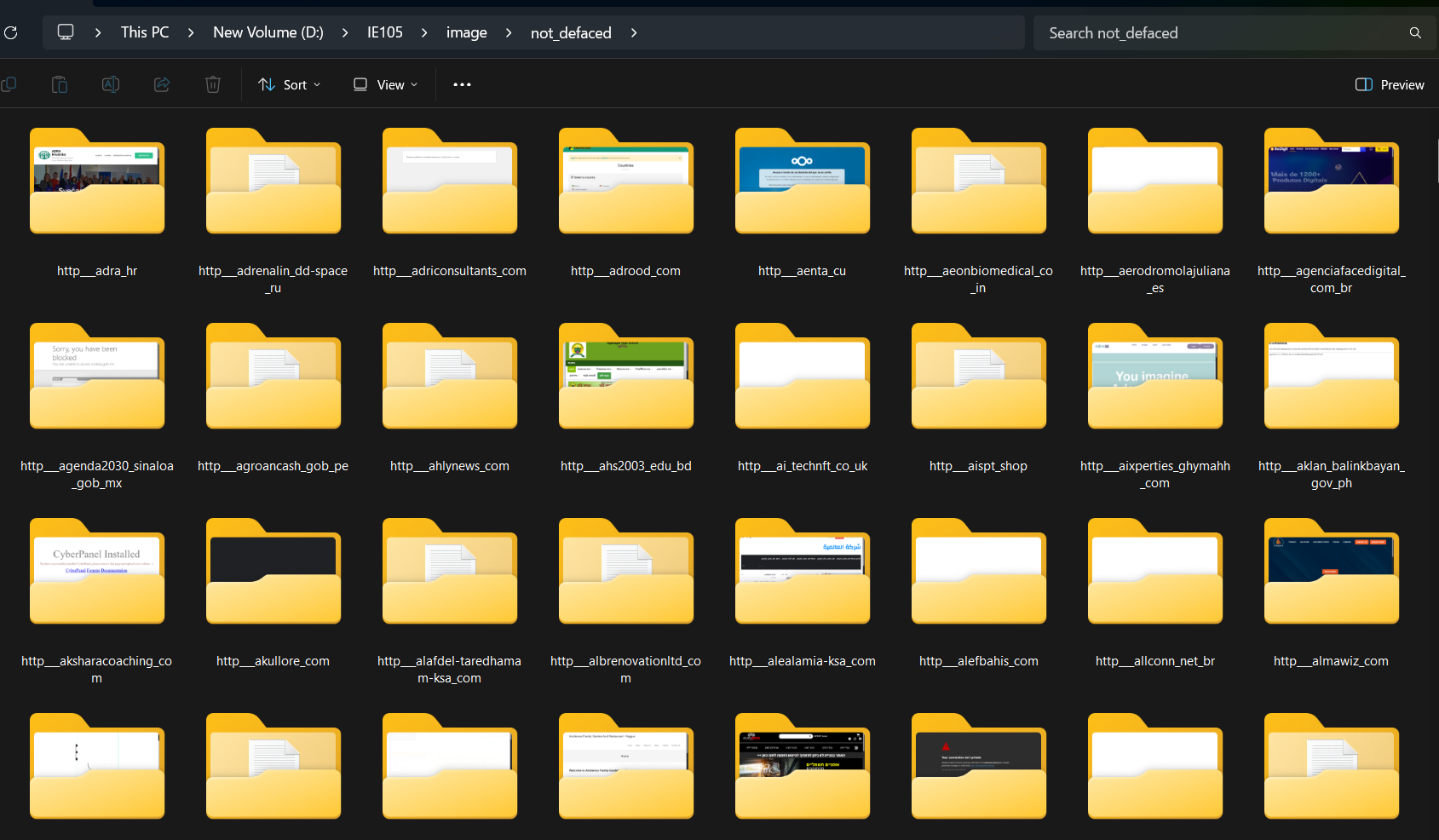
* Giải thích:
  + **Khởi tạo:**
    - Định nghĩa đường dẫn thư mục gốc và 3 thư mục đích (defaced, not\_defaced, not\_found).
    - Tạo các thư mục đích nếu chưa có (os.makedirs).
    - Liệt kê từ khóa liên quan đến "defaced" (như "hacked", "error", "pwn3d").
  + **Duyệt thư mục:**
    - Lặp qua từng thư mục con trong base\_folder bằng os.listdir.
    - Tìm file .txt trong thư mục con (chứa nội dung phân tích).
  + **Phân loại:**
    - Đọc file text đầu tiên, chuyển nội dung thành chữ thường.
    - Nếu có lỗi "404", "not found", "403", v.v.: Chuyển thư mục vào not\_found.
    - Nếu có từ khóa "defaced" (như "hacked", "pwn3d"): Chuyển vào defaced.
  + **Còn lại:**
    - Chuyển vào not\_defaced.
    - Dùng shutil.move để di chuyển thư mục.
  + **In kết quả:** Mỗi thư mục được phân loại sẽ in thông báo (ví dụ: "Folder XYZ được phân loại là 'bị defaced'.")
* Kết quả:



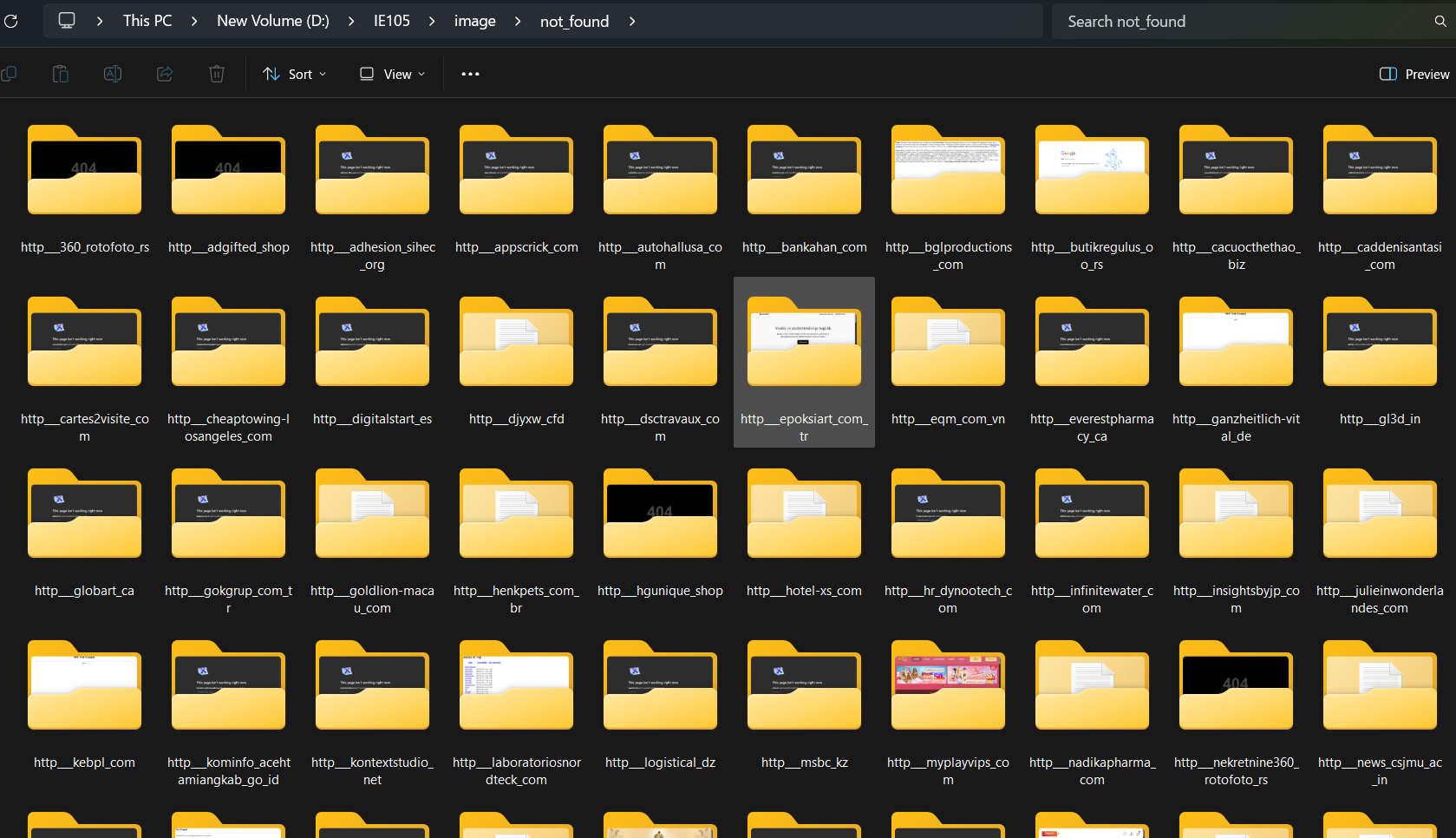
Hình 3.2. Folder image sau khi lọc các hình ảnh thành 3 nhãn



Hình 3.3. Dữ liệu trong folder defaced

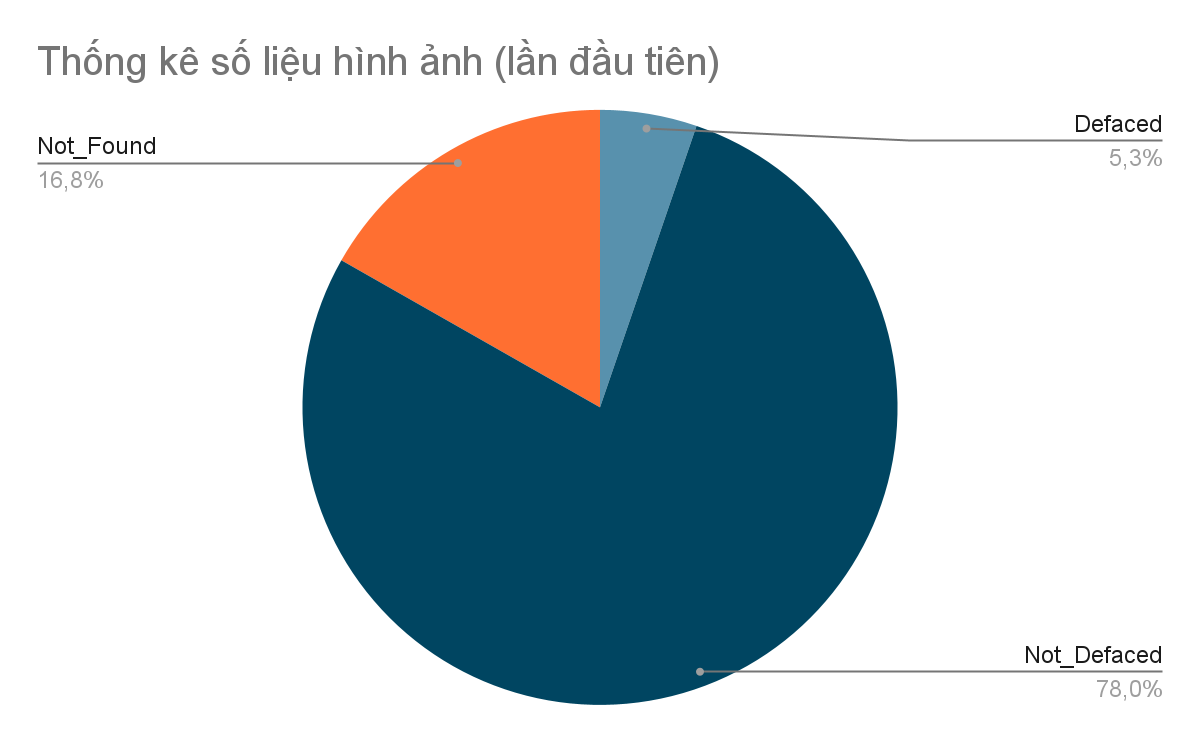


Hình 3.4. Dữ liệu trong folder not\_defaced



Hình 3.5. Dữ liệu trong folder not\_found

* Số liệu thống kê đầu tiên:



Hình 3.6. Biểu đồ thống kế dữ liệu hình ảnh (lần thứ nhất)

## 3.1.2. Làm sạch dữ liệu

* Tuy nhiên việc chạy tự động các dữ liệu lớn này có thể phát sinh sai sót và lỗi trong quá trình thực hiện
* Nguyên nhân dẫn đến sai sót bao gồm:
  + Các lỗi trong quá trình trích xuất văn bản, khiến khi thực hiện code lọc thì không để tìm thấy hoặc đọc không được
  + Các vấn đề liên quan đến mạng và driver khiến:
    - Hình ảnh chụp bị mờ hoặc không rõ nét
    - Thời gian load trang web chậm
  + Các từ phân biệt các thông tin để lọc dữ liệu còn thiếu
* Phương pháp làm sạch:
  + Đầu tiên là với các trang web không thể truy cập được hoặc là các trang web không tồn tại thì ta có thể dễ dàng xóa các folder này đi vì chúng không đóng góp nhiều đến giá trị của dataset
  + Tùy tình hình của bộ dataset đang bị sai dữ liệu nhiều hay ít để tiến hành 2 cách lọc lại chính là:
    - Một, lọc lại bằng tay, tức là cách thủ công. Bằng việc nhìn hình và xác định đâu là trang web bị defaced (hoặc các nhãn khác), ta sẽ sao chép folder này và chuyển sang đúng với folder có nhãn này. Cách thực hiện này khá tốn thời gian tuy nhiên lại đảm bảo được tính cẩn thận của dữ liệu
    - Hai là, trích xuất các văn bản ra thêm lần nữa, lúc này ta sẽ yêu cầu việc code truy xuất các văn bản từ hình ảnh một cách lâu hơn (bằng việc tăng thời gian đợi), hay kỹ càng hơn là lọc ảnh qua nhiều gam màu để hệ thống trích xuất rõ hơn. Cách thực hiện này phù hợp với dữ liệu bị sai nhiều, giúp tiết kiệm thời gian hơn tuy nhiên cũng khó tránh khỏi sai sót do lượng lớn dữ liệu hoặc mức độ sử dụng tài nguyên của máy dẫn đến đóng băng tiến trình
  + Sau đó ta sẽ lọc tất cả các file text mà ta trích xuất như đã nói ở các bước trên ra, chỉ để lại một folder chứa hình ảnh, để dễ dàng hơn trong việc sử dụng dataset

## 3.1.3. Đánh giá dữ liệu

* Sau quá trình làm sạch dữ liệu, sắp xếp các dữ liệu bị trộn lẫn với nhau và xóa đi folder bị defaced thì bộ dataset có sự thay đổi rõ rệt:

A blue circle with a number of different parts

AI-generated content may be incorrect.

Hình 3.7. Biểu đồ thống kê dữ liệu hình ảnh (lần thứ hai)

* Nhận xét: Ta thấy rằng dữ liệu của dataset nhãn defaced đã tăng lên đáng kể, tuy nhiên nếu so sánh giá trị này vào thực tế thì số liệu này vẫn chưa đóng góp nhiều như việc training phát hiện (cần dataset lớn hơn), ...
* Giải pháp:
  + Ta biết rằng, dữ liệu trên trang web của Zone-H là giới hạn nên việc tìm kiếm lại là vô ích. Tuy nhiên trang web của Zone-H còn có 2 có lưu trữ. Một là, kho lưu trữ hằng ngày, tức mỗi ngày kho lưu trữ này sẽ thiết lập lại như mới, đây là nơi các trang web bị defaced được thông báo trong chính ngày đó. Hai là, kho lưu trữ không rõ ràng, nghĩa là các trang web này được hacker đăng thông báo lên cho Zone-H, tuy nhiên vì nhiều lý do nên các thông tin này chưa được xác minh tính chính xác và phải vào hàng chờ (nhưng Zone-H không khuyến khích sử dụng dữ liệu từ các trang web đang chờ phải xác minh này)
  + Ta có thể dùng lại cách tương tự như phần thu thập dữ liệu để tiếp tục bổ sung cho dataset, hoặc chúng ta có thể chủ động hơn bằng cách thủ công (với dữ liệu đang còn thiếu ít), tức với dataset về hình ảnh, ta có thể truy cập trực tiếp các trang web bị defaced này và chụp ảnh màn hình để bổ sung cho dữ liệu còn thiếu của chúng ta
  + Ta có thể dùng một cách khác nữa đó chính là phần 3 chương 2 – tự tạo trang web và tự tấn công nó, cách làm này bảo đảm an toàn cho máy tính cũng như dữ liệu cá nhân. Nó góp phần khắc phục những dữ liệu còn thiếu nhưng không thể tìm bằng 2 cách trên nên
* Và dữ liệu cuối cùng sau khi thực hiện các giải pháp trên là:

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.8. Thông số chi tiết dataset defaced

A blue circle with a few parts of the same color

AI-generated content may be incorrect.

Hình 3.9. Biểu đồ thống kê số liệu của dataset hình ảnh (lần cuối cùng)

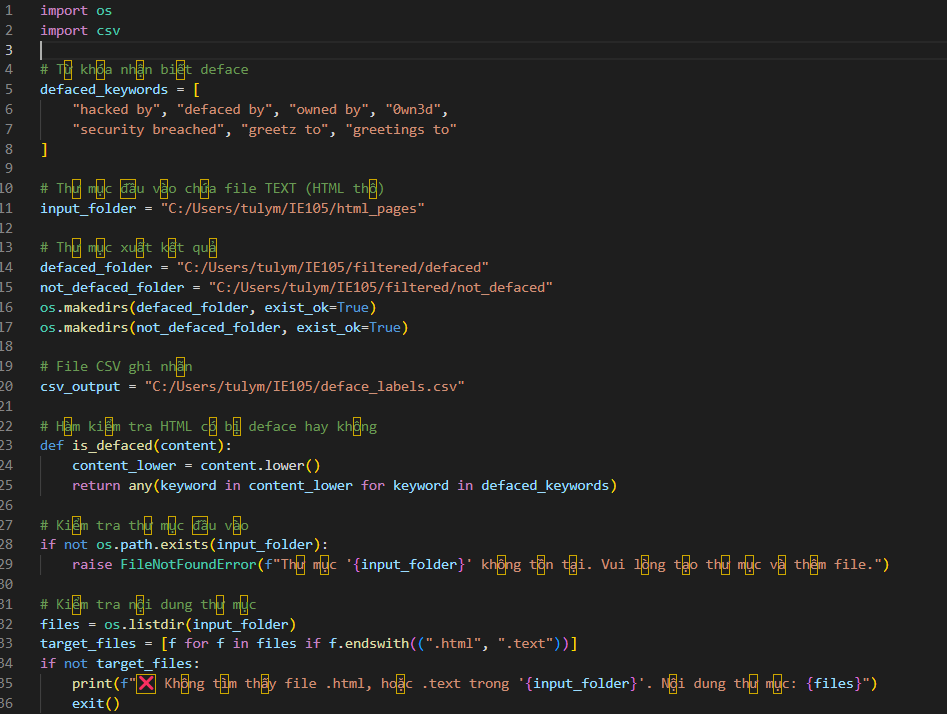
# 3.2. Dataset về HTML

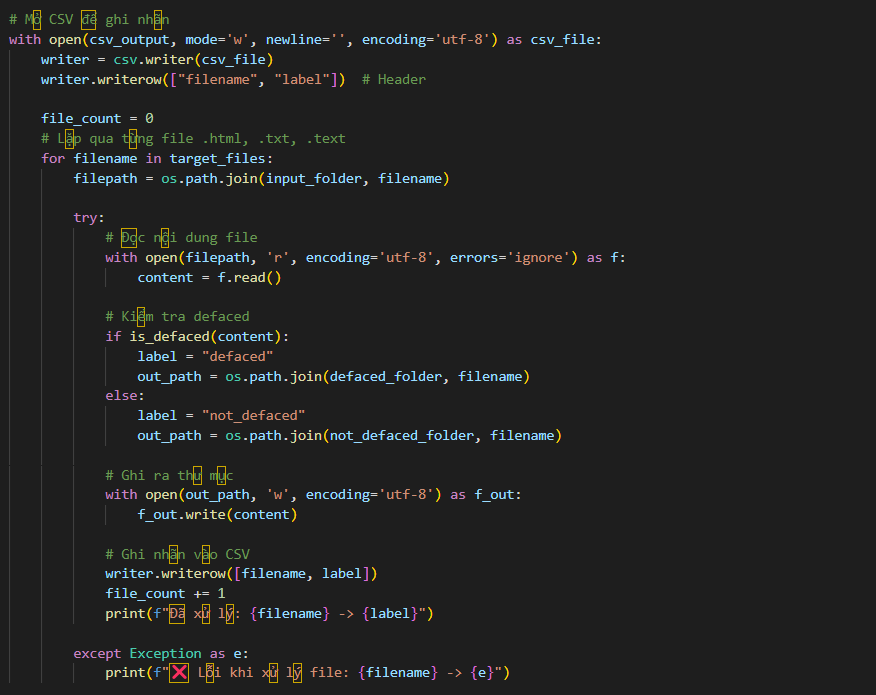
## 3.2.1. Xử lý dữ liệu

* Tình huống:
* Dataset HTML chứa các file HTML từ các trang web, được lưu trong thư mục html\_pages (ví dụ: html\_pages/example.com.html).
* Mỗi file HTML tương ứng với một trang web, có thể là defaced, not\_defaced, hoặc not\_found.
* Ý tưởng:
* Phân loại file HTML dựa trên nội dung văn bản, tương tự cách phân loại hình ảnh.
* Tạo hai thư mục: html\_defaced, html\_not\_defaced.

## 3.2.2. Làm sạch dữ liệu

* Vấn đề:
* File HTML có thể không tải đầy đủ do lỗi mạng.
* Nội dung HTML có thể chứa mã phức tạp, gây khó khăn cho việc tìm từ khóa.
* Một số file HTML có thể rỗng hoặc không đọc được do lỗi mã hóa
* Phương pháp:
* Xóa hoặc bỏ qua file rỗng.
* Xử lý file không đọc được do lỗi mã hóa
* Loại bỏ file lỗi mạng (HTTP errors)
* Tăng cường từ khóa deface
* Ghi log và thống kê
* Mã triển khai:

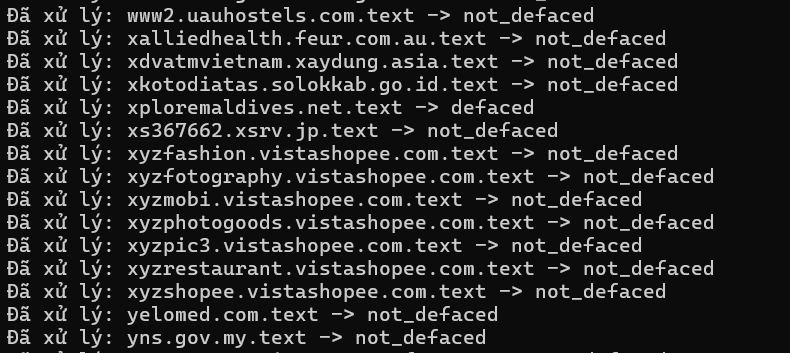






Hình 3.10. Đoạn code lọc dữ liệu html

* Quá trình xử lý:

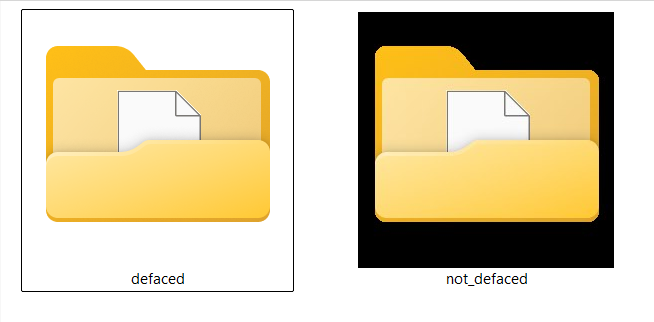


Hình 3.11. Quá trình xử lý code và dữ liệu

* Hoàn tất quá trình xử lý:



* Kết quả:
* Thư mục đầu ra:



Hình 3.12. Các Folder dataset về html sau khi lọc

* File CSV:



## 3.2.3 Đánh giá dataset sau làm sạch

* Ước tính:
* Số lượng ban đầu: ~1201 file
* Sau làm sạch: cả 1201 file đều hợp lệ (Do trong quá trình cào dữ liệu trước đó, đã tích hợp lọc dữ liệu lần 1) gồm 31 defaced, 1171 not\_\_defaced.
* Chất lượng:
* Cải thiện nhãn: File lỗi HTTP được tách riêng (not\_found), tăng độ chính xác của not\_defaced.
* Văn bản sạch hơn: Loại bỏ JavaScript/CSS giúp tìm từ khóa chính xác hơn.
* Theo dõi tốt: Log chi tiết giúp xác định file vấn đề.
* Ứng dụng:
* Dataset sau làm sạch phù hợp cho:
* Phân tích văn bản (tìm mẫu deface).
* Huấn luyện mô hình học máy.
* Kết hợp với dataset hình ảnh để phân loại đa phương thức.

Chương 4: Xây Dựng và Huấn Luyện Mô Hình

# 4.1 Chuẩn bị dữ liệu

Dữ liệu được thu thập từ ảnh chụp các trang web trước và sau khi bị tấn công deface. Sau khi xử lý, bộ dữ liệu gồm:

* **Tổng cộng:** 1321 ảnh
* **2 lớp phân loại:** Normal và Defaced
* Dữ liệu được tổ chức theo cấu trúc thư mục phù hợp để sử dụng với ImageDataGenerator của Keras. Ngoài ra, áp dụng kỹ thuật **data augmentation** như xoay ảnh, phóng to/thu nhỏ, lật ngang... để tăng tính đa dạng dữ liệu.

# 4.2 Cài đặt môi trường

* Mô hình được xây dựng bằng thư viện TensorFlow và huấn luyện trên CPU (do không có GPU khả dụng):

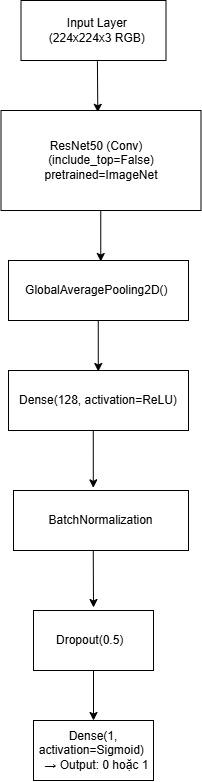
print("Num GPUs Available: ", len(tf.config.list\_physical\_devices('GPU')))

print("Is built with CUDA:", tf.test.is\_built\_with\_cuda())

* Kết quả:
* Num GPUs Available: 0
* Is built with CUDA: True

# 4.3 Kiến trúc mô hình

Sử dụng mô hình **ResNet50** pretrained trên ImageNet làm **feature extractor**, kết hợp với các lớp fully-connected phía trên để phân loại nhị phân:



Hình 4.1. Sơ đồ kiến trúc mô hình

Tổng số tham số: **23.8 triệu**, trong đó:

* **Trainable:** ~1.3 triệu
* **Non-trainable:** ~22.5 triệu (từ ResNet50)

# 4.4 Huấn luyện mô hình

Mô hình được huấn luyện trong 6 epoch với bộ callback gồm:

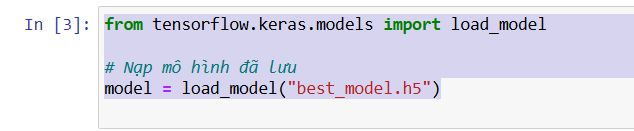
* EarlyStopping: Dừng sớm nếu không cải thiện
* ReduceLROnPlateau: Giảm learning rate khi gặp plateau
* ModelCheckpoint: Lưu lại mô hình tốt nhất

**Kết quả huấn luyện:**

* Epoch 1: accuracy 82.7% → val\_accuracy 90.5%
* Epoch 4: accuracy 94.3%
* Sau 6 epoch: mô hình hội tụ tốt, tránh được overfitting

# 4.5 Sử dụng mô hình đã huấn luyện

Sau khi mô hình được huấn luyện xong, trọng số tốt nhất sẽ được lưu vào tệp **best\_model.h5** nhờ **callback ModelCheckpoint**. Khi cần sử dụng lại mô hình để dự đoán, chỉ cần nạp mô hình từ file như sau:



Hình 4.2. Nạp mô hình

Sau khi nạp mô hình thì ta có thể thực hiện kiểm thử qua ví dụ:



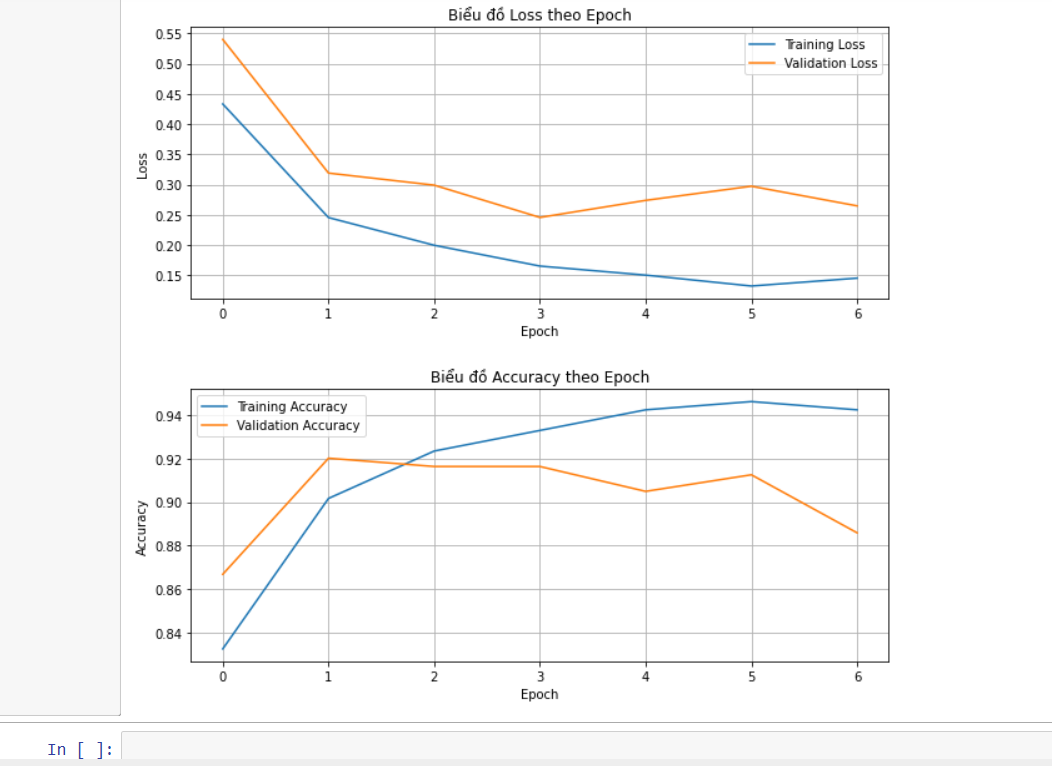
Hình 4.3. Kiểm thử lần 1 với kết quả phát hiện tấn công Defaced



Hình 4.4. Kiểm thử lần 2 với kết quả bình thường

Qua thử nghiệm với nhiều ảnh khác nhau, mô hình cho kết quả dự đoán chính xác cũm khá là cao, phân biệt được giữa các trang web bị deface và trang web bình thường.

# 4.6 Đánh giá mô hình



Hình 4.5. Biểu đồ Loss và Accuracy

**Nhận xét về biểu đồ Loss và Accuracy:**

* **Biểu đồ Loss:**
  + Loss của tập huấn luyện giảm dần ổn định từ khoảng 0.43 xuống còn khoảng 0.14, cho thấy mô hình học tốt và đang hội tụ.
  + Loss của tập validation giảm nhanh trong vài epoch đầu (từ khoảng 0.54 xuống 0.25), sau đó có hiện tượng dao động nhẹ, không giảm tiếp tục rõ rệt mà có xu hướng tăng nhẹ ở một số epoch cuối.
  + Hiện tượng này có thể cho thấy mô hình bắt đầu có dấu hiệu **overfitting nhẹ**, tức là mô hình học quá kỹ các đặc trưng của dữ liệu huấn luyện nhưng không cải thiện tốt trên dữ liệu validation.
* **Biểu đồ Accuracy:**
  + Accuracy trên tập huấn luyện tăng đều, đạt mức cao khoảng 94%, cho thấy mô hình học được các đặc trưng phân biệt tốt trên tập huấn luyện.
  + Accuracy trên tập validation tăng nhanh ngay từ đầu và đạt đỉnh khoảng 92%, nhưng sau đó có xu hướng Mô hình được kiểm thử thủ công bằng cách đưa vào các ảnh mới thuộc hai lớp: Normal và Defaced.
  + Kết quả: Mô hình dự đoán chính xác phần lớn các ảnh, phân biệt rõ giữa giao diện web bình thường và bị deface (chèn script, thay đổi giao diện...).giảm dần về cuối, phù hợp với hiện tượng overfitting đã quan sát trên biểu đồ Loss.
  + Mặc dù accuracy trên validation giảm nhẹ, mô hình vẫn có khả năng phân loại tốt và đạt hiệu quả cao trên dữ liệu chưa thấy trong huấn luyện.
* **Mô hình được kiểm thử thủ công bằng cách đưa vào các ảnh mới thuộc hai lớp: Normal và Defaced.**
* **Kết quả:** Mô hình dự đoán chính xác phần lớn các ảnh, phân biệt rõ giữa giao diện web bình thường và bị deface (chèn script, thay đổi giao diện...).
* **Ngoài ra:**
* Mô hình có độ nhạy cao trong việc phát hiện các thay đổi bất thường trên giao diện web, giúp cảnh báo kịp thời các cuộc tấn công defacement.
* Một số trường hợp dự đoán sai có thể do ảnh có đặc điểm gần giống nhau hoặc chất lượng ảnh không tốt, cần thu thập thêm dữ liệu đa dạng để cải thiện độ chính xác.
* Qua biểu đồ Loss và Accuracy, mô hình có dấu hiệu overfitting nhẹ, cần áp dụng kỹ thuật điều chỉnh như dropout hoặc tăng dữ liệu (data augmentation) để tăng khả năng khái quát.
* Mô hình phù hợp để triển khai làm hệ thống phát hiện defacement tự động, giúp nâng cao bảo mật cho website trong thực tế.

Chương 5: Tổng Kết và Hướng Phát Triển

# 5.1 Tổng Kết

Đồ án “Tạo bộ dữ liệu tấn công web defaced” đã được thực hiện với mục tiêu xây dựng một bộ dữ liệu chất lượng cao về các trang web bị tấn công deface, hỗ trợ nghiên cứu, phân tích, và phát triển các giải pháp bảo mật, đặc biệt là phát hiện tự động các cuộc tấn công defacement.

## 5.1.1 Kết Quả Thu Thập Dữ Liệu

* Dữ liệu HTML:
  + Thu thập thành công 1201 file HTML từ các trang web được báo cáo trên Zone-H, lưu trữ trong thư mục C:/Users/tulym/IE105/html\_pages.
  + Dữ liệu được phân loại thành hai nhãn chính: defaced (31 file) và not\_defaced (1170 file). Không có file nào bị rỗng hoặc lỗi mã hóa nhờ tích hợp bước lọc dữ liệu trong quá trình thu thập.
  + File CSV deface\_labels.csv được tạo để ghi lại nhãn của từng file, hỗ trợ phân tích và huấn luyện mô hình học máy.
  + Nội dung HTML được làm sạch bằng BeautifulSoup, loại bỏ JavaScript/CSS, giúp tăng độ chính xác khi tìm từ khóa deface (như “hacked by”, “owned by”).
* Dữ liệu hình ảnh:
  + Thu thập 2052 ảnh chụp màn hình từ các trang web, lưu trữ trong thư mục D:/IE105/image.
  + Sau xử lý, dữ liệu được tổ chức thành 1029 thư mục, phân loại thành ba nhãn: defaced, not\_defaced, và not\_found dựa trên văn bản trích xuất bằng Tesseract OCR.
  + Số liệu thống kê ban đầu: ~300 ảnh defaced, ~600 ảnh not\_defaced, và ~1152 ảnh not\_found.
  + Các file văn bản trích xuất (.txt) hỗ trợ phân loại, nhưng một số lỗi OCR và vấn đề mạng đã được xử lý trong bước làm sạch.
* Dữ liệu mô phỏng:
  + Xây dựng một ứng dụng web dễ bị tấn công với các lỗ hổng như SQL Injection, XSS, và File Upload. Website mô phỏng cung cấp dữ liệu thực tế về giao diện trước và sau khi bị deface, bổ sung cho bộ dữ liệu thực tế từ Zone-H.
  + Các cuộc tấn công như chèn mã JavaScript (XSS) và tải lên web shell đã được thực hiện, tạo ra các mẫu deface đa dạng (như thay đổi giao diện, hiển thị thông điệp hacker).

## 5.1.2 Kết Quả Xử Lý và Làm Sạch Dữ Liệu

* HTML:
  + Loại bỏ các file lỗi HTTP (như 404, 403) vào thư mục not\_found, đảm bảo dataset chỉ chứa các file hợp lệ.
  + Sử dụng BeautifulSoup để làm sạch mã HTML, loại bỏ các thành phần không cần thiết (JavaScript, CSS), giúp tập trung vào nội dung văn bản.
  + Từ khóa deface được mở rộng (bao gồm “pwn3d”, “h4ck3d”) để tăng độ bao phủ khi phân loại.
  + Log chi tiết (cleaning\_log.txt) được tạo để theo dõi file rỗng, lỗi mã hóa, và thống kê phân loại.
* Hình ảnh:
  + Phân loại tự động dựa trên văn bản trích xuất từ ảnh, sử dụng từ khóa như “hacked”, “404”, “not found”.
  + Làm sạch bằng cách xóa các thư mục not\_found, kiểm tra thủ công các file sai nhãn, và trích xuất lại văn bản với các tham số OCR tối ưu (grayscale, hỗ trợ đa ngôn ngữ).
  + Sau làm sạch, dataset hình ảnh gọn nhẹ hơn, chỉ giữ các thư mục ảnh phù hợp, hỗ trợ huấn luyện mô hình.

## 5.1.3 Kết Quả Huấn Luyện Mô Hình

* Xây dựng mô hình phân loại nhị phân (Normal vs Defaced) dựa trên ResNet50 pretrained, với 1321 ảnh được tổ chức thành hai lớp.
* Kết quả huấn luyện:
  + Accuracy: Đạt 94% trên tập huấn luyện và 92% trên tập validation sau 6 epoch.
  + Loss: Giảm ổn định trên tập huấn luyện (từ 0.43 xuống 0.14), nhưng tập validation có dấu hiệu dao động nhẹ (từ 0.54 xuống 0.25, sau đó tăng nhẹ).
  + Mô hình hội tụ tốt, nhưng có dấu hiệu overfitting nhẹ do accuracy validation giảm ở các epoch cuối.
* Kiểm thử thủ công cho thấy mô hình nhận diện chính xác các giao diện web bị deface (như chèn script, thay đổi giao diện), với độ nhạy cao trong phát hiện thay đổi bất thường.
* Một số dự đoán sai xảy ra với ảnh chất lượng thấp hoặc giao diện gần giống nhau, cho thấy cần cải thiện dữ liệu.

## 5.1.4 Đóng Góp của Đồ Án

* Bộ dữ liệu chất lượng cao:
  + Cung cấp dataset HTML (1201 file) và hình ảnh (2052 ảnh, sau làm sạch ~900 ảnh hợp lệ) với nhãn rõ ràng, phù hợp cho nghiên cứu và huấn luyện mô hình học máy.
  + Dataset đa dạng, bao gồm dữ liệu thực tế từ Zone-H và dữ liệu mô phỏng từ website tự xây dựng.
* Quy trình thu thập và xử lý:
  + Tự động hóa thu thập HTML và ảnh bằng Python (requests, selenium, BeautifulSoup, pytesseract).
  + Xây dựng pipeline làm sạch dữ liệu mạnh mẽ, xử lý các vấn đề như lỗi mạng, mã hóa, và nội dung phức tạp.
* Ứng dụng thực tiễn:
  + Mô hình phân loại đạt độ chính xác cao, có thể triển khai trong hệ thống giám sát website để phát hiện defacement tự động.
  + Dataset hỗ trợ nghiên cứu các kỹ thuật tấn công deface (SQL Injection, XSS, File Upload) và phát triển công cụ bảo mật.

## 5.1.5 Hạn Chế

* Dữ liệu HTML:
* Số lượng file defaced (31 file) ít hơn nhiều so với not\_defaced (1170 file), dẫn đến mất cân bằng lớp trong dataset.
* Một số từ khóa deface có thể chưa bao phủ hết các biến thể, gây sai sót trong phân loại.
* Dữ liệu hình ảnh:
  + Lỗi OCR khiến một số ảnh bị nhãn sai (do trích xuất văn bản không chính xác).
  + Ảnh chất lượng thấp hoặc giao diện tương tự giữa defaced và not\_defaced gây khó khăn cho mô hình.
  + Các trang not\_found chiếm tỷ lệ lớn (~1152 ảnh), làm giảm số lượng dữ liệu hợp lệ.
* Mô hình:
  + Dấu hiệu overfitting nhẹ ở các epoch cuối, cần thêm kỹ thuật điều chỉnh như dropout hoặc data augmentation.
  + Huấn luyện trên CPU giới hạn khả năng xử lý dataset lớn hơn hoặc sử dụng mô hình phức tạp hơn.
* Thu thập dữ liệu:
  + Phụ thuộc vào Zone-H, có thể gặp CAPTCHA hoặc giới hạn truy cập.
  + Dữ liệu mô phỏng chưa phản ánh đầy đủ các kịch bản tấn công thực tế (như Special Defacement).

# 5.2 Hướng Phát Triển

Dựa trên các kết quả và hạn chế, nhóm đề xuất các hướng phát triển sau để cải thiện bộ dữ liệu và mô hình trong tương lai:

## 5.2.1 Cải Thiện Bộ Dữ Liệu

* Tăng cường dữ liệu HTML:
  + Mở rộng thu thập từ các nguồn khác ngoài Zone-H, như diễn đàn hacker hoặc báo cáo bảo mật (ví dụ: Bkav, Viettel Cyber Security).
  + Thu thập thêm các file defaced để cân bằng dataset, sử dụng kỹ thuật như cào dữ liệu từ các website bị tấn công thời gian thực.
  + Tích hợp phân tích log máy chủ (Apache/Nginx) để phát hiện các mẫu tấn công, bổ sung thông tin cho dataset.
* Tối ưu dữ liệu hình ảnh:
  + Sử dụng các mô hình OCR tiên tiến (như Google Cloud Vision API) để cải thiện độ chính xác trích xuất văn bản.
  + Thu thập ảnh từ các nguồn đa dạng hơn (như các website chính phủ, doanh nghiệp lớn bị tấn công) để tăng tính đại diện.
  + Xây dựng pipeline kiểm tra thủ công định kỳ để sửa nhãn sai, đặc biệt với các ảnh có giao diện tương tự.
* Dữ liệu mô phỏng:
  + Mở rộng website mô phỏng với các lỗ hổng phức tạp hơn (như RFI, LFI, hoặc zero-day).
  + Tạo các kịch bản tấn công đa dạng, bao gồm Special Defacement và Redefacement, để phản ánh thực tế.
  + Sử dụng công cụ như Docker để triển khai nhiều phiên bản website, tăng số lượng mẫu mô phỏng.

#### 5.2.2 Nâng Cao Mô Hình Phân Loại

* Khắc phục overfitting:
  + Áp dụng các kỹ thuật như dropout, L2 regularization, hoặc batch normalization trong các lớp fully-connected.
  + Tăng cường data augmentation (xoay, lật, thay đổi độ sáng) để làm phong phú dữ liệu huấn luyện.
  + Sử dụng các mô hình nhẹ hơn (như MobileNetV2) để giảm tài nguyên tính toán, phù hợp với môi trường không có GPU.
* Mở rộng phân loại:
  + Huấn luyện mô hình phân loại đa lớp, bao gồm Normal, Defaced, và Not Found, để tận dụng toàn bộ dataset.
  + Tích hợp phân tích đa phương thức, kết hợp đặc trưng từ HTML (văn bản) và hình ảnh để tăng độ chính xác.
* Triển khai thời gian thực:
  + Phát triển hệ thống giám sát website tích hợp mô hình, sử dụng API để quét và phân loại website theo thời gian thực.
  + Kết hợp với các công cụ như Site24x7 hoặc VNCS Web Monitoring để phát hiện thay đổi bất thường ngay lập tức.

#### 5.2.3 Tối Ưu Quy Trình Thu Thập và Xử Lý

* Tự động hóa thu thập:
  + Sử dụng proxy luân phiên và headless browser (Puppeteer) để vượt qua CAPTCHA và giới hạn truy cập trên Zone-H.
  + Tích hợp Zone-H API (nếu có) để trích xuất dữ liệu nhanh hơn, giảm phụ thuộc vào web scraping.
* Xử lý dữ liệu lớn:
  + Áp dụng đa luồng (multiprocessing) hoặc xử lý phân tán (Spark) để xử lý dataset lớn hơn (~10,000 file HTML/ảnh).
  + Sử dụng cơ sở dữ liệu như MongoDB để lưu trữ và truy vấn dữ liệu hiệu quả.
* Làm sạch dữ liệu:
  + Phát triển pipeline tự động kiểm tra chất lượng dữ liệu, như phát hiện nội dung trùng lặp bằng hash (hashlib).
  + Tích hợp AI để tự động sửa nhãn sai dựa trên phân tích ngữ cảnh (NLP cho HTML, CNN cho ảnh).

#### 5.2.4 Ứng Dụng Thực Tiễn

* Hệ thống phát hiện defacement:
  + Triển khai mô hình như một dịch vụ (API) để các tổ chức sử dụng, tích hợp với WAF hoặc IDS.
  + Phát triển công cụ giám sát mã nguồn mở, hỗ trợ quản trị viên website nhỏ kiểm tra bảo mật.
* Nghiên cứu và giáo dục:
  + Công bố dataset (sau khi làm sạch và ẩn danh) để hỗ trợ cộng đồng nghiên cứu bảo mật.
  + Sử dụng đồ án làm tài liệu giảng dạy trong các khóa học về an ninh mạng tại trường.
* Kết hợp với các loại tấn công khác:
  + Mở rộng nghiên cứu để phát hiện các cuộc tấn công khác (phishing, ransomware) dựa trên đặc trưng HTML và hình ảnh.
  + Tích hợp với phân tích hành vi mạng (log, lưu lượng) để phát hiện tấn công toàn diện.

#### 5.2.5 Hợp Tác và Mở Rộng

* Hợp tác với doanh nghiệp bảo mật:
  + Kết nối với các công ty như Viettel Cyber Security, VNCS, hoặc Bkav để thu thập thêm dữ liệu thực tế và triển khai giải pháp.
  + Tham gia các chương trình bug bounty để thu thập thông tin về các lỗ hổng defacement.
* Cộng đồng mã nguồn mở:
  + Đóng góp mã nguồn thu thập và xử lý dữ liệu lên GitHub, khuyến khích cộng đồng cải thiện.
  + Tham gia các diễn đàn như WhiteHat.vn để chia sẻ kinh nghiệm và học hỏi từ các chuyên gia.

### 5.3 Kết Luận

Đồ án đã thành công trong việc xây dựng một bộ dữ liệu về tấn công web defaced, bao gồm cả HTML và hình ảnh, với quy trình thu thập, xử lý, và huấn luyện mô hình được tự động hóa. Mô hình phân loại đạt độ chính xác cao (94% trên tập huấn luyện, 92% trên tập validation), phù hợp cho các ứng dụng phát hiện defacement tự động. Dù còn một số hạn chế như mất cân bằng lớp, lỗi OCR, và dấu hiệu overfitting, các hướng phát triển đề xuất sẽ giúp khắc phục và nâng cao giá trị của đồ án.

Trong bối cảnh các cuộc tấn công defacement ngày càng gia tăng, đồ án không chỉ đóng góp về mặt học thuật mà còn có ý nghĩa thực tiễn trong việc nâng cao bảo mật website. Nhóm hy vọng rằng các kết quả và đề xuất sẽ là nền tảng cho các nghiên cứu tiếp theo, góp phần xây dựng các giải pháp bảo mật hiệu quả hơn trong tương lai.

TÀI LIỆU THAM KHẢO

* + - 1. **CyStack Blog về Defacement**  
         CyStack. “Defacement là gì? Hiểu biết và phòng chống tấn công thay đổi giao diện website.” *CyStack*, cystack.net/vi/blog/defacement-la-gi. Truy cập lần cuối ngày 1 tháng 5, 2025.
      2. **Wikipedia về SQL Injection**  
         “SQL Injection.” *Wikipedia, Bách khoa toàn thư mở*, Wikimedia Foundation, vi.wikipedia.org/wiki/SQL\_injection. Truy cập lần cuối ngày 28 tháng 4, 2025.
      3. **Vietnix về XSS**  
         Vietnix. “XSS là gì? Tìm hiểu về Cross-Site Scripting và cách phòng tránh.” *Vietnix*, vietnix.vn/xss-la-gi/. Truy cập lần cuối 18 tháng 5, 2025.
      4. **An Toàn Thông Tin về Deep Learning trong Defacement**  
         “Phát hiện tấn công thay đổi giao diện trang web sử dụng phương pháp học sâu.” *An Toàn Thông Tin*, antoanthongtin.vn/mat-ma-dan-su/phat-hien-tan-cong-thay-doi-giao-dien-trang-web-su-dung-phuong-phap-hoc-sau-110476. Truy cập lần cuối ngày 1 tháng 5, 2025.
      5. **VNCS về Defacement**  
         VNCS. “Tấn công thay đổi giao diện website (Defacement) và khuyến cáo từ các chuyên gia an ninh mạng.” *VNCS*, vncs.vn/vi/tin-tuc/detail-tan-cong-thay-doi-giao-dien-website-defacement-va-khuyen-cao-tu-cac-chuyen-gia-an-ninh-mang-323. Truy cập lần cuối ngày 20 tháng 4, 2025.
      6. **VinaHost về Deface**  
         VinaHost. “Deface là gì? Tấn công Deface website hoạt động như thế nào?” *VinaHost*, vinahost.vn/deface-la-gi/. Truy cập lần cuối ngày 2 tháng 5, 2025.
      7. **Exploit-DB về SQL Injection**  
         “SQL Injection Attacks.” *Exploit Database*, Offensive Security, [www.exploit-db.com/docs/44604](http://www.exploit-db.com/docs/44604). Truy cập lần cuối 25 tháng 4, 2025.
      8. **GitHub về Zone-H Grabber**  
         Ev3lM0rty. “Zone-H-Grabber.” *GitHub*, github.com/Ev3lM0rty/Zone-H-Grabber. Truy cập lần cuối 18 tháng 5, 2025.
      9. **Astera về Data Preprocessing**  
         Astera. “What is Data Preprocessing? Definition, Steps, and Tools.” *Astera*, [www.astera.com/type/blog/data-preprocessing/](http://www.astera.com/type/blog/data-preprocessing/). Truy cập lần cuối 12 tháng 4, 2025.
      10. **Unitrain về Trực quan hóa dữ liệu**  
          Unitrain. “Trực quan hóa dữ liệu với Python: Hành trình từ số liệu đến thông tin.” *Unitrain*, unitrain.edu.vn/truc-quan-hoa-du-lieu-voi-python/. Truy cập lần cuối ngày 10 tháng 5, 2025.
      11. **AWS về Data Visualization**  
          Amazon Web Services. “What is Data Visualization?” *AWS*, aws.amazon.com/vi/what-is/data-visualization/. Truy cập lần cuối ngày 2 tháng 5, 2025.
      12. **AWS về Data Analytics**  
          Amazon Web Services. “What is Data Analytics?” *AWS*, aws.amazon.com/vi/what-is/data-analytics/. Truy cập lần cuối ngày 20 tháng 4, 2025.
      13. **Viettel Cybersecurity về Deep Learning trong Defacement**  
          Viettel Cybersecurity. “Phát hiện website Defacement sử dụng công nghệ Deep Learning.” *Viettel Cybersecurity Blog*, blog.viettelcybersecurity.com/phat-hien-website-defacement-su-dung-cong-nghe-deep-learning/. Truy cập lần cuối ngày 18 tháng 5, 2025.

TÀI NGUYÊN BÁO CÁO

**Liên kết đến Google Drive của nhóm:** [**IE105 - Tạo bộ dữ liệu tấn công web defaced**](https://drive.google.com/drive/folders/1JXljh6KrTfGL-KWDjbPqU5gdjCZaIkxg?usp=sharing)

**Liên kết đến Github:** [**Web\_Defaced\_Attack**](https://github.com/Nhat22-archlinux/Web_Defaced_Attack)

**Liên kết đến Slide báo cáo của nhóm:** [**WebDeface\_Slides\_IE105**](https://www.canva.com/design/DAGnwZNpzpU/vrBKtSW12g069M6r-le66A/edit)

**Liên kết đến file training log (PDF):** [**Notebook**](https://drive.google.com/drive/u/1/folders/18wQHMWacqiIP26WM51sfA8oprcZ4vEwQ)**;** [**Training Log (PDF)**](https://drive.google.com/file/d/1ndRi0p-3Fksq4J2HB9pRq1fZS5ZWQ6w6/view?usp=drive_link)

**Liên kết đến file dataset:** [**Dataset**](https://drive.google.com/drive/u/1/folders/17j4t2Wa7b694CueTAWv6u8wnMxJ3P5k8)

**Liên kết đến Video demo của nhóm:** [**Demo**](https://drive.google.com/drive/u/1/folders/1W8lcCn-KXKD1veot0I7unvMT6O1WpIrC)