

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN



## TIỂU LUẬN CHUYÊN NGÀNH

### XÂY DỰNG FRAMEWORK TỰ ĐỘNG HÓA PENETRATION TESTING DỰA TRÊN AI AGENT

GV HƯỚNG DẪN: Th.S Nguyễn Thị Thanh Vân

SV THỰC HIỆN: Nguyễn Thắng Lợi - 22162023

Nguyễn Lưu Gia Bảo - 22162005

TP. HỒ CHÍ MINH – 2025

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày tháng 12 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

**Giáo viên hướng dẫn:** Th.S Nguyễn Thị Thanh Vân

**Sinh viên thực hiện:** Nguyễn Thắng Lợi - 22162023

Nguyễn Lưu Gia Bảo - 22162005

- Nhận xét về nội dung và khối lượng thực hiện:
- Đánh giá ưu nhược điểm:
- Điểm số: .....

Giảng viên hướng dẫn

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày tháng 12 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

Giáo viên phản biện: Th.S Nguyễn Đăng Quang

Sinh viên thực hiện: Nguyễn Thắng Lợi - 22162023

Nguyễn Lưu Gia Bảo - 22162005

- Nhận xét về nội dung và khối lượng thực hiện:
- Đánh giá ưu nhược điểm:
- Điểm số: .....

Giảng viên phản biện

## LỜI CẢM ƠN

Trước hết, nhóm thực hiện đề tài xin gửi lời cảm ơn chân thành đến Ban Giám hiệu Trường Đại học Sư phạm Kỹ thuật TP.HCM và quý thầy, cô Khoa Công nghệ Thông tin đã tạo điều kiện thuận lợi và cung cấp môi trường học tập và nghiên cứu hiện đại giúp nhóm có được nền tảng kiến thức vững chắc trong suốt quá trình học tập tại trường.

Đặc biệt, nhóm xin bày tỏ lòng biết ơn sâu sắc đến Th.S Nguyễn Thị Thanh Vân. Cô đã tận tình hướng dẫn, định hướng đề tài và đưa ra những lời khuyên, nhận xét chuyên môn quý báu giúp nhóm giải quyết các vấn đề khó khăn trong quá trình nghiên cứu về Penetration Testing và AI Agent. Những chỉ dẫn của cô là kim chỉ nam giúp nhóm hoàn thiện dự án một cách tốt nhất có thể. Nhóm cũng xin gửi lời cảm ơn đến các anh chị, bạn bè và đồng nghiệp tại Tập Đoàn FPT đã luôn động viên, chia sẻ tài liệu và góp ý trong quá trình xây dựng và kiểm thử hệ thống trên các môi trường giả lập cũng như thực tế.

Mặc dù đã nỗ lực hết mình để hoàn thành tiểu luận với đề tài "Xây dựng Framework tự động hóa Penetration Testing dựa trên AI Agent", nhưng do giới hạn về mặt thời gian và kiến thức đối với các công nghệ mới như LLM và Multi-Agent System, bài báo cáo khó tránh khỏi những thiếu sót. Nhóm thực hiện rất mong nhận được sự thông cảm và những ý kiến đóng góp quý báu từ quý cô để đề tài ngày càng hoàn thiện và có tính ứng dụng cao hơn trong tương lai.

Xin chân thành cảm ơn!

## LỜI NÓI ĐẦU

Trong kỷ nguyên chuyển đổi số toàn diện hiện nay, an ninh mạng đã trở thành yếu tố sống còn đối với mọi tổ chức. Song song với đó là sự tinh vi ngày càng tăng của các cuộc tấn công mạng. Điều này đã đặt ra những thách thức chưa từng có cho công tác đảm bảo an toàn thông tin trong kỷ nguyên mới của Internet, nơi những trí tuệ nhân tạo hay mô hình ngôn ngữ lớn phủ sóng và phát triển nồng độ ngày càng tăng.

Tuy nhiên, quy trình kiểm thử xâm nhập hiện tại đang đối mặt với một nghịch lý lớn về hiệu suất và độ chính xác. Các công cụ rà quét tự động (DAST) truyền thống tuy có tốc độ cao nhưng lại thiếu khả năng hiểu ngữ cảnh, thường xuyên bỏ sót các lỗ hổng logic nghiệp vụ quan trọng. Ngược lại, kiểm thử thủ công dù đảm bảo độ sâu nhưng lại tiêu tốn quá nhiều thời gian và nguồn lực con người cho các tác vụ lặp đi lặp lại. Đồng thời các công nghệ về trí tuệ nhân tạo (AI) đang dần phát triển và len lỏi sâu vào công tác bảo mật thông tin trong thời đại mới.

Từ thực tiễn đó, nhóm nghiên cứu đã lựa chọn đề tài "Xây dựng Framework tự động hóa Penetration Testing dựa trên AI Agent". Mục tiêu của đề tài là ứng dụng những tiến bộ mới nhất về AI, cụ thể là các mô hình ngôn ngữ lớn (LLM) và kiến trúc đa tác nhân (Multi-Agent System) nhằm xây dựng một giải pháp có khả năng "suy luận" và tự động hóa quy trình kiểm thử xâm nhập một cách thông minh. Tiêu luận này sẽ trình bày chi tiết về quá trình xây dựng framework AiPT - một hệ thống hoạt động theo mô hình cộng tác Human-in-the-loop, nơi AI đóng vai trò là trợ lý đắc lực giúp điều phối các công cụ bảo mật truyền thống, giải phóng con người khỏi các tác vụ nhàn chán để tập trung vào các vấn đề phức tạp hơn.

Mặc dù đã có nhiều nỗ lực trong quá trình nghiên cứu và thực nghiệm, nhưng do lĩnh vực AI Agent trong an ninh mạng còn khá mới mẻ và phức tạp, tiêu luận khó tránh khỏi những thiếu sót. Nhóm thực hiện rất mong nhận được sự thông cảm và những ý kiến đóng góp quý báu từ quý thầy-cô để đề tài ngày càng hoàn thiện hơn.

## MỤC LỤC

LỜI CẢM ƠN.....	i
LỜI NÓI ĐẦU.....	ii
MỤC LỤC.....	iv
DANH MỤC BẢNG BIÊU.....	vi
DANH MỤC HÌNH MINH HỌA.....	vii
PHẦN MỘT - MỞ ĐẦU.....	1
1. Lý do chọn đề tài.....	1
2. Mục tiêu nghiên cứu.....	1
3. Đối tượng và phạm vi nghiên cứu.....	2
4. Phương pháp nghiên cứu.....	2
PHẦN HAI - NỘI DUNG.....	4
CHƯƠNG 1: TỔNG QUAN VỀ PENETRATION TESTING.....	4
1.1. Giới thiệu về Penetration Testing.....	4
1.2. Các phương pháp luận phổ biến trong PT.....	6
1.3. Khái niệm về Automated Pentesting.....	12
1.4. Tổng kết chương.....	15
CHƯƠNG 2: CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN.....	16
2.1. Các công cụ PT truyền thống.....	16
2.2. Ứng dụng Machine Learning và Deep Learning trong PT.....	19
2.3. Các framework sử dụng AI Agent trong PT.....	22

2.4. Xác định vấn đề cần nghiên cứu.....	25
2.5. Tổng kết chương.....	28
CHƯƠNG 3: KIẾN TRÚC HỆ THỐNG VÀ TRIỂN KHAI.....	30
3.1. Kiến trúc tổng thể.....	30
3.2. Các thuật ngữ quan trọng trong AiPT.....	32
3.3. Cơ chế quản lý ngũ cảnh và tối ưu hóa bộ nhớ.....	34
3.4. Module Recon Agents.....	36
3.5. Module Exploit Agents.....	39
3.6. Cơ chế tích hợp công cụ và chiến lược.....	42
3.7. Phát triển giao diện web.....	43
3.8. Tổng kết chương.....	46
CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ.....	48
4.1. Môi trường và kịch bản thực nghiệm.....	48
4.2. Tiến hành kiểm thử.....	49
4.3. Đánh giá hiệu quả.....	61
4.4. Tổng kết chương.....	63
PHẦN BA - KẾT LUẬN.....	64
CHƯƠNG 5: 1. Kết quả đạt được.....	64
CHƯƠNG 6: 2. Hướng phát triển.....	65
TÀI LIỆU THAM KHẢO.....	67

**DANH MỤC BẢNG BIỂU**

Bảng 1: So sánh hiệu năng và đặc điểm của các framework DAST hàng đầu	18
Bảng 2. So sánh các công nghệ tự động hóa PT qua các thế hệ	26
Bảng 3: Thống kê số lượng lỗ hổng phát hiện trong môi trường lab	56

**DANH MỤC HÌNH MINH HỌA**

Hình 1.1. SP 800-115 được ban hành bởi NIST	9
Hình 1.2. Các kênh kiểm thử chính trong OSSTMM	10
Hình 1.3. Cấu trúc trong ISSAF	11
Hình 3.1. Sơ đồ kiến trúc tổng thể của hệ thống AiPT	30
Hình 3.2. Mô hình quan hệ giữa workflow, test-case và API endpoints	32
Hình 3.3. Sơ đồ chi tiết hoạt động của module Recon Agents	36
Hình 3.4. Kiến trúc chi tiết của module Exploit Agents	40
Hình 3.5. Giao diện khởi tạo cuộc kiểm thử mới với đầu vào là HTTP traffic	43
Hình 3.6. Bảng giám sát trạng thái hoạt động của các agent và terminal output	45
Hình 3.7. Danh sách các lỗ hổng bảo mật được phát hiện và phân loại theo mức độ nghiêm trọng	46
Hình 4.1. Ứng dụng OWASP Juice Shop	48
Hình 4.2. Website Thư viện SGU	49
Hình 4.3. Recon Agents hoạt động để dò tìm các API của web-app	50
Hình 4.4. Màn hình khởi tạo Assessment với API request do Recon-agents gửi	51
Hình 4.5. Chiến lược được Agent đề xuất và thực thi	52
Hình 4.6. Quan sát logs trong quá trình Exploit Agents thực thi nhiệm vụ	53

Hình 4.7. Màn hình thể hiện các kết quả thu được từ quá trình kiểm thử của Exploit-Agents	54
Hình 4.8. Chi tiết mỗi findings của agent	55
Hình 4.9. Khởi tạo bài kiểm thử cho kịch bản thứ hai	59
Hình 4.10. Các findings của agent trên mục tiêu trong kịch bản thứ hai	61

## PHẦN MỘT - MỞ ĐẦU

### 1. Lý do chọn đề tài

Trong kỷ nguyên chuyển đổi số, an ninh mạng đã trở thành yếu tố sống còn đối với mọi tổ chức. Nhu cầu kiểm thử bảo mật (Penetration Testing - Pentest) đang gia tăng đột biến nhằm đối phó với sự bùng nổ của tội phạm mạng, khi các hoạt động của hacker ngày càng trở nên tinh vi, phức tạp và có tổ chức hơn. Tuy nhiên, quy trình kiểm thử xâm nhập hiện tại đang đối mặt với một nghịch lý về công cụ và hiệu suất. Một mặt, các công cụ quét lỗ hổng tự động như DAST tuy có tốc độ cao nhưng lại thiếu khả năng hiểu ngữ cảnh và logic nghiệp vụ, dẫn đến tỷ lệ dương tính giả cao và bỏ sót các lỗ hổng sâu. Mặt khác, kiểm thử thủ công đảm bảo được độ sâu và tư duy logic nhưng lại tiêu tốn quá nhiều thời gian và nguồn lực con người cho các tác vụ lặp đi lặp lại như thu thập thông tin hay rà quét cơ bản.

Xuất phát từ thực tiễn đó, việc ứng dụng trí tuệ nhân tạo, cụ thể là các AI Agent, đang nổi lên như một giải pháp đột phá. Giải pháp này hứa hẹn khả năng tự động hóa các tác vụ thủ công nhưng vẫn giữ được khả năng "suy luận" và thao tác linh hoạt như con người. Vì vậy, đề tài nghiên cứu xây dựng framework tự động hóa Pentest dựa trên AI Agent là cấp thiết và có ý nghĩa thực tiễn cao.

### 2. Mục tiêu nghiên cứu

Mục tiêu cốt lõi của đề tài là xây dựng và phát triển AiPT - một framework hỗ trợ tự động hóa quy trình kiểm thử xâm nhập dành cho ứng dụng Web và API. Dự án hướng đến các mục tiêu cụ thể sau:

- Tự động hóa các tác vụ lặp lại: Giúp các chuyên gia kiểm thử (Pentester) giảm tải khỏi lượng công việc nhảm chán như thu thập thông tin, kiểm tra cấu hình và rà quét các lỗi cơ bản.

- Tối ưu hóa nguồn lực con người: Bằng cách chuyển giao các tác vụ cấp thấp cho AI, Pentester có thể tập trung trí tuệ và thời gian vào việc phân tích các lỗ hổng logic nghiệp vụ phức tạp và các chuỗi tấn công nâng cao mà máy móc chưa thể thực hiện.
- Xây dựng cơ chế kiểm thử linh hoạt: Thiết kế hệ thống cho phép tùy biến cao thông qua các *workflow* và *test-case* (định dạng YAML), giúp framework dễ dàng thích nghi với các môi trường và yêu cầu kiểm thử đa dạng.

### 3. Đối tượng và phạm vi nghiên cứu

Để đảm bảo tính khả thi và độ sâu của đề tài, nghiên cứu này được giới hạn cụ thể về đối tượng và phạm vi như sau:

- Đối tượng nghiên cứu: Tập trung vào các ứng dụng web hiện đại và các giao diện lập trình ứng dụng theo chuẩn RESTful API. Đây là hai bề mặt tấn công phổ biến và quan trọng nhất trong hạ tầng số hiện nay.
- Phạm vi nghiên cứu:
  - Về mặt lỗ hổng: Tập trung phát hiện và khai thác các nhóm lỗ hổng phổ biến thuộc danh sách OWASP Top 10, bao gồm các lỗi nghiêm trọng như Injection, Broken Access Control.
  - Về mặt cấu hình: Bao gồm các kiểm thử liên quan đến cấu hình bảo mật và các lỗi rò rỉ thông tin.

### 4. Phương pháp nghiên cứu

Đề tài áp dụng kết hợp phương pháp nghiên cứu lý thuyết và thực nghiệm để giải quyết vấn đề:

- Phương pháp nghiên cứu lý thuyết: Tập trung thu thập, phân tích và tổng hợp các tài liệu khoa học, công trình nghiên cứu về LLM và kiến trúc AI Agent để xây dựng cơ sở lý luận vững chắc cho việc thiết kế hệ thống.

- Phương pháp thực nghiệm:
  - Tiến hành thiết kế và lập trình framework AiPT dựa trên các công nghệ đã nghiên cứu.
  - Triển khai kiểm thử đánh giá hiệu quả của công cụ trên các môi trường phòng thí nghiệm tiêu chuẩn để đo lường khả năng phát hiện lỗ hổng.
  - Thủ nghiệm giới hạn trên các ứng dụng thực tế (mục đích giáo dục, không gây hại, không tấn công) để đánh giá khả năng hoạt động trong môi trường production.

## PHẦN HAI - NỘI DUNG

### CHƯƠNG 1: TỔNG QUAN VỀ PENETRATION TESTING

#### 1.1. Giới thiệu về Penetration Testing

##### a) Định nghĩa về Penetration Testing

Penetration Testing (PT), thường được gọi là "pentest" hay kiểm thử xâm nhập, là một quy trình hợp pháp và được cấp phép nhằm đánh giá một cách chủ động tình trạng an ninh của một tài sản kỹ thuật số (hệ thống máy tính, mạng, ứng dụng web,...) bằng cách mô phỏng một cuộc tấn công có chủ đích từ các tác nhân độc hại (còn gọi là tin tặc mũ đen). Đây là một hình thức tấn công có đạo đức (ethical hacking), với mục tiêu chính là xác định và khai thác các lỗ hổng bảo mật tiềm ẩn trước khi chúng bị kẻ xấu phát hiện và lợi dụng. Quá trình này không chỉ dừng lại ở việc phát hiện lỗ hổng mà còn bao gồm việc đánh giá mức độ ảnh hưởng của chúng đến tổ chức, từ đó đưa ra các khuyến nghị khắc phục hiệu quả.

Trong bối cảnh an ninh mạng hiện đại, PT được xem là một trong những phương pháp hiệu quả nhất để kiểm tra khả năng phòng thủ của một hệ thống trước các đối thủ có kỹ năng cao và là nền tảng của các phương pháp luận an ninh mạng.

##### b) Vai trò trong vòng đời phát triển phần mềm an toàn

Kiểm thử xâm nhập đóng một vai trò tối quan trọng và không thể thiếu trong chiến lược phòng thủ theo chiều sâu của một tổ chức. Thay vì chỉ được thêm vào như một bước kiểm tra cuối cùng, các biện pháp an ninh cần được tích hợp trong suốt vòng đời phát triển phần mềm. PT hoạt động như một cơ chế xác thực thực tế, giúp trả lời câu hỏi: "Liệu các biện pháp kiểm soát an ninh của chúng ta có thực sự hiệu quả trước một cuộc tấn công thực tế hay không?". Những lợi ích chính mà PT mang lại bao gồm:

- Xác định lỗ hổng một cách chủ động: Tìm ra các điểm yếu trong thiết kế, triển khai và cấu hình hệ thống trước khi chúng bị khai thác.
- Đánh giá tác động thực tế: Cung cấp cái nhìn rõ ràng về mức độ thiệt hại mà một cuộc tấn công thành công có thể gây ra, giúp tổ chức ưu tiên các nỗ lực khắc phục.
- Kiểm tra tuân thủ: Nhiều tiêu chuẩn và quy định bảo mật như PCI DSS, HIPAA và GDPR yêu cầu các tổ chức thực hiện PT định kỳ để đảm bảo tuân thủ.
- Nâng cao nhận thức an ninh: Kết quả từ PT giúp nâng cao nhận thức về các mối đe dọa an ninh cho cả đội ngũ phát triển và nhân viên trong tổ chức.

### c) Các giai đoạn của một quy trình Penetration Testing

Một quy trình PT điển hình được cấu trúc thành nhiều giai đoạn tuần tự để đảm bảo tính toàn diện và hệ thống. Mặc dù tên gọi và số lượng các giai đoạn có thể khác nhau tùy theo phương pháp luận, chúng thường bao gồm các hoạt động cốt lõi sau đây:

1. Giai đoạn 1 - Footprinting & Reconnaissance: Đây là giai đoạn đầu tiên và quan trọng nhất, nơi pentest thu thập càng nhiều thông tin về mục tiêu càng tốt. Thông tin có thể bao gồm địa chỉ IP, tên miền, công nghệ sử dụng (hệ điều hành, máy chủ web), thông tin về nhân viên và cấu trúc tổ chức. Giai đoạn này được thực hiện một cách thụ động (không tương tác trực tiếp) hoặc chủ động.
2. Giai đoạn 2 - Scanning: Dựa trên thông tin thu thập được, pentester bắt đầu tương tác với hệ thống mục tiêu để tìm kiếm các điểm yếu tiềm tàng. Các kỹ thuật trong giai đoạn này bao gồm quét cổng (port scanning) để xác định các

dịch vụ đang chạy, quét lỗ hổng (vulnerability scanning) để tìm các điểm yếu đã biết, và vẽ bản đồ mạng.

3. Giai đoạn 3 - Enumeration & Exploitation: Trong giai đoạn này, pentester cố gắng giành quyền truy cập vào hệ thống bằng cách khai thác các lỗ hổng đã được xác định ở giai đoạn trước. Các kỹ thuật khai thác có thể bao gồm tấn công ứng dụng web (như SQL Injection, XSS) hoặc tấn công hệ thống (như tràn bộ đệm, bẻ khóa mật khẩu). Mục tiêu là giành được quyền kiểm soát ban đầu trên một thành phần của hệ thống.
4. Giai đoạn 4 - Post-Exploitation: Sau khi đã xâm nhập thành công, pentester sẽ cố gắng duy trì quyền truy cập, leo thang đặc quyền (ví dụ từ người dùng thường lên quản trị viên), và di chuyển ngang (lateral movement) sang các hệ thống khác trong cùng mạng lưới. Mục tiêu của giai đoạn này là để xác định giá trị thực sự của hệ thống bị xâm nhập và đánh giá tác động tổng thể của cuộc tấn công.
5. Giai đoạn 5 - Reporting: Đây là giai đoạn cuối cùng và quan trọng nhất, nơi tất cả các phát hiện được tổng hợp thành một báo cáo chi tiết. Báo cáo phải mô tả rõ ràng các lỗ hổng đã tìm thấy, các bước để tái tạo chúng, bằng chứng khai thác thành công, đánh giá mức độ rủi ro, và đưa ra các khuyến nghị cụ thể để khắc phục.

## 1.2. Các phương pháp luận phổ biến trong PT

Để chuẩn hóa và đảm bảo chất lượng cho các quy trình PT, cộng đồng an ninh mạng đã phát triển nhiều phương pháp luận và tiêu chuẩn được công nhận rộng rãi. Việc tuân theo một phương pháp luận nhất quán giúp đảm bảo bài kiểm thử được thực hiện một cách toàn diện, có hệ thống và có thể lặp lại.

a) *Penetration Testing Execution Standard*

**Penetration Testing Execution Standard (PTES)** [1] là tiêu chuẩn toàn diện được thiết kế để cung cấp một lộ trình rõ ràng và nhất quán cho một cuộc kiểm thử xâm nhập hiệu quả. Thay vì chỉ tập trung vào khía cạnh kỹ thuật, PTES bao quát toàn bộ vòng đời của một dự án pentest, đảm bảo chất lượng và mang lại giá trị thực tiễn cho doanh nghiệp. Tiêu chuẩn này chia quy trình thành bảy giai đoạn chính:

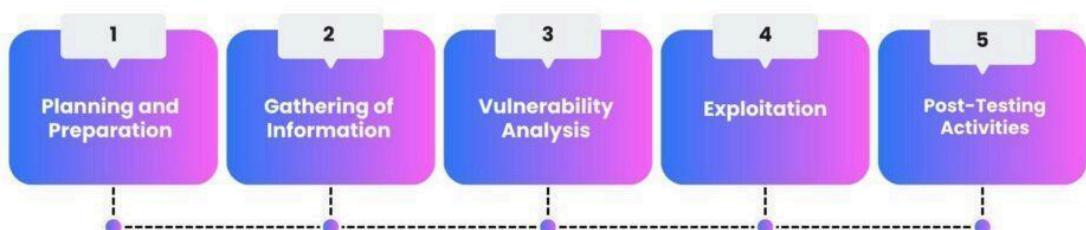
- Pre-engagement Interactions: Giai đoạn nền tảng, tập trung vào việc xác định phạm vi, mục tiêu, các quy tắc và giới hạn của cuộc kiểm thử. Việc thống nhất các điều khoản này giúp đảm bảo cả hai bên (đội ngũ pentest và khách hàng) có cùng một kỳ vọng và hiểu biết về những gì sẽ được thực hiện.
- Intelligence Gathering: Giai đoạn thu thập thông tin một cách thụ động và chủ động về mục tiêu. Các kỹ thuật như OSINT (Open-Source Intelligence) được sử dụng để tìm hiểu về cơ sở hạ tầng, nhân sự và công nghệ của tổ chức, tạo tiền đề cho các bước tấn công sau này.
- Threat Modeling: Dựa trên thông tin đã thu thập, giai đoạn này tập trung vào việc xác định các vector tấn công tiềm năng và các mối đe dọa có khả năng xảy ra cao nhất đối với hệ thống. Việc này giúp đội ngũ pentest ưu tiên các nỗ lực vào những khu vực rủi ro nhất.
- Vulnerability Analysis: Kết hợp các công cụ tự động và kỹ thuật thủ công để xác định các điểm yếu, lỗ hổng bảo mật cụ thể trong các ứng dụng, hệ thống và mạng của mục tiêu.
- Exploitation: Giai đoạn tấn công thử nghiệm, trong đó các chuyên gia sẽ tìm cách khai thác các lỗ hổng đã được xác định để giành quyền truy cập trái phép vào hệ thống. Mục tiêu là để chứng minh sự tồn tại và mức độ nghiêm trọng của lỗ hổng.

- Post-Exploitation: Sau khi đã xâm nhập thành công, giai đoạn này tập trung vào việc xác định giá trị của hệ thống bị xâm nhập, duy trì quyền truy cập và leo thang đặc quyền để mở rộng phạm vi kiểm soát trong mạng lưới của tổ chức.
- Reporting: Giai đoạn cuối cùng và quan trọng nhất, tổng hợp tất cả các phát hiện, bằng chứng, phân tích rủi ro và đề xuất các biện pháp khắc phục chi tiết. Một báo cáo PTES chất lượng không chỉ liệt kê lỗ hổng mà còn cung cấp một lộ trình rõ ràng để doanh nghiệp cải thiện tình hình an ninh.

b) *NIST SP 800-115 (Technical Guide to Information Security Testing and Assessment)*

Được ban hành bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ, **SP 800-115** [2] không phải là một phương pháp luận pentest cứng nhắc mà là một bộ hướng dẫn kỹ thuật toàn diện. Trọng tâm chính của tài liệu này là cung cấp một khuôn khổ có cấu trúc để các tổ chức có thể lập kế hoạch, thực hiện và duy trì các hoạt động kiểm tra và đánh giá an ninh thông tin một cách hiệu quả. Điểm cốt lõi của NIST SP 800-115 là cách tiếp cận dựa trên quản lý rủi ro. Nó nhấn mạnh rằng việc kiểm thử bảo mật phải là một phần không thể thiếu trong chu trình quản lý rủi ro của tổ chức. Thay vì chỉ đơn thuần tìm kiếm lỗ hổng, nó hướng dẫn các tổ chức với 5 giai đoạn cụ thể như mô tả ở hình 1, việc này giúp:

## The 5 Phase Methodology of NIST SP 800 115



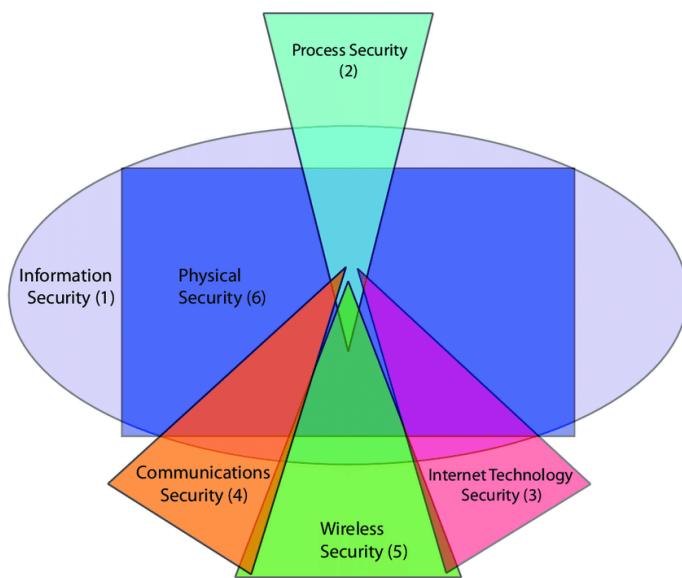
Hình 1.1: SP 800-115 được ban hành bởi NIST

- Xác định các kỹ thuật đánh giá phù hợp: Hướng dẫn lựa chọn các phương pháp kiểm thử (ví dụ: rà quét lỗ hổng, pentest, đánh giá mã nguồn) dựa trên mục tiêu và yêu cầu cụ thể.
- Xây dựng kế hoạch kiểm thử: Cung cấp một quy trình chi tiết để lập kế hoạch, bao gồm xác định phạm vi, quy tắc, nhân sự và các nguồn lực cần thiết.
- Phân tích kết quả và báo cáo: Đưa ra các chỉ dẫn về cách phân tích các phát hiện và trình bày chúng một cách rõ ràng, gắn liền với các rủi ro kinh doanh cụ thể để ban lãnh đạo có thể đưa ra quyết định phù hợp.

### c) Open Source Security Testing Methodology Manual

**Open Source Security Testing Methodology Manual (OSSTMM)** [3] nổi bật với cách tiếp cận khoa học và định lượng. Được phát triển bởi Viện Nghiên cứu An ninh và Mở (ISECOM), OSSTMM không chỉ là một quy trình mà còn là một phương pháp luận kiểm toán (audit) bảo mật. Mục tiêu của nó là cung cấp các kết quả có thể kiểm chứng, đo lường và lặp lại được.

OSSTMM tập trung vào việc phân tích "operational security" (an ninh vận hành) và chia các kênh kiểm thử thành năm loại chính (xem hình 2):



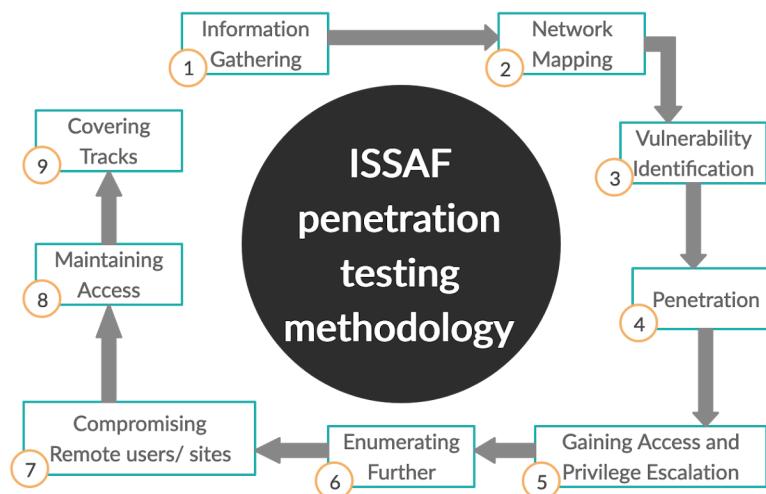
Hình 1.2: Các kênh kiểm thử chính trong OSSTMM

- An ninh con người (Human Security): Đánh giá các yếu tố liên quan đến con người như social engineering.
- An ninh vật lý (Physical Security): Kiểm tra các biện pháp kiểm soát truy cập ở cấp độ vật lý.
- An ninh không dây (Wireless Security): Bao gồm Wi-Fi, Bluetooth và các giao thức không dây khác.
- An ninh viễn thông (Telecommunications Security): Tập trung vào hệ thống điện thoại, VoIP.
- An ninh dữ liệu (Data Networks Security): Bao gồm các mạng theo mô hình TCP/IP truyền thống.

Một trong những điểm đặc biệt nhất của OSSTMM là việc sử dụng RAV (Risk Assessment Value), một hệ thống số liệu cho phép định lượng mức độ rủi ro và tình trạng an ninh tổng thể của một tổ chức. Điều này giúp chuyển các khái niệm an ninh trừu tượng thành các con số cụ thể, dễ dàng hơn cho việc so sánh và theo dõi tiến trình cải thiện.

#### *d) Information Systems Security Assessment Framework*

**Information Systems Security Assessment Framework (ISSAF)** [4] là một khung làm việc chuyên sâu, tập trung mạnh vào khía cạnh kỹ thuật của việc kiểm thử xâm nhập. Điểm độc đáo của ISSAF là cấu trúc chi tiết, được thiết kế để cung cấp cho các chuyên gia pentest một bộ công cụ và quy trình toàn diện cho từng giai đoạn đánh giá. ISSAF được xây dựng dựa trên triết lý mọi cuộc kiểm thử đều phải được tùy chỉnh để phù hợp với môi trường công nghệ và nhu cầu kinh doanh.



Hình 1.3: Cấu trúc trong ISSAF

Hình 3 mô tả cấu trúc của ISSAF, phân loại các lĩnh vực đánh giá (ví dụ: đánh giá máy chủ, đánh giá mạng, đánh giá ứng dụng) và trong mỗi lĩnh vực, nó cung cấp:

- Quy trình từng bước: Hướng dẫn chi tiết cách thực hiện các tác vụ kiểm thử.

- Ánh xạ công cụ: Liên kết rõ ràng giữa mỗi tác vụ và các công cụ (cả thương mại và mã nguồn mở) có thể được sử dụng để thực hiện nó.
- Giải thích kết quả: Hướng dẫn cách diễn giải kết quả từ các công cụ và các kỹ thuật tấn công.

Cách tiếp cận này làm cho ISSAF trở thành một tài nguyên vô giá cho các chuyên gia pentest, đặc biệt là những người cần một hướng dẫn chi tiết về "làm thế nào" để thực hiện các cuộc tấn công và đánh giá kỹ thuật phức tạp.

Mặc dù các phương pháp luận nói trên cung cấp một khung làm việc vững chắc cho PT thủ công, việc áp dụng chúng vào một bối cảnh hoàn toàn tự động gặp nhiều thách thức, chẳng hạn như xử lý các kết quả dương tính giả, sự phức tạp của các bài kiểm thử đa giai đoạn, và khả năng thích ứng với các tình huống bất ngờ.

### 1.3. Khái niệm về Automated Pentesting

Kiểm thử xâm nhập tự động (Automated Penetration Testing) là một phương pháp tiếp cận kỹ thuật, trong đó các nền tảng phần mềm và scripts chuyên dụng được sử dụng để tự động hóa một cách có hệ thống quy trình phát hiện và xác minh các lỗ hổng bảo mật. Thay vì mô phỏng một cuộc tấn công đơn lẻ, các công cụ này thực thi hàng loạt các thuật toán và signature-based checks (kiểm tra dựa trên dấu hiệu) trên toàn bộ bề mặt tấn công (attack surface) của hệ thống, bao gồm ứng dụng web, cơ sở hạ tầng mạng và các API. Mục tiêu cốt lõi là gia tăng tốc độ và quy mô của việc đánh giá an ninh, cung cấp một cơ chế phản hồi gần như tức thì trong các quy trình phát triển hiện đại (DevSecOps) và cho phép các tổ chức nhanh chóng xác định và khắc phục các mối đe dọa đã biết.

Lợi ích của tự động hóa không chỉ dừng lại ở việc tiết kiệm chi phí mà còn dựa trên các nguyên tắc khoa học về tính nhất quán và khả năng tái tạo. Con người bị giới hạn bởi khả năng xử lý tuần tự, trong khi máy móc có thể thực hiện các tác vụ

song song với tốc độ vượt trội. Một chuyên gia pentest có thể mất vài giờ để kiểm tra thủ công 100 tham số đầu vào của một ứng dụng web cho lỗ hổng SQL Injection. Ngược lại, một công cụ như SQLMap hoặc Burp Suite Pro Scanner có thể gửi hàng nghìn payload (dữ liệu tấn công) được chế tạo tinh vi đến tất cả các điểm cuối (endpoints) của ứng dụng trong vài phút, bao phủ toàn diện các biến thể tấn công (ví dụ: time-based, boolean-based, error-based). Tương tự, Nmap có thể quét toàn bộ 65,535 cổng của một máy chủ trong thời gian cực ngắn.

Tính nhất quán và khả năng tái hiện là một nguyên tắc nền tảng của phương pháp khoa học. Tự động hóa đảm bảo rằng mỗi lần kiểm tra đều được thực hiện theo cùng một kịch bản, loại bỏ hoàn toàn các biến số do con người gây ra như mệt mỏi, thiếu kinh nghiệm hay sai sót. Kết quả nhất quán cho phép các tổ chức thiết lập security baseline. Bằng cách so sánh kết quả quét theo thời gian, họ có thể đo lường chính xác hiệu quả của các biện pháp và lỗi và xác định các sai lệch về an ninh một cách đáng tin cậy. Điều này đặc biệt quan trọng trong môi trường CI/CD, nơi các thay đổi mã nguồn diễn ra thường xuyên.

Bằng cách tự động hóa các tác vụ lặp đi lặp lại và có tính quy luật cao các chuyên gia bảo mật được giải phóng. Họ có thể tập trung vào các nhiệm vụ đòi hỏi tư duy bậc cao như phân tích các lỗ hổng logic phức tạp, lập mô hình mối đe dọa, và nghiên cứu các kỹ thuật tấn công mới.

Mặc dù có nhiều ưu điểm, các giải pháp tự động hóa hiện tại vẫn đối mặt với những rào cản kỹ thuật đáng kể, hạn chế hiệu quả thực tiễn của chúng. Hệ sinh thái công cụ pentest là một tập hợp các ứng dụng rời rạc, mỗi công cụ được tối ưu cho một nhiệm vụ cụ thể. Một quy trình tấn công có thể bắt đầu bằng nmap để khám phá cổng, sau đó kết quả (ví dụ: cổng 80 đang mở) được chuyển thủ công sang nikto để quét lỗ hổng web server. Nếu nikto phát hiện một ứng dụng web, chuyên gia lại phải khởi chạy burp suite để phân tích sâu hơn. Quy trình "copy-paste" thủ công

này không chỉ làm giảm hiệu suất mà còn tiềm ẩn nguy cơ bỏ sót thông tin. Chúng thiếu các cấu trúc điều khiển luồng như if-then-else, vòng lặp, hay thực thi song song dựa trên kết quả trung gian. "Nếu nmap phát hiện cổng 3389 (rdp) đang mở, thì thực thi hydra để kiểm tra mật khẩu yếu; ngược lại, nếu cổng 22 (ssh) mở, thì kiểm tra các thuật toán mã hóa yếu". Việc tự động hóa chuỗi logic điều kiện này hiện tại là bất khả thi với các công cụ đơn lẻ.

Các công cụ khác nhau tạo ra kết quả với các định dạng và cấu trúc dữ liệu hoàn toàn khác biệt (ví dụ: Nmap có thể xuất ra XML, Nessus có thể xuất ra file .nessus hoặc CSV, các công cụ khác lại là JSON hoặc văn bản thô). Việc thiếu một data schema chung cho việc mô tả lỗ hổng gây ra một rào cản lớn cho việc tổng hợp và tương quan hóa kết quả. Việc phân tích tự động một chuỗi tấn công yêu cầu phải chuẩn hóa các thực thể (như địa chỉ IP, cổng, tên lỗ hổng, CVE) từ nhiều nguồn dữ liệu khác nhau, một bài toán kỹ thuật phức tạp.

Các công cụ tự động thường hoạt động dựa trên việc khớp mẫu (pattern matching) hoặc kiểm tra phiên bản phần mềm mà thiếu đi khả năng phân tích ngữ cảnh. Một scanner có thể xác định máy chủ web đang chạy phiên bản Apache/2.4.29, một phiên bản được biết là có lỗ hổng X (CVE-YYYY-ZZZZ). Nó sẽ ngay lập tức báo cáo đây là một lỗ hổng nghiêm trọng. Tuy nhiên, nó không thể xác minh được rằng quản trị viên hệ thống đã áp dụng một bản vá lỗi từ nhà cung cấp hệ điều hành (như Red Hat, Debian) để khắc phục lỗ hổng X mà không thay đổi chuỗi phiên bản của Apache. Đây là một báo động giả, và việc xác minh thủ công hàng trăm cảnh báo như vậy làm lãng phí thời gian của chuyên gia.

"Điểm mù" với lỗ hổng logic có lẽ là hạn chế lớn nhất của tự động hóa. Các công cụ không thể "hiểu" được mục đích hay quy trình nghiệp vụ của một ứng dụng. Hãy xem xét quy trình thanh toán của một trang web thương mại điện tử. Một lỗ hổng logic nghiệp vụ có thể cho phép người dùng thay đổi giá của một mặt hàng trong

giỏ hàng thành 1 dòng bằng cách gửi một yêu cầu HTTP đã bị sửa đổi ở bước cuối cùng. Một scanner, chỉ tìm kiếm các dấu hiệu kỹ thuật như SQL Injection hay XSS, sẽ hoàn toàn bỏ lỡ loại lỗ hổng này vì nó đòi hỏi sự hiểu biết về quy trình "thêm vào giỏ hàng → xác nhận → thanh toán".

#### **1.4. Tổng kết chương**

Chương này đã trình bày một cái nhìn toàn diện về lý thuyết và thực tiễn của PT, khẳng định vai trò không thể thiếu của nó trong quy trình đảm bảo an ninh mạng hiện đại. Thông qua việc phân tích các phương pháp luận tiêu chuẩn như PTES, NIST SP 800-115 và OSSTMM, chương này đã xác lập được các quy trình chuẩn mực cần tuân thủ khi thực hiện một cuộc kiểm thử xâm nhập.

Đồng thời, nội dung chương cũng đã làm rõ sự dịch chuyển từ kiểm thử thủ công sang Automated Pentesting. Mặc dù tự động hóa mang lại ưu điểm vượt trội về tốc độ, tính nhất quán và khả năng mở rộng quy mô, nhưng các công cụ hiện tại vẫn tồn tại những hạn chế cốt lõi: sự rời rạc trong quy trình, thiếu khả năng phân tích ngữ cảnh và hoàn toàn "mù lòa" trước các lỗ hổng logic nghiệp vụ. Những thách thức này chính là động lực để nghiên cứu các giải pháp tự động hóa thông minh hơn, sẽ được phân tích chi tiết trong các chương tiếp theo.

## CHƯƠNG 2: CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN

### 2.1. Các công cụ PT truyền thống

Trong hơn hai thập kỷ qua, Dynamic Application Security Testing (DAST) đã đóng vai trò là trụ cột của tự động hóa pentest. Các công cụ như Burp Suite Professional, OWASP ZAP, Acunetix và Nessus hoạt động dựa trên phương pháp "hộp đen" (black-box), tương tác với ứng dụng đang chạy từ bên ngoài mà không cần truy cập vào mã nguồn. Cơ chế cốt lõi của chúng dựa trên chu trình: Thu thập thông tin → Kiểm thử kiểu fuzzing → Phân tích phản hồi.

Cơ chế crawling của DAST gặp khó khăn khi đối diện với các ứng dụng hiện đại. Khả năng phát hiện lỗ hổng của một máy quét DAST phụ thuộc trực tiếp vào khả năng "nhìn thấy" ứng dụng của nó. Các máy quét truyền thống sử dụng trình thu thập thông tin dựa trên HTTP để phân tích HTML tĩnh và tìm các liên kết. Tuy nhiên, sự bùng nổ của các ứng dụng Single Page Application (SPA) sử dụng React, Vue hay Angular đã làm tê liệt các trình thu thập thẻ cũ này, do nội dung chỉ được tải thông qua JavaScript và AJAX.

Để giải quyết, các công cụ hiện đại như Burp Suite hay OWASP ZAP đã tích hợp trình thu thập dựa trên trình duyệt (Browser-based crawlers/AJAX Spiders). Ví dụ, Acunetix sử dụng công nghệ DeepScan, thực thi JavaScript trong một trình duyệt headless để xây dựng mô hình Document Object Model (DOM) đầy đủ trước khi quét. Dẫu vậy, các nghiên cứu benchmark năm 2024 cho thấy độ bao phủ vẫn là một vấn đề lớn [5]. Trong một thử nghiệm so sánh, Burp Suite thể hiện khả năng vượt trội trong việc xử lý các trạng thái ứng dụng phức tạp còn ZAP cung cấp một giải pháp cân bằng tốt cho các quy trình CI/CD nhờ khả năng script hóa mạnh mẽ [6].

Một trong những rào cản lớn nhất của DAST truyền thống là tỷ lệ dương tính giả cao, buộc các kỹ sư bảo mật phải tốn hàng giờ để xác minh thủ công các cảnh báo.

Các công cụ hoạt động dựa trên việc so sánh phản hồi HTTP với cơ sở dữ liệu chữ ký - signatures-based. Ví dụ, nếu một payload SQL injection (như ' OR 1=1) trả về mã lỗi 500 hoặc một chuỗi văn bản cụ thể, scanner sẽ đánh dấu là lỗ hổng. Tuy nhiên, cơ chế này dễ bị đánh lừa bởi các trang lỗi tùy chỉnh (custom error pages, luôn trả về mã 200) hoặc các cơ chế bảo vệ như WAF.

Để giải quyết vấn đề này, một số công cụ thương mại như Invicti và Acunetix đã phát triển công nghệ "Proof-based Scanning" [5]. Thay vì chỉ dựa vào chữ ký phản hồi, công cụ sẽ cố gắng thực hiện một cuộc khai thác an toàn (ví dụ: thực sự trích xuất tên cơ sở dữ liệu hoặc thực thi một phép toán) để xác nhận lỗ hổng tồn tại 100%. Nghiên cứu so sánh cho thấy các công cụ có cơ chế xác minh này giảm thiểu đáng kể thời gian triage so với các công cụ mã nguồn mở thuận tay.

*Bảng 1: So sánh hiệu năng và đặc điểm của các framework DAST hàng đầu*

Tiêu chí	Burp Suite Pro	OWASP ZAP	Acunetix	Nessus
<b>Loại hình</b>	Thương mại	Mã nguồn mở	Thương mại	Thương mại
<b>Cơ chế crawling</b>	Hybrid (Browser + HTTP), xử lý SPA tốt	AJAX Spider (Selenium), mạnh mẽ nhưng chậm	DeepScan (Headless Chrome)	Crawler mạng và web cơ bản

<b>Độ chính xác</b>	Cao nhất trong các thử nghiệm phức tạp	Tốt, nhưng cần cấu hình thủ công nhiều	Cao nhờ Proof-based Scanning	Tốt cho hạ tầng, trung bình cho Web App
<b>Tỷ lệ dương tính giả</b>	Trung bình - Thấp	Trung bình - Cao (nếu cấu hình mặc định)	Rất thấp (Proof-based)	Trung bình
<b>Khả năng tích hợp</b>	Mạnh (BApp Store phong phú)	Rất mạnh (API, CI/CD pipeline)	Mạnh (Tích hợp Issue Trackers)	Rất mạnh (Quét diện rộng)

Mặc dù DAST rất hiệu quả trong việc phát hiện các lỗ kỹ thuật bề mặt - Technical Vulnerabilities như SQL Injection hay Cross-Site Scripting (XSS), chúng hoàn toàn bất lực trước các lỗ hổng logic nghiệp vụ - Business Logic Vulnerabilities. Lỗ hổng logic nghiệp vụ phát sinh không phải do lỗi trong mã nguồn hay thư viện mà là do sai sót trong quy trình thiết kế hoặc triển khai luồng nghiệp vụ.

- Ví dụ: Sự cố hệ thống đặt vé của Amadeus năm 2019 [7]. Kẻ tấn công có thể thay đổi vé của người khác chỉ bằng cách thay đổi mã PNR 6 ký tự trên URL. Đối với một framework DAST như ZAP hay Burp, gói tin HTTP này hoàn toàn hợp lệ: cú pháp đúng, máy chủ trả về mã 200 OK. Scanner không có khả năng hiểu ngữ cảnh xã hội và nghiệp vụ rằng "người dùng A không được phép xem dữ liệu của người dùng B".

- Lỗ hổng kiểm soát truy cập (Broken Access Control): Tương tự, các công cụ DAST gặp khó khăn cực lớn trong việc phát hiện lỗ hổng Insecure Direct Object Reference (IDOR). Để phát hiện IDOR, một công cụ cần phải hiểu khái niệm về "phiên làm việc" (session) và "quyền sở hữu dữ liệu". Nó cần đăng nhập bằng tài khoản A, cố gắng truy cập tài nguyên của tài khoản B, và phân tích xem hệ thống có từ chối hay không. Dù một số plugin của Burp Suite (như AuthMatrix) hỗ trợ việc này nhưng nó đòi hỏi cấu hình thủ công phức tạp và không thể tự động hóa hoàn toàn trên quy mô lớn.

Các công cụ truyền thống thiếu khả năng suy luận về trạng thái - statefulness và mối quan hệ logic trong luồng nghiệp vụ. Đây chính là động lực lớn thúc đẩy sự chuyển dịch sang các phương pháp tiếp cận dựa trên dữ liệu và trí tuệ nhân tạo.

## 2.2. Ứng dụng Machine Learning và Deep Learning trong PT

Việc ứng dụng machine learning (ML) và deep learning (DL) vào PT là hướng nghiên cứu đầy hứa hẹn, nhằm mục tiêu trang bị cho các hệ thống tự động khả năng "tự duy" và "học hỏi" giống như con người. Các nghiên cứu này chủ yếu sử dụng phương pháp phân tích “white-box” để phát hiện các lỗ hổng bảo mật trong mã nguồn của ứng dụng.

Thách thức lớn nhất khi áp dụng ML/DL vào mã nguồn là việc chuyển đổi source code thành dạng dữ liệu số học mà mạng nơ-ron có thể xử lý đồng thời bảo toàn được ý nghĩa ngữ nghĩa của nó. Các phương pháp ban đầu coi mã nguồn như văn bản tự nhiên - Natural Language Processing (NLP), sử dụng các mô hình như LSTM hay RNN. Tuy nhiên, mã nguồn có cấu trúc phi tuyến tính và phụ thuộc lẫn nhau rất chặt chẽ. Việc coi nó như một chuỗi văn bản thuận túy sẽ làm mất đi các thông tin quan trọng về luồng dữ liệu. Các nghiên cứu tiên phong như VulDeePecker [11] đã giới thiệu khái niệm "Code Gadgets" - bản chất là các đoạn

mã tập trung vào ngữ nghĩa, nhóm các dòng lệnh có liên quan đến nhau về mặt dữ liệu (ví dụ: nơi biến được khai báo và nơi nó được sử dụng trong hàm strcpy) thay vì đọc toàn bộ file.

Tiếp nối VulDeePecker, framework SySeVR [12] đã nâng cao độ chính xác bằng cách kết hợp cả thông tin cú pháp và ngữ nghĩa vào các biểu diễn vector.. Các thử nghiệm trên bộ dữ liệu tổng hợp (SARD) và thực tế (NVD) cho thấy SySeVR đạt độ chính xác lên tới 98% trong một số trường hợp cụ thể, vượt xa các công cụ phân tích mã nguồn tĩnh - SAST truyền thống như Flawfinder hay RATS.

Bước nhảy vọt thực sự trong lĩnh vực này đến từ việc ứng dụng mạng nơ-ron đồ thị (GNN). Mã nguồn về bản chất có thể được biểu diễn dưới dạng đồ thị. Framework Devign [13] là một ví dụ điển hình cho kiến trúc này. Thay vì đọc mã theo dòng, Devign xây dựng một đồ thị hỗn hợp - Composite Graph bao gồm:

1. AST (Abstract Syntax Tree): Biểu diễn cấu trúc ngữ pháp của mã.
2. CFG (Control Flow Graph): Biểu diễn thứ tự thực thi của các lệnh (ví dụ: các nhánh if, vòng lặp while).
3. DFG (Data Flow Graph): Theo dõi sự di chuyển và biến đổi của dữ liệu qua các biến.
4. NCS (Natural Code Sequence): Trình tự mã tự nhiên.

Bằng cách sử dụng Gated Graph Neural Networks (GGNN), Devign có thể "học" được các mẫu lỗi hỏng phức tạp nằm rải rác trong mã nguồn mà mắt thường khó nhận thấy. Ví dụ, một lỗi hỏng Use-After-Free (UAF) thường liên quan đến việc một con trỏ được giải phóng ở một nơi nhưng lại được sử dụng ở một nơi khác rất xa trong luồng điều khiển; GNN có thể kết nối hai điểm này thông qua các cạnh của đồ thị DFG/CFG một cách hiệu quả. Các biến thể mới hơn như VulGCANet và các mô

hình dựa trên đồ thị không đồng nhất (Heterogeneous Graphs) tiếp tục cải thiện khả năng xử lý các loại quan hệ phức tạp giữa các thành phần mã [14] [15].

Mặc dù đạt được các chỉ số ấn tượng trên giấy tờ, việc áp dụng các mô hình ML/DL vào pentest thực tế gặp phải những trở ngại nghiêm trọng về mặt nền tảng lý thuyết, được gọi là hiện tượng "Clever Hans" (chú ngựa biết làm toán nhưng thực ra chỉ quan sát thái độ của người huấn luyện). Các kết quả được công bố tại USENIX Security 2024 [16] đã cho thấy rằng nhiều mô hình phát hiện lỗ hổng hàng đầu thực chất không học được bản chất của lỗ hổng. Thay vào đó, chúng học các tương quan giả và thiên kiến dữ liệu.

- Thiên kiến đặt tên biến: Các mô hình thường học rằng các biến có tên như buf, len, hay count thường liên quan đến lỗ hổng tràn bộ đệm. Nếu một đoạn mã an toàn bị đổi tên biến thành buf, mô hình có thể báo cáo sai là có lỗ hổng (dương tính giả).
- Không phân biệt được bản vá: Đáng nói hơn, các mô hình này thường thát bại trong việc phân biệt giữa một hàm chứa lỗ hổng và phiên bản đã được vá của chính hàm đó, nếu cấu trúc tổng thể không thay đổi nhiều. Điều này chứng tỏ mô hình đang "nhìn" vào bối cảnh xung quanh thay vì logic của chính lỗ hổng đó.
- Vẫn đề khái quát hóa: Các mô hình được huấn luyện trên dữ liệu tổng hợp (như SARD) thường hoạt động rất kém trên mã nguồn thực tế do sự khác biệt về phân phối dữ liệu.

Trong khi DL tập trung vào việc nhìn (phát hiện), Reinforcement Learning (RL) tập trung vào việc hành động (khai thác). Nhiều nghiên cứu đã đề xuất sử dụng RL để xây dựng các tác nhân có khả năng tự học hỏi và tìm ra các chuỗi tấn công tối ưu trong một môi trường mạng.

Ghanem và Chen (2018) [8] đã mô hình hóa bài toán PT như một quá trình quyết định Markov quan sát được một phần (POMDP) và sử dụng RL để tìm ra chính sách hành động tối ưu, cho phép agent tự động thích ứng với sự thay đổi của môi trường. Clintswood và cộng sự (2023) [9] cũng đề xuất một quy trình sử dụng thuật toán RL, được huấn luyện với framework MITRE ATT&CK, để tạo ra một công cụ PT tự động. Hướng tiếp cận này cho thấy tiềm năng to lớn trong việc tự động hóa giai đoạn lập kế hoạch tấn công.

Framework DeepExploit [10] là một nỗ lực tiên phong trong việc sử dụng RL để tự động hóa giai đoạn khai thác của PT. DeepExploit mô hình hóa PT dưới dạng một Markov Decision Process - MDP:

- Trạng thái - State: Thông tin về mục tiêu (OS, Port, Services).
- Hành động - Action: Chọn và cấu hình payload từ Metasploit Framework.
- Phần thưởng - Reward: Nhận điểm dương nếu mở được reverse shell, điểm âm nếu thất bại.

Qua việc “self-play” hàng nghìn lần, DeepExploit có thể học được các chuỗi tấn công tối ưu. Tuy nhiên, các mô hình RL thường đối mặt với thách thức về state/action space explosion trong các hệ thống thực tế, đòi hỏi tài nguyên tính toán khổng lồ và thời gian huấn luyện dài.

### 2.3. Các framework sử dụng AI Agent trong PT

Sự ra đời của các mô hình ngôn ngữ lớn (LLM) như GPT, Claude hay Gemini đã mở ra giai đoạn thứ hai của tự động hóa: Tự động hóa nhận thức - Cognitive Automation. Khác với các mô hình ML/DL chuyên biệt, LLM có khả năng suy luận tổng quát, lập kế hoạch và sử dụng công cụ. Điều này cho phép chúng hoạt động như các "tác nhân AI" (AI Agents) tự chủ.

PentestGPT [17] là framework tiêu biểu cho thế hệ đầu tiên của AI pentesting. Nó không cố gắng thay thế hoàn toàn con người mà đóng vai trò như một người dẫn đường thông minh. Kiến trúc của PentestGPT được chia thành ba module chính để giải quyết vấn đề mất ngữ cảnh thường gặp ở LLM:

1. Reasoning Module: Hoạt động như bộ não để giúp duy trì một "cây trạng thái" cấp cao của quá trình pentest. Nó phân tích tình hình hiện tại và quyết định bước đi chiến lược tiếp theo (ví dụ: "Cần quét kỹ hơn port 8080").
2. Generation Module: Phụ trách chuyển đổi chiến lược thành các lệnh kỹ thuật cụ thể (ví dụ: tạo câu lệnh nmap với các cờ tối ưu hoặc viết script Python để khai thác SQLi).
3. Parsing Module: Đọc kết quả thô từ terminal (thường rất dài và nhiều), tóm tắt các thông tin quan trọng (như open ports, version numbers) và đưa trở lại Reasoning Module.

Trong các thử nghiệm trên nền tảng HackTheBox và các giải CTF, PentestGPT đã chứng minh hiệu quả vượt trội, giúp tăng tỷ lệ hoàn thành tác vụ lên 228.6% so với việc sử dụng ChatGPT thuận túy [17]. Tuy nhiên, nó vẫn yêu cầu người dùng phải copy-paste lệnh và kết quả qua lại, tạo ra độ trễ và gián đoạn.

Khắc phục hạn chế của PentestGPT, AutoPentester [18] và các hệ thống thương mại như XBOW [19] hướng tới mô hình tự chủ hoàn toàn. AutoPentester sử dụng kiến trúc đa tác nhân (Multi-Agent System - MAS), nơi các agent chuyên biệt phối hợp với nhau: Reconnaissance Agent - chuyên trách thu thập thông tin, Exploitation Agent - chuyên trách thực hiện tấn công và Reporting Agent - tổng hợp báo cáo. Điểm đột phá của AutoPentester nằm ở các thành phần hỗ trợ:

- Agent-Computer Interface: Cho phép Agent có thể trực tiếp tương tác với shell của hệ điều hành, thực thi lệnh và đọc kết quả mà không cần con người can thiệp.
- RAG-based Generator: Sử dụng kỹ thuật Retrieval-Augmented Generation để truy xuất kiến thức chính xác về cú pháp công cụ và các CVE mới nhất từ cơ sở dữ liệu véc-tơ, giảm thiểu ảo giác.
- Repetition Identifier: Một module giám sát để phát hiện khi Agent bị kẹt trong vòng lặp vô hạn (ví dụ: thử đi thử lại một mật khẩu sai) - một vấn đề phổ biến của các LLM Agent.

Kết quả thực nghiệm cho thấy AutoPentester đạt tỷ lệ bao phủ lỗ hổng cao hơn 39.5% so với PentestGPT và có khả năng phát hiện các lỗ hổng phức tạp mà không cần bất kỳ sự can thiệp nào của con người [18].

Một trong những thách thức lớn nhất của AI Agent là khả năng xử lý các chuỗi tấn công dài. Một cuộc tấn công thực tế có thể yêu cầu hàng chục bước liên tiếp: Recon → Scan → Initial Access → Privilege Escalation → Pivot. Nếu Agent sai lầm ở bước 1 sẽ dẫn đến toàn bộ chuỗi thất bại. Framework CurriculumPT [20] giải quyết vấn đề này bằng phương pháp "học theo giáo trình". Thay vì ném Agent vào một bài toán khó ngay lập tức, hệ thống huấn luyện Agent qua các bài tập có độ khó tăng dần: từ khai thác lỗ hổng đơn lẻ đến các chuỗi tấn công đa giai đoạn. Hơn nữa, CurriculumPT tích hợp một "cơ sở tri thức động" - Dynamic Experience Knowledge Base, cho phép các Agent ghi nhớ các chiến lược thành công và thất bại để áp dụng cho các mục tiêu mới. Cách tiếp cận này giúp cải thiện tỷ lệ thành công lên tới 18% so với các baseline hiện tại và giảm đáng kể thời gian thực thi [20].

## 2.4. Xác định vấn đề cần nghiên cứu

Để cung cấp cái nhìn tổng quan, bảng 2 so sánh ba thể hệ công nghệ dựa trên các tiêu chí cốt lõi:

Bảng 2. So sánh các công nghệ tự động hóa PT qua các thể hệ

Đặc điểm	DAST	Deep Learning	AI Agent
<b>Phạm vi Phát hiện</b>	Lỗ hổng kỹ thuật bề mặt	Lỗ hổng mã nguồn	Lỗ hổng logic, chuỗi tấn công phức tạp
<b>Ngữ cảnh</b>	Không có	Cục bộ	Toàn cục
<b>Cơ chế hoạt động</b>	Rule-based & signature matching	Pattern Recognition	Reasoning, planning & tool orchestration
<b>Khả năng giải thích</b>	Rất cao	Rất thấp	Cao
<b>Mức độ tự chủ</b>	Tự động hóa tác vụ	Tự động hóa phân loại	Tự chủ ra quyết định
<b>Điểm yếu</b>	Bỏ sót hoàn toàn logic nghiệp vụ	Overfitting, dữ liệu không “sạch”	Ảo giác, chi phí cao, vòng lặp

Có thể thấy bức tranh toàn cảnh về tự động hóa trong kiểm thử xâm nhập đang tồn tại những mảnh ghép rời rạc. DAST truyền thống mạnh về tốc độ và độ bao phủ bề mặt nhưng thiếu tư duy ngữ cảnh. Các mô hình ML/DL có khả năng phát hiện

mẫu mã nguồn tốt nhưng gặp khó khăn về tính khái quát hóa và dữ liệu. Trong khi đó, các AI Agent thế hệ mới tuy có khả năng suy luận nhưng lại thiếu sự ổn định khi phải thực hiện các chuỗi thao tác dài và phức tạp.

Vấn đề đặt ra cho nghiên cứu này không phải là lựa chọn thay thế hoàn toàn một công nghệ nào mà là làm thế nào để dung hòa và tận dụng ưu điểm của chúng trong một kiến trúc thống nhất. Dựa trên cơ sở đó, nghiên cứu này xác định hai vấn đề cốt lõi cần phải giải quyết:

*a) Tích hợp và điều phối công cụ truyền thông*

Các framework AI hiện tại thường có xu hướng “reinvent the wheel” bằng cách cố gắng để LLM tự sinh ra các payload tấn công, trong khi các công cụ chuyên dụng (như SQLMap, Burp Active Scan) đã làm điều này rất tốt trong hàng chục năm qua. Việc AI cố gắng thay thế hoàn toàn vai trò của các scanner thường dẫn đến hiệu suất kém và thiếu chính xác.

Do đó, vấn đề nghiên cứu được xác định là: làm thế nào để xây dựng một AI Agent đóng vai trò là "nhạc trưởng" thay vì là "nhạc công"? Tại sao phải tự gửi từng gói tin HTTP, điều này là không hề hiệu quả. Chính vì thế, AI Agent trong framework AiPT sẽ được thiết kế để:

- Hiểu ngữ nghĩa của mục tiêu và lựa chọn công cụ quét phù hợp.
- Tự động cấu hình tham số và điều khiển các scanner truyền thông thông qua công nghệ Model Context Protocol (MCP).
- Phân tích kết quả đầu ra từ các công cụ này, loại bỏ nhiễu và sử dụng chúng làm đầu vào cho các bước suy luận tiếp theo.

Hướng tiếp cận này đảm bảo cho framework vẫn giữ được độ tin cậy và security baseline của DAST trong khi đồng thời bổ sung thêm lớp trí tuệ để xử lý logic.

### b) Mô hình cộng tác Human - AI

Một rào cản lớn của các hệ thống tự động hoàn toàn là tỷ lệ dương tính giả và nguy cơ thực hiện các hành động gây hại cho hệ thống đích (như DoS vô ý hoặc xóa dữ liệu). Hơn nữa, AI hiện tại vẫn chưa thể thay thế trực giác và tư duy sáng tạo của con người trong việc phát hiện các lỗi logic nghiệp vụ phức tạp.

Vì vậy, nhóm nghiên cứu xác định hướng đi không phải là tạo ra một "pentester nhân tạo" thay thế con người mà là một "trợ lý thông minh". Mô hình cộng tác được đề xuất sẽ giải quyết các vấn đề:

- Giảm tải: AI Agent đảm nhận các tác vụ nhảm chán, lặp đi lặp lại và tiêu tốn nhiều thời gian như thu thập thông tin, rà quét cổng và kiểm tra các lỗi cơ bản.
- Tăng cường khả năng của con người: Pentester sẽ được giải phóng khỏi các tác vụ cấp thấp để tập trung toàn bộ năng lượng vào việc phân tích các lỗi logic, leo thang đặc quyền phức tạp và đánh giá rủi ro kinh doanh - những khu vực mà AI chưa thể chạm tới.
- Cơ chế kiểm soát: Con người đóng vai trò là người ra quyết định cuối cùng ở các chốt chặn quan trọng, xác minh các phát hiện của AI và định hướng chiến lược tấn công khi AI gặp bế tắc.

Nói tóm lại, mục tiêu của đề tài này là xây dựng framework AiPT dựa trên kiến trúc AI Agent có khả năng điều phối các công cụ kiểm thử truyền thống, hoạt động theo mô hình phối hợp Human-in-the-loop nhằm tối ưu hóa hiệu suất và độ chính xác của quy trình kiểm thử xâm nhập web.

## 2.5. Tổng kết chương

Chương này đã tiến hành khảo sát và đánh giá các công trình nghiên cứu liên quan theo tiến trình phát triển của công nghệ kiểm thử tự động, từ các công cụ truyền thống đến các ứng dụng AI hiện đại:

1. Công cụ truyền thống (DAST): Được xác định là nền tảng vững chắc cho việc phát hiện các lỗ hổng kỹ thuật bề mặt, tuy nhiên hạn chế lớn nhất là thiếu tư duy ngữ cảnh và không thể xử lý các ứng dụng web phức tạp (SPA) hoặc logic nghiệp vụ.
2. Machine Learning & Deep Learning: Việc áp dụng ML/DL vào mã nguồn (White-box) và khai thác (RL) đã cho thấy tiềm năng nhưng vẫn gặp trở ngại về tính khái quát hóa, thiên kiến dữ liệu và hiện tượng "Clever Hans".
3. AI Agent: Sự xuất hiện của các framework như PentestGPT và AutoPentester đánh dấu bước ngoặt sang "tự động hóa nhận thức" với khả năng suy luận và lập kế hoạch. Tuy nhiên, độ ổn định và khả năng duy trì ngữ cảnh trong các chuỗi tấn công dài vẫn là thách thức cần giải quyết.

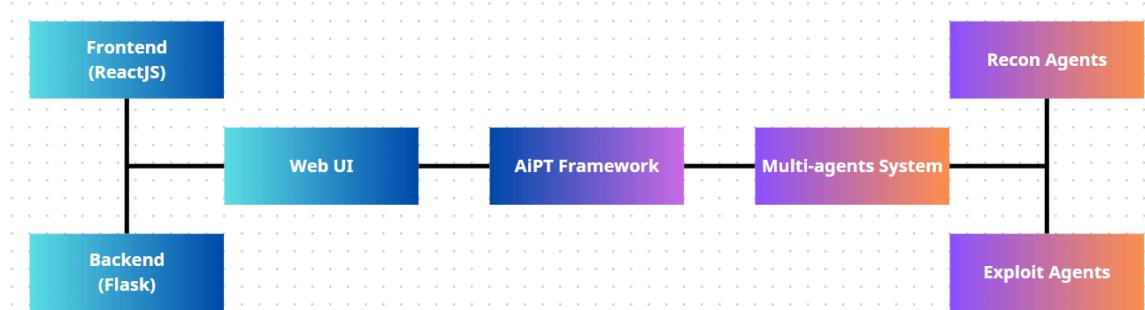
Từ sự so sánh và đối chiếu các phương pháp trên (bảng 2), chương này đã xác định được khoảng trống nghiên cứu và đề xuất hướng tiếp cận cho framework AiPT. Đó là sự kết hợp giữa khả năng suy luận của AI Agent với độ tin cậy của các scanner truyền thống, hoạt động dưới sự giám sát và hỗ trợ của con người.

## CHƯƠNG 3: KIẾN TRÚC HỆ THỐNG VÀ TRIỂN KHAI

### 3.1. Kiến trúc tổng thể

Để khắc phục các hạn chế về tính linh hoạt của các công cụ rà quét truyền thống và giải quyết bài toán điều phối phức tạp trong các hệ thống AI, framework AiPT được xây dựng dựa trên sự kết hợp chiến lược giữa kiến trúc hướng dịch vụ (Service-Oriented Architecture - SOA) và mô hình AI đa tác nhân (AI Multi-Agent System - MAS). Thiết kế này đảm bảo sự phân tách rành mạch giữa các thành phần: giao diện tương tác, xử lý nghiệp vụ và lối trí tuệ nhân tạo, từ đó tối ưu hóa khả năng mở rộng, bảo trì và tính modular của hệ thống.

Mô hình kiến trúc tổng thể (xem hình 4) được tổ chức thành ba tầng chức năng chính, tương tác chặt chẽ với nhau thông qua các giao thức chuẩn hóa:



Hình 3.1: Sơ đồ kiến trúc tổng thể của hệ thống AiPT

#### a) Giao diện và dịch vụ

Tầng này đóng vai trò là lớp tương tác Human-AI, giúp đảm bảo quy trình kiểm thử sẽ được kiểm soát chặt chẽ bởi con người:

- Giao diện (Frontend): Được phát triển trên nền tảng ReactJS, module này cung cấp một bảng điều khiển tập trung giúp trực quan hóa dữ liệu. Tại đây, pentester sẽ thực hiện: định nghĩa phạm vi mục tiêu, thiết lập các luồng làm

việc với workflow, giám sát tiến trình rà quét theo thời gian thực và phân tích danh sách các bẻ mặt tấn công (cụ thể là API) được hệ thống phát hiện.

- Dịch vụ (Backend): Sử dụng Flask làm nền tảng, phân hệ này hoạt động như một API gateway trung tâm. Nhiệm vụ chính của nó bao gồm xử lý các yêu cầu HTTP từ Frontend, quản lý tính toàn vẹn của cơ sở dữ liệu và kích hoạt các module chức năng bên trong AiPT Framework.

#### b) Hệ thống AI Agent

Đây là trung tâm xử lý và ra quyết định của toàn bộ hệ thống, được kiến tạo dựa trên Google Agent Development Kit (ADK) kết hợp với framework CrewAI. Tầng này chịu trách nhiệm khởi tạo, quản lý vòng đời và điều phối cơ chế cộng tác giữa các tác nhân AI.

Hệ thống sử dụng LLM Gemini-2.5-Flash làm mô hình suy luận chính. Việc lựa chọn Gemini-2.5-Flash dựa trên hai ưu điểm kỹ thuật quan trọng: khả năng xử lý ngữ cảnh dài (context window 1 triệu token) - thiết yếu cho việc phân tích mã nguồn và log và độ trễ thấp - tối ưu cho các tác vụ thời gian thực. Trong kiến trúc MAS của AiPT, các tác nhân được chuyên biệt hóa thành hai nhóm chức năng:

1. Recon Agents: Đảm nhiệm vai trò thu thập thông tin tình báo, rà quét bẻ mặt tấn công và định danh các API endpoints.
2. Exploit Agents: Thực hiện phân tích chuyên sâu, lập kế hoạch tấn công và thực thi các trường hợp kiểm thử (test-case) dựa trên các kịch bản (workflow) đã được xác định.

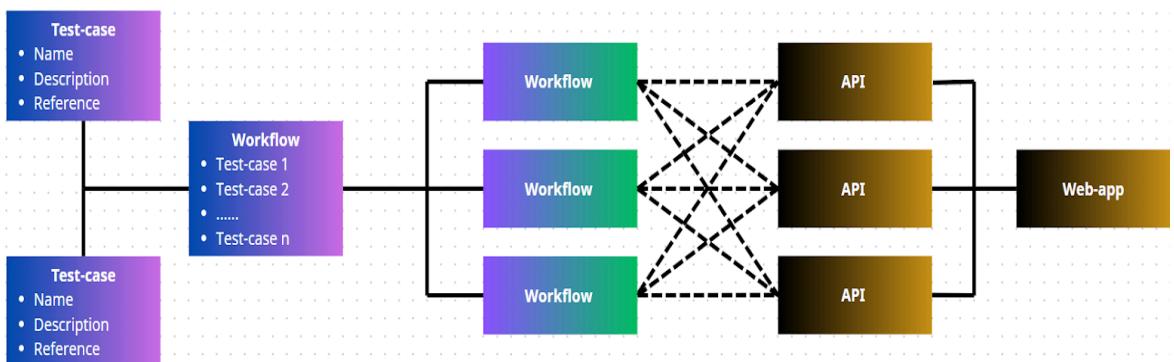
#### c) Cơ chế giao tiếp và tích hợp công cụ

Framework AiPT tích hợp các công cụ kiểm thử truyền thống (như Burp Suite, Sqlmap, Nuclei) thông qua giao thức ngữ cảnh mô hình - Model Context Protocol

(MCP). Thay vì để LLM tự sinh ra các câu lệnh (để dẫn đến sai cú pháp hoặc ảo giác), MCP sẽ hoạt động như một lớp trừu tượng hóa. Nó chuẩn hóa các chức năng phức tạp của công cụ bảo mật thành các "tools" định nghĩa sẵn, cho phép các Agent gọi và thực thi một cách chính xác và an toàn. Kiến trúc này cho phép AiPT tận dụng sức mạnh suy luận linh hoạt của AI đồng thời vẫn đảm bảo độ tin cậy và chính xác của các scanner truyền thống.

### 3.2. Các thuật ngữ quan trọng trong AiPT

Để vận hành quy trình kiểm thử tự động một cách hiệu quả và có kiểm soát, AiPT xây dựng một hệ thống các khái niệm cốt lõi nhằm chuẩn hóa dữ liệu đầu vào và quy trình thực thi. Các khái niệm này định hình cách thức người dùng giao tiếp với hệ thống và cách các Agent hiểu nhiệm vụ của mình.



Hình 3.2: Mô hình quan hệ giữa workflow, test-case và API endpoints

#### a) Đơn vị kiểm thử cơ sở (test-case)

Trong AiPT, test-case được định nghĩa là đơn vị kiểm thử cơ sở (atomic unit) đại diện cho một kỹ thuật tấn công cụ thể hoặc một bước kiểm thử chuyên biệt. Thay vì để AI tự do sinh ra các kịch bản tấn công, mỗi test-case được cấu trúc hóa chặt chẽ với các thuộc tính như sau:

- Name: Tên kỹ thuật tấn công (ví dụ: "Brute-force authen credentials").

- Description: Mô tả chi tiết về phương thức thực thi và mục tiêu cần đạt được của bài kiểm thử.
- Reference: Liên kết đến các cơ sở tri thức bảo mật tiêu chuẩn (như OWASP Top 10, CWE) và các tài liệu hướng dẫn khai thác (như PayloadAllTheThings, Hack Tricks), cung cấp ngữ cảnh bổ sung cho Agent trong quá trình suy luận.

Việc modular các kỹ thuật tấn công thành các đối tượng test-case độc lập giúp tách biệt phần "tri thức" khỏi phần "logic suy luận". Điều này cho phép hệ thống dễ dàng cập nhật các lỗ hổng mới (CVEs) mà không cần can thiệp hay tái cấu trúc logic lỗi của các Agent.

#### b) Kịch bản phối hợp (workflow)

Workflow là tập hợp có trật tự của các test-case, được sắp xếp theo một kịch bản logic nhất định nhằm kiểm thử một nhóm chức năng nghiệp vụ hoặc một lớp lỗ hổng cụ thể. Trong kiến trúc AiPT, workflow đóng vai trò như một bản chỉ dẫn chiến lược cho Exploit Agents:

- Định hướng mục tiêu: Thay vì thực thi ngẫu nhiên, workflow giới hạn phạm vi các kỹ thuật tấn công phù hợp với từng API (ví dụ: workflow cho chức năng "Đăng nhập" sẽ chỉ bao gồm các test-case liên quan như Brute-force, SQL Injection, NoSQL Injection).
- Tối ưu hóa tài nguyên & Giảm thiểu rủi ro: Workflow giúp ngăn chặn việc thực thi lan man, tiết kiệm tài nguyên tính toán và đặc biệt là giảm thiểu rủi ro gây gián đoạn dịch vụ (DoS) cho hệ thống mục tiêu do gửi quá nhiều yêu cầu không cần thiết.

Qua cơ chế gán workflow cho từng API cụ thể trên Dashboard, người dùng (ở đây là pentester) luôn giữ quyền kiểm soát tối cao đối với chiến lược kiểm thử, đảm bảo AI hoạt động trong khuôn khổ an toàn và hiệu quả.

### 3.3. Cơ chế quản lý ngữ cảnh và tối ưu hóa bộ nhớ

Trong hệ thống multi-agent sử dụng LLM, thách thức lớn nhất không chỉ nằm ở khả năng suy luận mà còn ở việc quản lý “context window”. Đối với quy trình kiểm thử xâm nhập, dữ liệu thu thập được từ các công cụ (như logs, HTTP responses, scan results) thường có dung lượng rất lớn. Nếu nạp toàn bộ dữ liệu đó vào LLM sẽ dẫn đến tràn bộ nhớ (context overflow), tăng chi phí token và gây ra hiện tượng “nhiều”, điều đó khiến Agent mất tập trung vào mục tiêu chính.

Để giải quyết vấn đề này, AiPT áp dụng chiến lược quản lý ngữ cảnh đa tầng, tận dụng tối đa khả năng quản lý trong framework như CrewAI và ADK:

#### a) Truyền ngữ cảnh tuần tự và có chọn lọc

Thay vì duy trì một bộ nhớ chung không lồ cho toàn bộ phiên làm việc, AiPT thiết kế luồng dữ liệu theo cơ chế đường ống - pipeline. Output của agent trước (ví dụ: Recon Agents) sẽ được chọn lọc để trở thành context cho tác nhân kế tiếp (ví dụ: Exploit Agents).

- Cơ chế thực hiện: Hệ thống định nghĩa rõ ràng dependency giữa các tasks. Chỉ những thông tin thực sự liên quan (ví dụ: danh sách API endpoints, tham số khả nghi) mới được trích xuất và chuyển tiếp. Các thông tin dư thừa (như mã nguồn HTML tĩnh không chứa form, hình ảnh) sẽ bị loại bỏ ngay tại lớp parser trước khi đi vào LLM.

#### b) Chuẩn hóa đầu ra của công cụ

Một trong những nguyên nhân chính gây lãng phí tài nguyên tính toán là việc LLM phải đọc các kết quả thô từ terminal (thường chứa nhiều ký tự trang trí, thanh tiến trình, banner).

- Giải pháp: Các công cụ tích hợp trong AiPT thông qua giao thức MCP được thiết kế để trả về kết quả dưới dạng dữ liệu có cấu trúc (JSON hoặc Python dictionary) thay vì văn bản thô.
- Lợi ích: Dữ liệu có cấu trúc giúp LLM "hiểu" ngay lập tức các trường quan trọng (như vulnerability\_type, severity, proof) mà không cần tốn token để phân tích cú pháp, đồng thời giảm đáng kể kích thước context cần truyền tải.

#### c) *Tối ưu hóa bằng Prompt Engineering*

Mặc dù hệ thống hỗ trợ người dùng tương tác bằng tiếng Việt nhưng ở system prompts, các chỉ thị cho Agent (như role, goal, backstory) được thiết kế và tối ưu hóa bằng tiếng Anh.

- Lý do: Các mô hình LLM hiện đại (như Gemini hay GPT) được huấn luyện chủ yếu trên dữ liệu tiếng Anh, do đó khả năng token hóa tiếng Anh hiệu quả hơn và tiêu tốn ít token hơn so với tiếng Việt cho cùng một lượng thông tin. Chiến lược này giúp mở rộng tối đa không gian bộ nhớ khả dụng cho việc chứa dữ liệu kiểm thử thực tế.

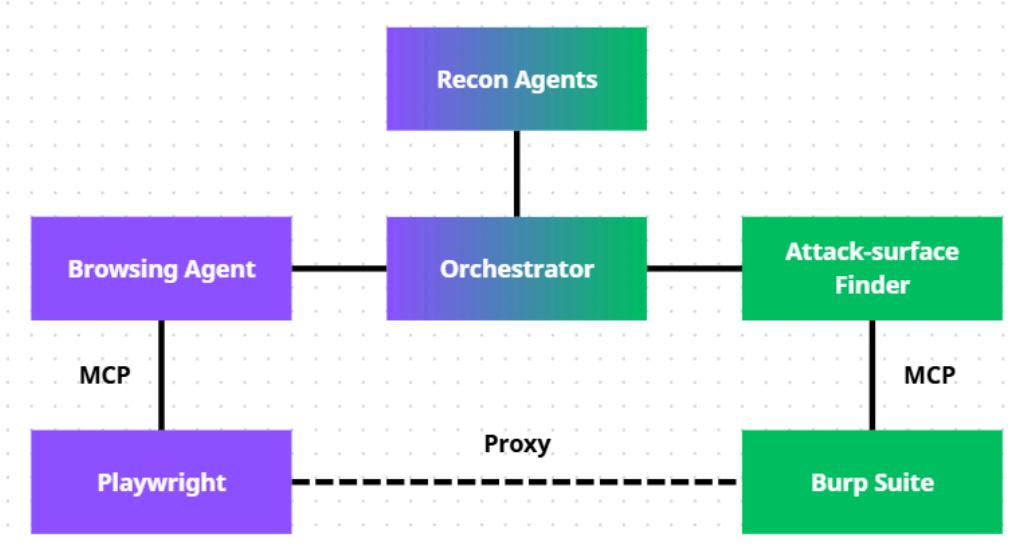
#### d) *Vai trò tổng hợp thông tin*

Để tránh việc báo cáo cuối cùng bị quá tải bởi dữ liệu kỹ thuật, một Agent chuyên biệt được bố trí ở cuối quy trình. Nhiệm vụ của Agent này không phải là tìm kiếm lỗ hổng mới, mà là "nén" toàn bộ lịch sử hoạt động và các phát hiện rời rạc từ các bước trước thành một bản tóm tắt cô đọng, loại bỏ các thông tin trùng lặp trước khi xuất ra định dạng cuối cùng.

### 3.4. Module Recon Agents

Trong quy trình kiểm thử xâm nhập tự động, giai đoạn reconnaissance đóng vai trò nền tảng, quyết định độ bao phủ và chất lượng của toàn bộ bài kiểm thử. Khắc phục nhược điểm của các trình rà quét truyền thống vốn dựa trên cơ chế phân tích liên kết tĩnh, module Recon Agents trong AiPT chuyển đổi sang mô hình mô phỏng hành vi người dùng để tương tác sâu với ứng dụng.

Kiến trúc của module này được thiết kế dựa trên sự phối hợp chặt chẽ giữa hai thành phần chính: Browsing Agent (tương tác chủ động) và Attack-surface Finder (quan sát thụ động) đặt dưới sự điều phối của một Orchestrator (xem hình 6).



Hình 3.3: Sơ đồ chi tiết hoạt động của module Recon Agents

#### a) Browsing Agent

Browsing Agent là thành phần chịu trách nhiệm tương tác trực tiếp với giao diện ứng dụng web mục tiêu. Để đạt được khả năng xử lý các ứng dụng web hiện đại phức tạp, Agent này không gửi các HTTP request mà điều khiển một trình duyệt

thực sự thông qua bộ công cụ Playwright của Microsoft. Các đặc điểm chính của Browsing Agent:

- Kết nối qua giao thức MCP: Browsing Agent không giao tiếp trực tiếp với trình duyệt bằng các lệnh cứng nhắc. Thay vào đó, Playwright được bao bọc trong một lớp giao diện MCP server. LLM (Gemini-2.5-Flash) đóng vai trò là bộ não xử lý, phân tích cấu trúc DOM (Document Object Model) và giao diện người dùng để ra quyết định hành động (click, nhập liệu, cuộn trang,...) rồi sau đó gửi lệnh thực thi qua giao thức MCP.
- Xử lý ứng dụng SPA: Nhờ khả năng "đọc hiểu" cấu trúc DOM thông qua AI, Agent có thể xử lý hiệu quả các ứng dụng SPA được xây dựng trên ReactJS, VueJS hoặc Angular - nơi nội dung được tải động qua JavaScript mà các crawler truyền thống thường bỏ sót.
- Chiến lược duyệt dựa trên ngữ nghĩa: Agent được lập trình để ưu tiên khám phá các luồng nghiệp vụ trọng yếu như đăng ký, đăng nhập, gio hàng thay vì duyệt ngẫu nhiên. Nó mô hình hóa toàn bộ ứng dụng web thành một đồ thị đơn giản như cây thư mục, đánh dấu các điểm đã đi qua hoặc chưa. Điều này là rất quan trọng vì nó giúp kiểm soát tiến độ của toàn bộ quy trình, giúp AI tránh việc “đi lạc” hoặc lặp lại các thao tác cũ.
- Xử lý yêu cầu về dữ liệu: System prompt được tối ưu hóa qua nhiều lần thử nghiệm, giúp AI có khả năng nhận diện các form và trường dữ liệu. Từ đó, hệ thống có thể xử lý các yêu cầu về dữ liệu và đúng logic nghiệp vụ của ứng dụng thay vì những ký tự ngẫu nhiên vô nghĩa. Đồng thời cũng có thể tránh bị phát hiện và ngăn cản bởi các cơ chế anti-bot như captcha. Vì Playwright cung cấp khả năng chụp ảnh màn hình và LLM được sử dụng có khả năng xử lý dữ liệu dạng ảnh nên việc vượt quá các captcha đơn giản là không khó khăn. Hiểu ngữ cảnh, biết cách điền đúng thông tin, di chuyển và sử dụng

các chức năng trên web là những ưu điểm vượt trội mà các công cụ non-AI khó làm được.

### b) *Attack-surface Finder*

Song song với quá trình tương tác trên trình duyệt của Browsing Agent, thành phần Attack-surface Finder hoạt động giống như một trạm quan trắc mạng thụ động để thu thập dữ liệu. Từ đó nó xác định và lọc ra các API endpoints - đầu vào cho các quy trình pentest web và di động. Cơ chế hoạt động diễn ra như sau:

- Sử dụng proxy: Toàn bộ lưu lượng truy cập phát sinh từ Browsing Agent được định tuyến qua một proxy trung gian tích hợp với Burp Suite. Sự trung chuyển được hỗ trợ hoàn toàn bởi Playwright.
- Tự động phát hiện API: Attack-surface Finder phân tích các cặp HTTP request-response trong thời gian thực trong tab Proxy History của Burp Suite. Kết nối giữa LLM với Burp Suite được thực hiện nhờ extension MCP Server do chính Portswigger - nhà phát hành của Burp Suite phát triển. Hệ thống kết hợp giữa so khớp mẫu và suy luận AI để:
  - Tách lọc các lời gọi API (RESTful, GraphQL) khỏi các tài nguyên tĩnh (CSS, Images,...).
  - Định danh phương thức và cấu trúc tham số (JSON Body/Parameters).
  - Mô tả chức năng của API, các lưu ý về logic nghiệp vụ của chúng.
  - Phát hiện các điểm cuối nhạy cảm có khả năng tiềm tàng lỗ hổng.

### c) *Quy trình vận hành*

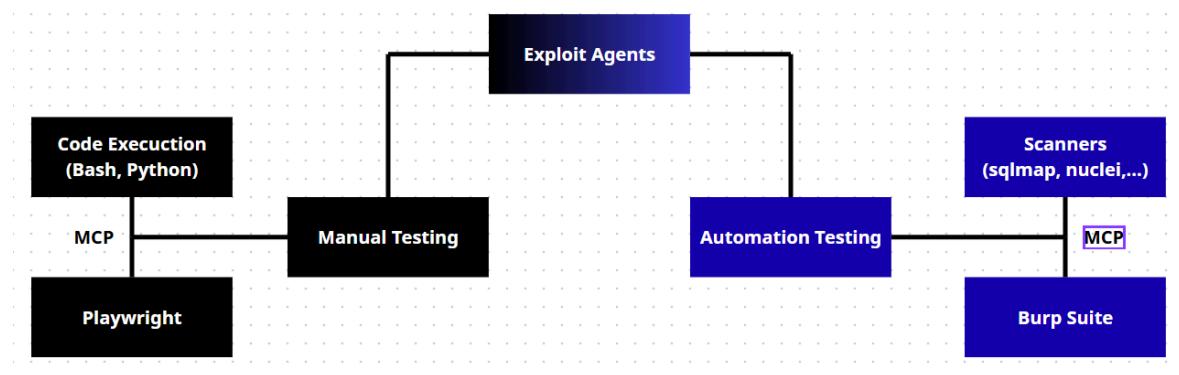
Dữ liệu thu thập từ quá trình reconnaissance không được lưu trữ cục bộ mà được đồng bộ hóa theo thời gian thực lên hệ thống trung tâm. Quy trình thực thi diễn ra qua 4 bước như sau:

1. Khởi tạo: Người dùng cung cấp thông tin mục tiêu (target domain, techstack,...) trên Web UI. Hệ thống kích hoạt Recon Agents.
2. Thực thi: Browsing Agent tiến hành duyệt web mô phỏng hành vi người dùng. Mọi traffic HTTP được Attack-surface Finder ghi lại và phân tích.
3. Cập nhật Dashboard: Ngay khi một API mới được phát hiện, thông tin chi tiết (URL, Method, HTTP request-response) được gửi về Backend và hiển thị trên list API của Dashboard.
4. Xác nhận phạm vi: Người dùng có cái nhìn toàn cảnh về bề mặt tấn công. Từ danh sách API này, người dùng thực hiện chỉ định workflow kiểm thử cụ thể cho từng API, chuẩn bị đầu vào chính xác cho giai đoạn tấn công tiếp theo.

Quy trình này đảm bảo tính chính xác cao của dữ liệu đầu vào cho Exploit Agents, đồng thời cho phép người dùng kiểm soát hoàn toàn phạm vi kiểm thử.

### 3.5. Module Exploit Agents

Nếu module Recon Agents đóng vai trò là hệ thống quan sát để định vị mục tiêu, thì module Exploit Agents được thiết kế như "attack engine" chủ lực của framework AiPT. Module này chịu trách nhiệm thực hiện các bài kiểm thử xâm nhập dựa trên ngữ cảnh cụ thể của ứng dụng. Hoạt động này được vận hành theo nguyên lý định hướng mục tiêu (target-oriented) thay vì rà quét diện rộng (spray-and-pray) như các công cụ tự động truyền thống DAST. Hình 7 mô tả cấu trúc của Exploit Agents.



Hình 3.4: Kiến trúc chi tiết của module Exploit Agents

Sự khác biệt cốt lõi của Exploit Agents nằm ở khả năng "hiểu" nghiệp vụ trước khi hành động. Quy trình xử lý đầu vào diễn ra như sau:

- Đầu vào: Agent tiếp nhận cặp dữ liệu gồm thông tin kỹ thuật của API (URL, Method, HTTP request-response) và workflow kiểm thử do người dùng chỉ định trước đó.
- Phân tích ngữ nghĩa: Sử dụng năng lực suy luận của mô hình Gemini-2.5-Flash để phân tích cấu trúc API, từ đó xác định ngữ nghĩa của các tham số. Ví dụ: với API /api/login, hệ thống tự động nhận diện username và password là các injection points cho các kịch bản tấn công xác thực thay vì thử nghiệm mù quáng các payload XSS vô nghĩa vào các trường này.

Quá trình kiểm thử một test-case không diễn ra đơn tuyến mà tuân theo một vòng lặp phản hồi khép kín gồm 3 bước, đảm bảo tính thích ứng cao:

- Lập kế hoạch: Căn cứ vào định nghĩa của test-case (ví dụ: "SQL Injection Time-based"), Agent xây dựng chiến lược tấn công và lựa chọn loại payload phù hợp nhất với công nghệ đằng sau của mục tiêu.

2. Thực thi qua MCP: Agent không trực tiếp tạo gói tin mà hoạt động như một nhạc trưởng, điều phối các công cụ chuyên dụng thông qua giao thức MCP.
  - Khả năng thực thi Python/Bash để sinh mã khai thác (PoC) tùy chỉnh.
  - Điều khiển các scanners (như sqlmap, nuclei,...) với các tham số đã được tối ưu hóa cho ngữ cảnh hiện tại.
3. Đánh giá phản hồi: Agent phân tích kết quả trả về bao gồm HTTP status code, thời gian phản hồi và nội dung phản hồi.
  - Nếu phát hiện dấu hiệu lỗ hổng (ví dụ: độ trễ phản hồi khớp với payload time-based), kết quả được ghi nhận.
  - Nếu bị chặn (như WAF trả về 403), Agent tự động điều chỉnh chiến lược bypassing hoặc dừng lại để tránh rủi ro bị khóa IP/account.

Để đảm bảo tính toàn vẹn và an toàn cho hệ thống mục tiêu, Exploit Agents tuân thủ nghiêm ngặt các nguyên tắc vận hành:

- Xử lý tuần tự và kế thừa: Các test-case trong một workflow được thực hiện tuần tự. Kết quả đầu ra của bước trước được sử dụng làm đầu vào cho bước sau (ví dụ: token thu được từ test-case "Đăng nhập" sẽ được tự động chèn vào header cho test-case kế tiếp).
- Human-in-the-loop: Nhằm giảm thiểu rủi ro AI hoạt động mất kiểm soát, AiPT tích hợp các chốt chặn an toàn:
  - Agent chỉ được phép tấn công vào các API đã được người dùng xác nhận (đã gán workflow).
  - Các lệnh thực thi nguy hiểm mang tính phá hủy (như DROP TABLE, rm -rf) sẽ bị bộ lọc tự động loại bỏ hoặc yêu cầu phê duyệt thủ công.

- Mọi hành động đều được ghi logs chi tiết để phục vụ tra soát về sau.

### 3.6. Cơ chế tích hợp công cụ và chiến lược

#### a) Quản lý payloads tập trung

Thay vì phụ thuộc hoàn toàn vào việc AI tự sinh ra các payload tấn công (vốn có thể sai cú pháp hoặc thiếu hiệu quả), nhóm nghiên cứu đã xây dựng một thư viện payload chuyên biệt (nằm trong thư mục payloads).

- Cơ chế hoạt động:
  - Các Exploit Agent không tự viết lại payload từ đầu cho mỗi lần tấn công. Thay vào đó, chúng truy xuất các payload mẫu (như Polyglot XSS, Time-based SQLi) từ kho dữ liệu nội bộ.
  - Nhiệm vụ của AI là lựa chọn payload phù hợp nhất với tech stack của mục tiêu và tinh chỉnh nếu cần thiết.
- Ưu điểm: Cách tiếp cận này đảm bảo chất lượng và độ tin cậy của các bài kiểm tra, giảm thiểu tỷ lệ dương tính giả do payload bị lỗi cú pháp.

#### b) Chuẩn hóa đầu ra công cụ

Để tối ưu hóa khả năng xử lý của Agent, các công cụ tích hợp trong hệ thống được thiết kế lại để trả về kết quả dưới dạng dữ liệu có cấu trúc (JSON) thay vì văn bản thô.

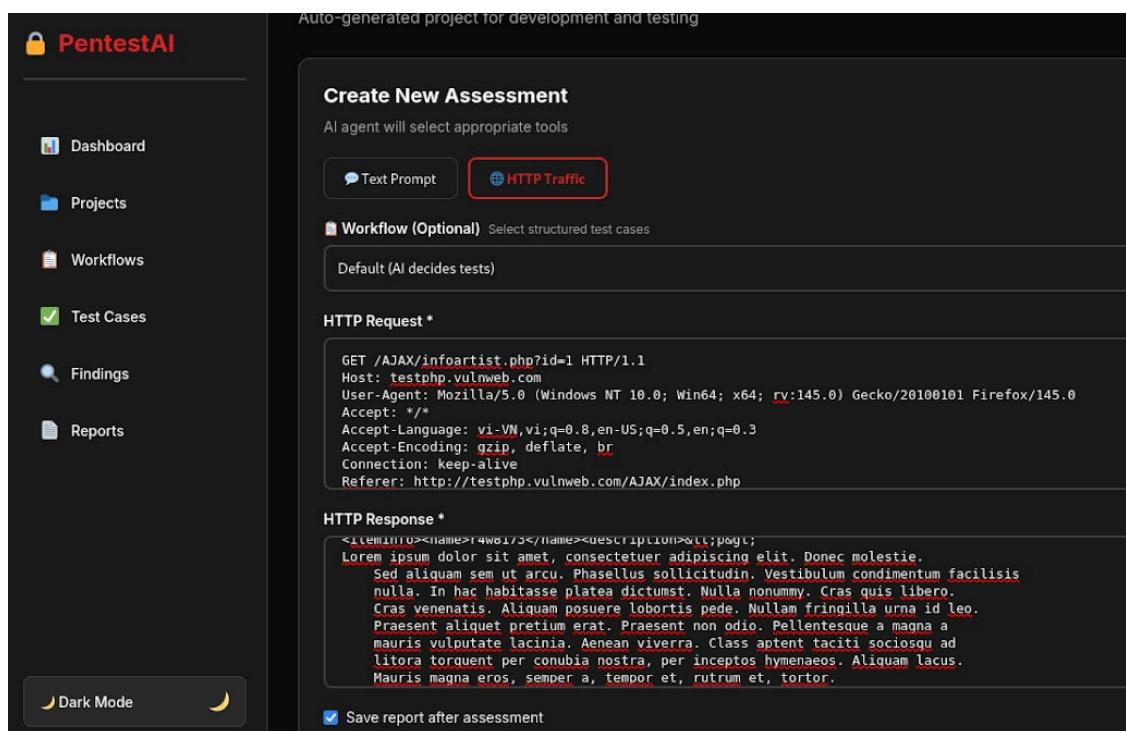
- Tối ưu hóa ngữ cảnh: Việc trả về dữ liệu có cấu trúc giúp LLM dễ dàng trích xuất các thông tin quan trọng (như danh sách URL, tham số lỗ hổng) mà không cần tốn nhiều tài nguyên token để phân tích (parse) các định dạng văn bản phức tạp từ terminal.

- Hỗ trợ chuỗi suy luận: Kết quả đầu ra từ công cụ này (ví dụ: danh sách tham số) có thể được nạp trực tiếp làm đầu vào cho công cụ tiếp theo (ví dụ: XSS scanner) một cách liền mạch, tạo nên chuỗi tấn công tự động hóa hoàn chỉnh.

### 3.7. Phát triển giao diện web

Để hiện thực hóa kiến trúc của hệ thống multi-agent và mang lại trải nghiệm trực quan cho người dùng, giao diện của AiPT được xây dựng dưới dạng một web SPA sử dụng thư viện ReactJS và Flask. Giao diện này không chỉ là nơi hiển thị kết quả mà còn đóng vai trò là trạm điều khiển trung tâm, cho phép pentester tương tác sâu với quy trình vận hành của các agent. Thiết kế giao diện tập trung giải quyết ba bài toán cốt lõi: cấu hình linh hoạt, giám sát minh bạch và báo cáo trực quan.

#### a) Cấu hình cuộc kiểm thử



Hình 3.5: Giao diện khởi tạo cuộc kiểm thử mới với đầu vào là HTTP traffic

Hình 3.5 mô tả điểm khởi đầu của một quy trình kiểm thử. Giao diện được thiết kế để tối ưu hóa thao tác nhập liệu, thay thế các dòng lệnh CLI phức tạp bằng các biểu mẫu trực quan.

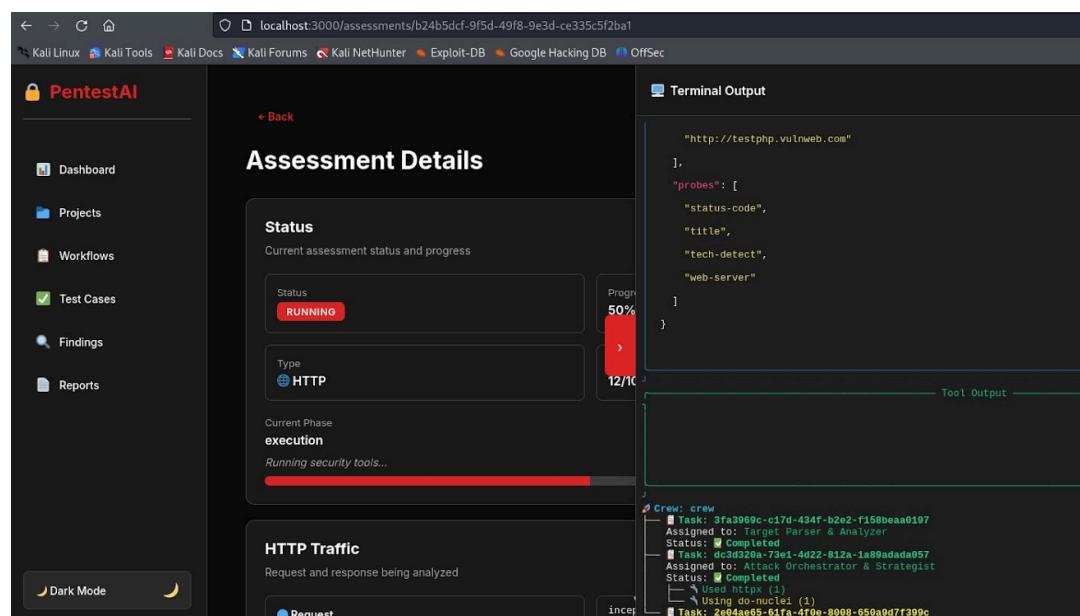
- **Đa dạng hóa đầu vào:** hệ thống hỗ trợ linh hoạt hai phương thức khởi tạo
  - Tự động hóa từ Recon Agents: có thể bắt đầu ngay với các API đã được thu thập, chỉ cần chỉ định workflow cụ thể muốn thực hiện.
  - Thủ công: người dùng dán trực tiếp gói tin HTTP thô (thu thập từ Burp Suite hoặc DevTools). Hệ thống sẽ tự động phân tích cú pháp (parse) gói tin này để trích xuất các thông tin kỹ thuật.
- **Phân tích ngữ nghĩa tiền xử lý:** Trước khi kích hoạt AI Agent, hệ thống hiển thị bản xem trước của các trường thông tin quan trọng đã được phân tích. Bước này giúp người dùng rà soát độ chính xác của dữ liệu đầu vào, đảm bảo AI nhận được ngữ cảnh đúng đắn.
- **Lựa chọn chiến lược:** Người dùng có quyền chỉ định chiến lược kiểm thử cụ thể (ví dụ: Profile chuyên biệt cho SQL Injection) hoặc để chế độ "Default" cho AI tự quyết định.

#### b) Giám sát trạng thái

Một trong những thách thức lớn nhất của việc ứng dụng AI là khả năng giám sát các hoạt động của AI. Framework AiPT giải quyết bài toán giám sát này thông qua cơ chế giám sát đa tầng tại màn hình "Assessment Details" (xem hình 9):

- **Tầng logic (Task Crew Logs):** Hiển thị danh sách các tác vụ (Tasks) đang được phân phối cho từng Agent cụ thể (ví dụ: Agent nào đang phân tích mục tiêu, Agent nào đang thực thi tấn công) cùng trạng thái xử lý thời gian thực.

- Tầng kỹ thuật (Terminal Output): Cung cấp cửa sổ giả lập terminal, hiển thị luồng dữ liệu thô (Raw Logs/JSON) trả về từ các công cụ nền (như httpx, nuclei). Tính năng này cho phép các pentester thực hiện việc "kiểm tra chéo" và debug hành vi của hệ thống ngay lập tức, đảm bảo công cụ đang chạy đúng như mong đợi.



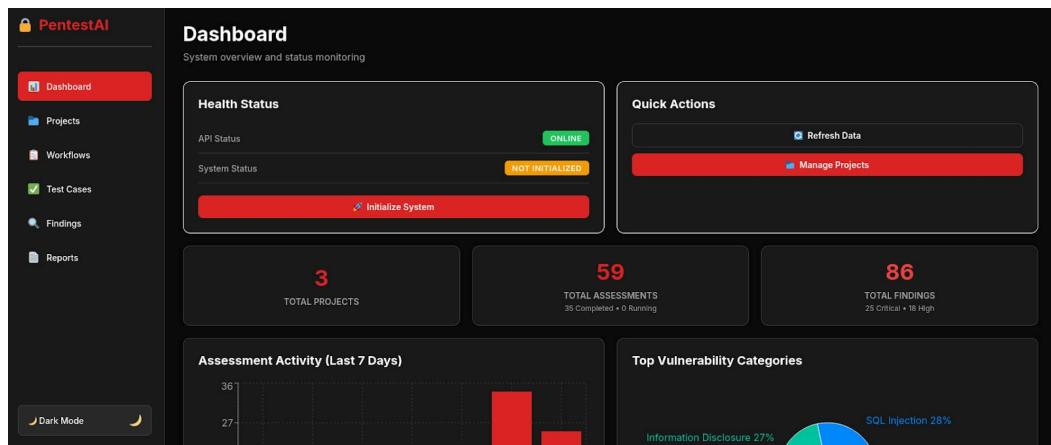
Hình 3.6: Bảng giám sát trạng thái hoạt động của các agent và terminal output

### c) Quản trị và báo cáo

Hình 3.7 cho thấy màn hình quản lý kết quả tập trung vào việc chuyển đổi dữ liệu kỹ thuật khô khan thành thông tin mang tính thực tế:

- Phân cấp mức độ nghiêm trọng: Các lỗ hổng phát hiện được tự động gắn nhãn màu sắc trực quan dựa trên hệ thống điểm CVSS 3.1: High (Đỏ), Medium (Cam), Low/Info (Xanh).
- Bảng chứng khai thác (Proof of Concept - PoC): Mỗi mục lỗ hổng không chỉ chứa tiêu đề và mô tả, mà còn cung cấp khả năng tương tác mở rộng. Người

dùng có thể xem chi tiết gói tin HTTP (Request/Response) đã gây ra lỗ hổng và các bước tái hiện cụ thể do Agent thực hiện.



Hình 3.7: Danh sách các lỗ hổng bảo mật được phát hiện và phân loại theo mức độ nghiêm trọng

Tổng thể, việc phân tách rõ ràng ba chức năng (cấu hình → giám sát → phân tích) tạo nên một quy trình làm việc khép kín và chuyên nghiệp với AiPT.

### 3.8. Tổng kết chương

Chương 3 đã trình bày chi tiết về thiết kế và quy trình triển khai của framework AiPT. Dựa trên cơ sở lý thuyết đã phân tích ở Chương 2, hệ thống được xây dựng với các đặc điểm kỹ thuật nổi bật sau:

1. Kiến trúc hướng Agent: Hệ thống chuyển dịch từ mô hình rà quét tuyến tính truyền thống sang mô hình mạng lưới cộng tác giữa Recon Agents và Exploit Agents, được điều phối bởi framework ADK và CrewAI sử dụng sức mạnh từ mô hình LLM Gemini-2.5-Flash.
2. Tích hợp công cụ qua MCP: Giải quyết triệt để "ảo giác" của LLM bằng cách sử dụng giao thức Model Context Protocol để tiêu chuẩn hóa việc

gọi các công cụ kiểm thử (Burp Suite, Nuclei,...), đảm bảo tính chính xác và an toàn khi thực thi.

3. Quy trình Human-in-the-loop: Thiết kế hệ thống đều đặt con người vào vị trí trung tâm của việc ra quyết định, đảm bảo AI đóng vai trò là trợ lý đắc lực thay vì hoạt động mất kiểm soát.

Kiến trúc này không chỉ đáp ứng được yêu cầu về tự động hóa các tác vụ lặp lại mà còn mở ra khả năng xử lý các kịch bản kiểm thử phức tạp đòi hỏi tư duy logic. Chương tiếp theo sẽ đi sâu vào việc thực nghiệm hệ thống trên các môi trường lab tiêu chuẩn để đánh giá hiệu quả thực tế của các giải pháp đã đề xuất.

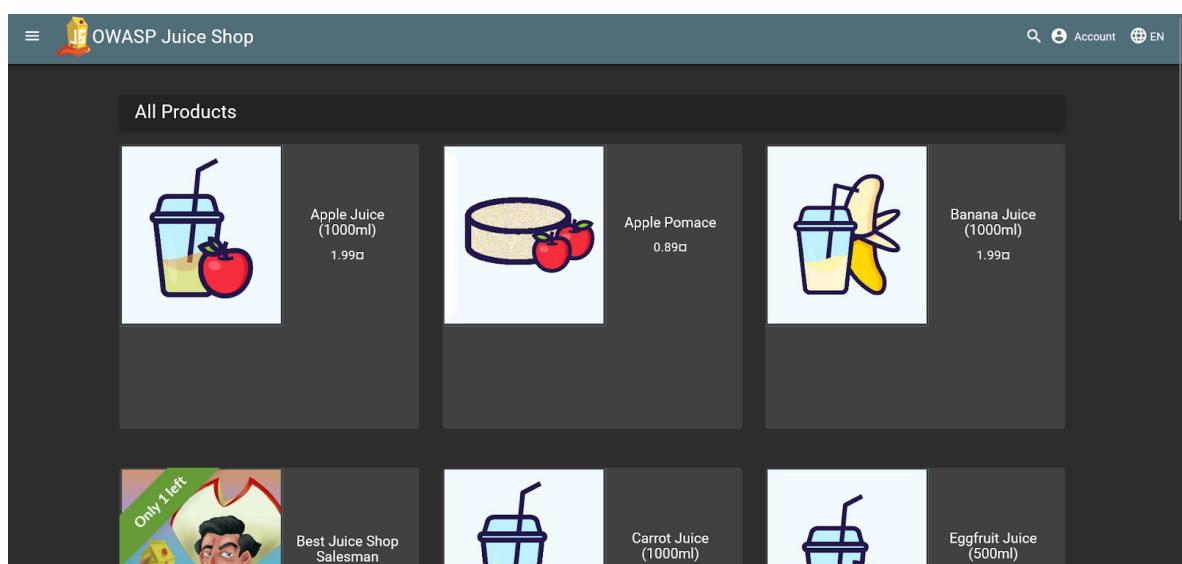
## CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

### 4.1. Môi trường và kịch bản thực nghiệm

#### a) Môi trường lab - OWASP Juice Shop

Môi trường thực nghiệm được thiết lập dựa trên OWASP Juice Shop, một ứng dụng web mẫu do tổ chức OWASP phát triển nhằm mô phỏng toàn bộ các lỗ hổng bảo mật thuộc danh sách OWASP Top 10. Được xây dựng trên nền tảng công nghệ Node.js, Express và Angular, hệ thống này là đại diện tiêu biểu cho các ứng dụng Single Page Application hiện đại.

Để đảm bảo tính cách ly và độ ổn định cho quá trình thử nghiệm, ứng dụng được triển khai cục bộ dưới dạng Docker Container (hình 11). Mục tiêu chính của quá trình kiểm thử trên môi trường này là tập trung khai thác và phân tích các lỗ hổng liên quan đến Broken Authentication và Injections trong các quy trình nghiệp vụ như đăng nhập và quản lý giỏ hàng.

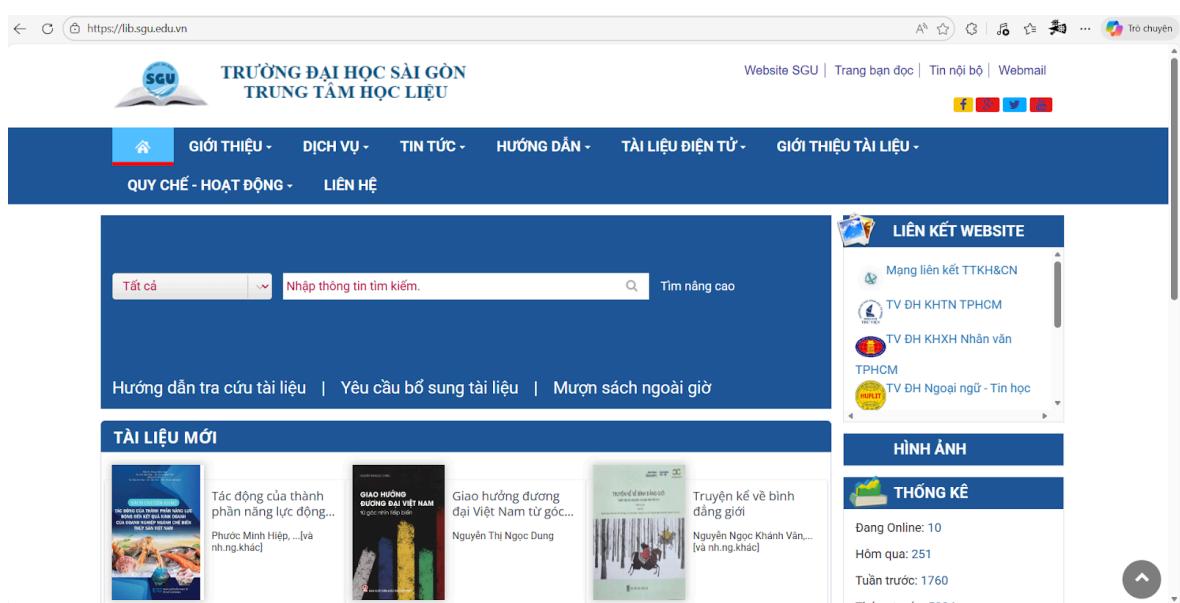


Hình 4.1: Ứng dụng OWASP Juice Shop

#### b) Môi trường thực tế - Website Thư viện SGU

Đối với thử nghiệm trong môi trường thực tế, nghiên cứu lựa chọn website Thư viện trường Đại học Sài Gòn (tại địa chỉ: <https://lib.sgu.edu.vn>) làm đối tượng khảo sát\*. Đây là một ứng dụng web thực tiễn được xây dựng và vận hành trên nền tảng công nghệ Microsoft IIS kết hợp với ASP.NET. Quá trình kiểm thử được thực hiện trong phạm vi giới hạn theo phương pháp Non-destructive testing, tác động chủ yếu lên các tham số truy vấn công khai.

Mục tiêu của hoạt động này nhằm đánh giá khả năng phát hiện các lỗ hổng bảo mật như SQL Injection và Cross-Site Scripting (XSS) trong điều kiện môi trường vận hành thực tế đã được trang bị các cơ chế bảo vệ cơ bản như WAF hoặc bộ lọc dữ liệu.



Hình 4.2: Website Thư viện SGU

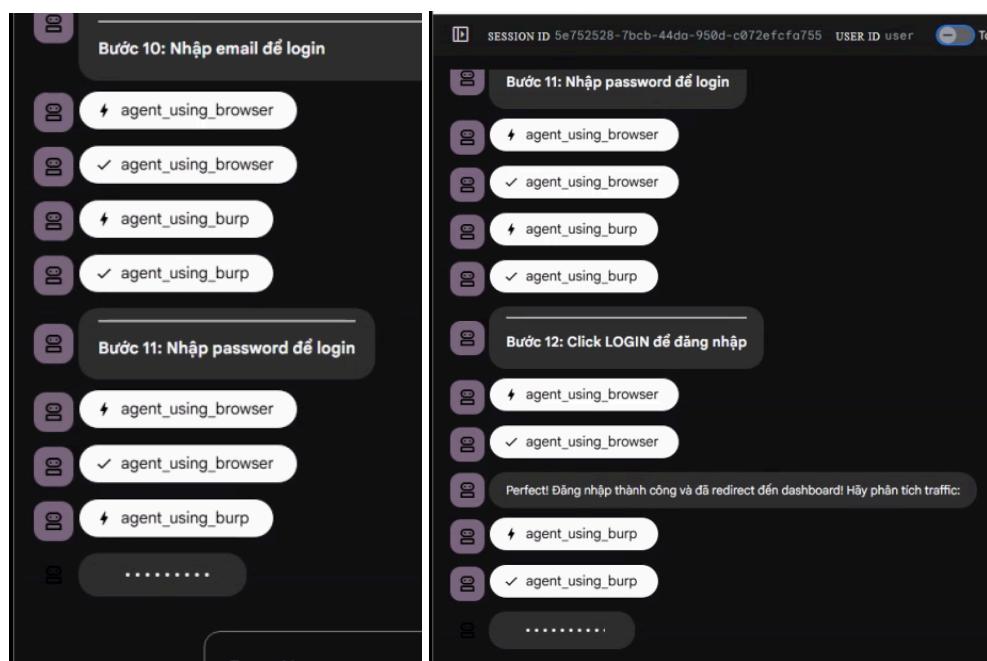
\* *Kiểm thử tại kịch bản thứ hai chỉ vì mục đích học thuật, không thực hiện khai thác hoặc gây ảnh hưởng về an toàn thông tin và toàn vẹn hệ thống.*

## 4.2. Tiến hành kiểm thử

*a) Kịch bản 1: Khai thác lỗ hổng xác thực trên OWASP Juice Shop*

Trong kịch bản này, nhóm tập trung kiểm thử chức năng "Đăng nhập" của Juice Shop. Đây là chức năng quan trọng nhất của mọi ứng dụng web và thường là điểm khởi đầu cho các cuộc tấn công chiếm quyền kiểm soát.

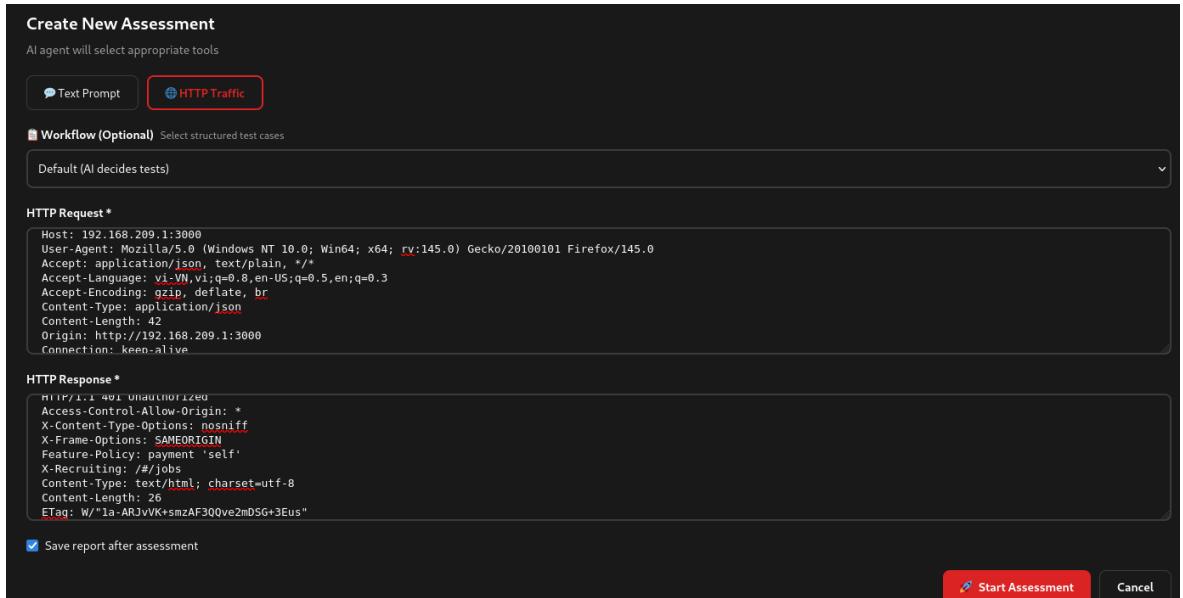
### Bước 1: Khởi tạo và phân tích đầu vào



Hình 4.3: Recon Agents hoạt động để dò tìm các API của web-app

- Tiến hành quá trình Reconnaissance: người dùng truy cập vào màn hình chat của Recon Agents để tiến hành gửi thông tin về web app mục tiêu muốn kiểm thử. Tại đây, agent sẽ nhận lệnh và tiến hành rà soát mục tiêu, tìm kiếm các API và sau đó gửi thông tin đến backend. Thông tin được gửi bao gồm HTTP request và response

tiêu chuẩn của API kèm mô tả về chức năng và logic nghiệp vụ có liên quan.



*Hình 4.4: Màn hình khởi tạo Assessment với API request do Recon-agents gửi*

- Thiết lập workflow: Người dùng có thể gán các workflow có sẵn trên AiPT cho mỗi API cụ thể hoặc tự tạo các testcase và workflow tùy chỉnh cho kịch bản muôn kiểm thử. Mặc định thì workflow sẽ được thiết lập ở chế độ "Default" - trao quyền cho Agent tự quyết định chiến lược tấn công dựa trên ngữ cảnh phân tích được.

## Bước 2: Tiến hành kiểm thử theo workflow

Sau khi nhận nhiệm vụ, Agent bắt đầu quy trình suy luận. Dựa trên nhật ký hoạt động và đầu ra của terminal, ghi nhận Agent đã xây dựng một kế hoạch tấn công gồm nhiều giai đoạn mạch lạc:

The screenshot displays the PentesAI interface. On the left is a dark sidebar with navigation links: Dashboard, Projects, Workflows, Test Cases (highlighted), Findings, and Reports. A 'Dark Mode' toggle is at the bottom.

**Assessment Details:**

- Status:** RUNNING (Progress: 50%)
- Type:** HTTP
- Created:** 12/10/2025, 2:26:06 PM
- ID:** 90da965e-83bc-4bb9-b144-8621cfbd3cf1
- Current Phase:** execution (Running security tools...)

**HTTP Traffic:** Request and response being analyzed. It shows a Request tab and a Response tab.

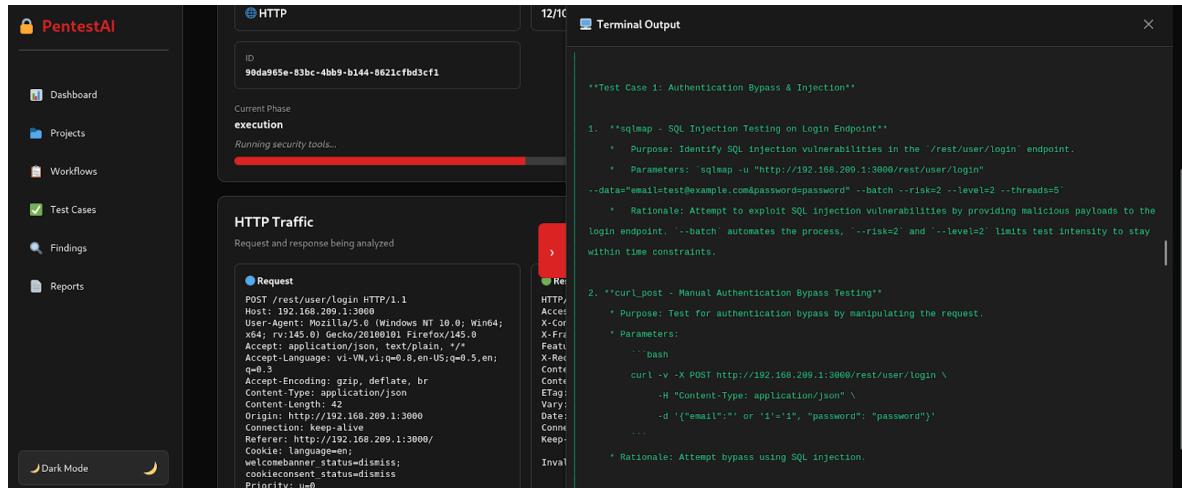
**Activity Timeline:** Real-time assessment progress. It lists several events with icons and status markers (green checkmarks, red error icons).

**HTTP Request Detail:** Shows a detailed view of an HTTP POST request to /rest/user/login. Headers include Host, User-Agent, Accept, Accept-Language, Content-Type, Content-Length, Origin, Referer, Cookie, and Priority. The body contains JSON data: {"username": "admin", "password": "password", "rememberMe": true}.

**Terminal Output:** A terminal window titled '12/10/2025 - Terminal Output'. It shows a completed task (Task ID: 43731476-dcfe-4082-a37a-d9cfcfb0ab52e) assigned to 'Attack Orchestrator & Strategist'. The 'Final Answer' section contains a penetration testing plan and a phase 1 reconnaissance section. The output also includes configuration details for nmap and curl commands.

Hình 4.5: Chiến lược được Agent đề xuất và thực thi

- Hình 4.5 cho thấy Agent nhận diện được endpoint “/rest/user/login” có khả năng bị tấn công SQL Injection → quyết định sử dụng sqlmap để kiểm tra các tham số trong JSON body.
- Agent có khả năng tự động cấu hình các chế độ tối ưu cho sqlmap tùy theo tình huống như --batch (chạy tự động không hỏi), --risk=2, --level=2 (tăng độ sâu kiểm thử) và --threads=5 (tăng tốc độ).

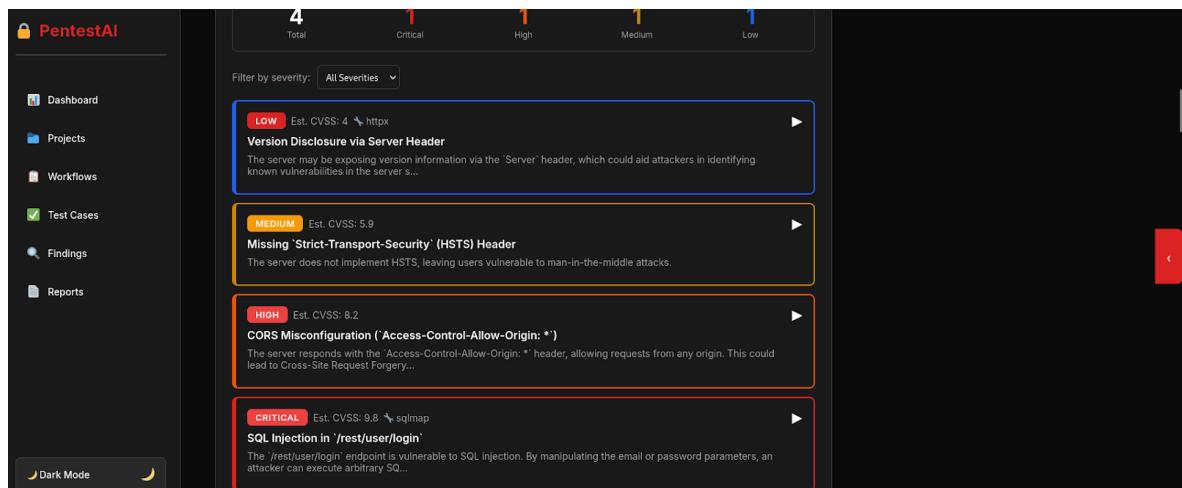


Hình 4.6: Quan sát logs trong quá trình Exploit Agents thực thi(en) nhiệm vụ

- Quan sát hình 4.6, ta thấy Agent biết cách sử dụng curl để thực hiện các kỹ thuật Authentication Bypass đơn giản (ví dụ: SQLi payload ' OR '1'='1) nhằm xác minh nhanh trước khi chạy quét sâu.
- Khả năng tự lập kế hoạch này minh chứng cho ưu điểm vượt trội của AiPT so với các scanner truyền thống: nó không chỉ chạy công cụ một cách mù quáng mà có tư duy chiến lược - biết "nhìn trước ngó sau".

### Bước 3: Tổng hợp kết quả và báo cáo

Sau khi các công cụ hoàn tất, một sub-agent là Parser Agent thực hiện thu thập toàn bộ dữ liệu đầu ra, loại bỏ nhiễu và tổng hợp thành các findings cụ thể. Kết quả cuối cùng trên Dashboard cho thấy framework AiPT đã phát hiện thành công nhiều vấn đề bảo mật nghiêm trọng và có giá trị.



Hình 4.7: Màn hình thể hiện các kết quả thu được từ quá trình kiểm thử của Exploit-Agents

### 1. SQL Injection in /rest/user/login:

- Đây là lỗ hổng nghiêm trọng nhất. AiPT đã xác định chính xác rằng tham số email trong JSON Body dễ bị tồn thương với các dạng tấn công Time-based blind và Boolean-based blind.
- Framework cung cấp bằng chứng cụ thể là lệnh curl mô phỏng cuộc tấn công thành công, cho phép kẻ tấn công đăng nhập mà không cần mật khẩu hoặc trích xuất dữ liệu từ database SQLite của Juice Shop.

### 2. CORS Misconfiguration:

- Hệ thống phát hiện header “Access-Control-Allow-Origin: \*”, cho phép bất kỳ trang web nào cũng có thể gửi yêu cầu đến API này, tiềm ẩn nguy cơ CSRF hoặc rò rỉ dữ liệu người dùng.

### 3. Missing Security Headers:

- Thiếu header HSTS (Strict-Transport-Security) và lộ thông tin phiên bản máy chủ (Version Disclosure).

The screenshot displays two panels of the PentesAI application. The left panel is a sidebar with navigation links: Dashboard, Projects, Workflows, Test Cases (highlighted with a green checkmark), Findings, and Reports. It also includes a 'Dark Mode' toggle. The right panel is divided into sections:

- High Severity:**
  - CORS Misconfiguration (Access-Control-Allow-Origin: \*)**
  - Description:** The server responds with the Access-Control-Allow-Origin: \* header, allowing requests from any origin. This could lead to Cross-Site Request Forgery (CSRF) attacks or unauthorized data access.
  - Evidence:** curl -v -H "Origin: http://evil.com" http://192.168.209.1:3000/
 

```
curl -v -H "Origin: http://evil.com" http://192.168.209.1:3000/
```
  - Impact:** An attacker can make requests to the API on behalf of a user, potentially performing actions without the user's knowledge or consent. Sensitive data could be exposed to malicious websites.
  - Est. CVSS:** 8.2
- Medium Severity:**
  - Missing Strict-Transport-Security (HSTS) Header**
  - Description:** The server does not implement HSTS, leaving users vulnerable to man-in-the-middle attacks.
  - Evidence:** The curl -v -I http://192.168.209.1:3000/ command does not show the Strict-Transport-Security header in the response.
  - Impact:** Attackers can intercept communication between the user and the server, potentially stealing sensitive information.
  - Est. CVSS:** 5.9
- Low / Unimportant:**

The second panel is titled **Recommendations** and contains two sections:

- Immediate Actions (Priority 1):**
  1. Remediate the SQL Injection Vulnerability in /rest/user/login
 

```
// Example (Java/JDBC - parameterized query)
String sql = "SELECT * FROM users WHERE email = ? AND password = ?";
PreparedStatement pstmt = connection.prepareStatement(sql);
pstmt.setString(1, email);
pstmt.setString(2, password);
ResultSet rs = pstmt.executeQuery();
```
  - Why:** SQL injection poses a critical risk, allowing attackers to bypass authentication and potentially compromise the entire system.
  - How:** Implement parameterized queries or prepared statements to prevent SQL injection. Sanitize user inputs and validate data before using it in SQL queries. Use an ORM framework to abstract database interactions and further mitigate SQL injection risks.
- Short-term Actions (Priority 2):**
  1. Implement Proper CORS Configuration
 

```
# Example Nginx configuration
location / {
    add_header 'Access-Control-Allow-Origin' 'https://yourdomain.com';
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
    add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Content-Type,Content-Length,Content-Range';
```

Hình 4.8: Chi tiết mỗi findings của agent

Điểm đặc biệt của AiPT là khả năng đưa ra khuyến nghị sửa lỗi phù hợp ngữ cảnh của ứng dụng. Như hình 4.8, đối với lỗi SQL Injection, thay vì chỉ đưa ra lời khuyên chung chung, AI cung cấp một đoạn mã mẫu sử dụng Prepared Statements (cơ chế tham số hóa câu lệnh truy vấn) trong Java/JDBC, giúp lập trình viên dễ dàng hình dung cách vá lỗi. Kết quả thực nghiệm trên môi trường lab (OWASP Juice Shop) đã chứng minh AiPT hoạt động hoàn toàn đúng theo thiết kế: Nhận diện mục tiêu → Lập kế hoạch trinh sát/tấn công → Điều phối công cụ PT → Tổng hợp báo cáo chính xác.

*Bảng 3. Thống kê số lượng lỗ hổng phát hiện trong môi trường lab*

<b>Loại lỗ hổng</b>	<b>Tổng số lượng</b>	<b>Tỷ lệ phát hiện</b>
Sensitive Data Exposure	15	66,67%
Improper Input Validation	12	58,33%
Broken Access Control	11	45,45%
Injections	11	72,72%
Vulnerable Components	9	66,67%
XSS	9	66,67%
Broken Authentication	9	88,89%
Cryptographic Issues	5	20,00%
Observability Failures	4	N/A
Security Misconfiguration	4	100%
Broken Anti-Automation	4	N/A
Miscellaneous	7	42,86%
Security through Obscurity	3	33,33%
Insecure Deserialization	3	33,33%
Unvalidated Redirects	2	100%
XXE	2	100%

Kết quả thống kê từ bảng 3 cho thấy framework AiPT đã đạt được hiệu quả khá quan trọng trong việc nhận diện và khai thác các lỗ hổng bảo mật web. Cụ thể:

- Thứ nhất, khả năng phát hiện các lỗ hổng nghiêm trọng (Critical/High) thuộc danh sách OWASP Top 10 đạt tỷ lệ rất khả quan. Điển hình là nhóm lỗ hổng Broken Authentication đạt tỷ lệ phát hiện lên tới 88,89% và nhóm Injections

(như SQL Injection) đạt 72,72%. Đây là một kết quả đáng ghi nhận. Điều này chứng minh cơ chế tích hợp công cụ thông qua giao thức MCP hoạt động ổn định, tận dụng tốt sức mạnh của các scanner nền tảng để không bỏ sót các lỗi technical.

- Thứ hai, độ chính xác tuyệt đối được ghi nhận ở các nhóm lỗ hổng liên quan đến cấu hình và kỹ thuật cụ thể. Các nhóm Security Misconfiguration, Unvalidated Redirects và XXE đều đạt tỷ lệ phát hiện 100%. Kết quả thể hiện khả năng phân tích và nhận biết về bảo mật của agent rất chính xác.
- Thứ ba, về mặt số lượng, nhóm Sensitive Data Exposure chiếm tỷ lệ 66,67%. Kết quả này cho thấy khả năng của module Recon Agents và Attack-surface Finder trong việc rà soát và phân tích các phản hồi HTTP để tìm kiếm thông tin rò rỉ là rất hiệu quả.

Tuy nhiên, bảng số liệu cũng chỉ ra hạn chế của hệ thống đối với nhóm lỗ hổng Cryptographic Issues, chỉ đạt tỷ lệ phát hiện 20,00%. Nguyên nhân do việc kiểm thử từ bên ngoài gặp khó khăn trong việc đánh giá chính xác chất lượng thuật toán mã hóa được cài đặt sâu trong mã nguồn backend, đòi hỏi cần có các phương pháp tiếp cận chuyên sâu hơn hoặc kết hợp với phân tích mã nguồn. Các dạng lỗ hổng đặc biệt như Broken Anti-Automation hay Observability Failures đòi hỏi tư duy rất cao và AI agent gặp khó khăn khi phân tích các vấn đề này. Các nhóm lỗ hổng khó khai thác và không có công cụ scanner như Security through Obscurity hay Insecure Deserialization thường đòi hỏi khả năng phân tích và custom payload theo ngữ cảnh. Điều này AI agent cũng chưa thể làm được. Nhìn chung AI agent chưa thể hoàn toàn đáp ứng được toàn bộ yêu cầu như một pentester con người thực thụ. Đây là hạn chế mà sẽ cần thêm nhiều thời gian để cải tiến trong tương lai.

b) Kịch bản 2: kiểm thử trên môi trường thực tế (Website thư viện SGU)

Khác với môi trường lab vốn được thiết kế để "dễ bị tấn công", môi trường thực tế thường được trang bị các cơ chế bảo vệ như Web Application Firewall (WAF) và cấu hình máy chủ cứng hơn. Kịch bản này nhằm đánh giá khả năng thích ứng và độ chính xác của AiPT khi đối mặt với một ứng dụng .NET chạy trên nền tảng Microsoft IIS.

### **Bước 1: Khởi tạo và xác định bệ mặt tấn công**

Mục tiêu được lựa chọn là trang chi tiết tin tức của thư viện trường Đại học Sài Gòn, thông tin cụ thể như sau:

- URL: <https://lib.sgu.edu.vn/trangtingioithieuchitiet.aspx?id=32>
- Đặc điểm kỹ thuật: URL chứa tham số id=32, một dấu hiệu kinh điển cho việc truy vấn cơ sở dữ liệu dựa trên khóa chính. Đuôi mở rộng .aspx cho thấy công nghệ phía server là ASP.NET.

Người dùng cung cấp URL này vào AiPT. Hệ thống tự động phân tích và xác định tham số id là “injection point” trọng yếu (xem hình 19).

The screenshot displays the PentesAI web application interface. On the left, there is a sidebar with navigation links: Dashboard, Projects, Workflows, Test Cases (selected), Findings, and Reports. Below the sidebar are 'Dark Mode' and 'Light Mode' buttons.

**Assessment Details:**

- Status:** COMPLETED (green button)
- Progress:** 100%
- Type:** HTTP
- Created:** 12/10/2025, 4:51:46 PM
- ID:** 4354bba7-6481-4577-9704-26c6f9fa8cbe

**Activity Timeline:**

- Assessment completed successfully!
- Agent completed: Reporter Agent
- Phase: reporting - Processing
- Agent started: Reporter Agent
- Phase: reporting - Generating comprehensive report...
- Agent completed: Executor Agent
- Phase: execution - Tools execution completed

**HTTP Traffic:**

Request and response being analyzed.

**Request:**

```
GET /trangtingioithieuchiet.aspx?id=32
HTTP/2.0.1
Host: lib.sgu.edu.vn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:145.0) Gecko/20100101 Firefox/145.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.4
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u0, i
Te: trailers
Connection: keep-alive
```

**Response:**

```
HTTP/2.00 OK
Content-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/10.0
Set-Cookie: ASP.NET_SessionId=uypspkrssamerrplamqvh; path=/; HttpOnly; SameSite=Lax
X-Firefox-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Content-Security-Policy: default-src 'self';
script-src 'self' 'unsafe-inline' 'unsafe-eval'
https://connect.facebook.net; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com
https://cdn.jsdelivr.net
https://www.cloudflare.com; font-src 'self'
data: https://fonts.gstatic.com
https://cdn.jsdelivr.net
https://cdnjs.cloudflare.com; img-src 'self'
data: https://cdn.jsdelivr.net https://*.vn;
```

Hình 4.9: Khởi tạo bài kiểm thử cho kịch bản thứ hai

## Bước 2: Thực thi kiểm thử

- Phát hiện SQL Injection: Agent nhận thấy tham số id có phản hồi thay đổi khi chèn các ký tự đặc biệt. Nó quyết định kích hoạt sqlmap với các tham số tối ưu cho môi trường thực tế.

- Câu lệnh thực thi: sqlmap -u "..." --batch --risk=2 ...
- Kết quả: Phát hiện lỗ hổng SQL Injection dạng Boolean-based blind và Time-based blind.

- Nhận xét: Đáng chú ý, Agent xác định chính xác DBMS backend là Microsoft SQL Server 2022. Điều này chứng tỏ khả năng "fingerprinting" (nhận diện công nghệ) rất tốt của hệ thống.
2. Phân tích XSS: Agent thử nghiệm chèn payload <script>alert('XSS')</script> vào tham số id. Server trả về lỗi 500 (Internal Server Error).
- Thay vì khẳng định ngay về kết quả (như nhiều scanner DAST cũ thường làm khi thấy lỗi 500), AiPT đánh dấu đây là "Potential Reflected Cross-Site Scripting" và ghi chú rõ ràng: "This might indicate the payload was processed... Requires further investigation" (Điều này có thể cho thấy payload đã được xử lý... Cần điều tra thêm).
  - Cách xử lý này cho thấy sự thận trọng và tư duy logic của AI, giúp giảm tỷ lệ dương tính giả, không báo động sai gây hoang mang cho người quản trị.
3. Lộ thông tin máy chủ: Hệ thống cũng báo cáo về việc lộ phiên bản máy chủ (Microsoft-IIS/10.0) qua header Server và ghi nhận các Security Headers đang hiện hữu.

The screenshot displays two views of the PentesAI platform interface:

- Security Findings Tab:**
  - Dashboard:** Shows a summary of findings: Total 4, Critical 1, Medium 1, Low 1, Info 1.
  - Findings List:** A list of vulnerabilities:
    - INFO**: curl - Security Headers Present: The server implements several security headers.
    - LOW**: Est. CVSS: 4.0 https - Information Disclosure - Web Server Version: The 'Server: Microsoft-IIS/10.0' header exposes the web server version. While not directly exploitable, this information can be used by attackers to...
    - MEDIUM**: Est. CVSS: 6.1 curl - Potential Reflected Cross-Site Scripting (XSS): Attempting to inject an XSS payload into the 'id' parameter resulted in a 500 Internal Server Error. This \*might\* indicate the payload was processed b...
    - CRITICAL**: Est. CVSS: 9.8 sqlmap - SQL Injection Vulnerability: The 'id' parameter in '/trantinggioithieuchitiet.aspx' is vulnerable to SQL injection. This allows an attacker to execute arbitrary SQL queries on the...
- Technical Findings Tab:**
  - Discovered Services:**

Port	Service	Version	State
80	http	Microsoft IIS/10.0	open
443	https	Microsoft IIS/10.0	open
  - Identified Vulnerabilities:**
    - Critical Severity:**
      - SQL Injection Vulnerability:**
        - Description:** The 'id' parameter in '/trantinggioithieuchitiet.aspx' is vulnerable to SQL injection. This allows an attacker to execute arbitrary SQL queries on the backend database.
        - Evidence:** SQLMap successfully identified and exploited boolean-based blind and time-based blind SQL injection. The backend DBMS is Microsoft SQL Server 2022.
        - Impact:** An attacker can potentially read sensitive data from the database, modify data, or even execute arbitrary commands on the server.
        - Est. CVSS:** 9.8 (Critical)

```
sqlmap -u "http://lib.sgu.edu.vn/trantinggioithieuchitiet.aspx?id=32" --batch --risk=2 --level=2
```
    - Medium Severity:**
      - Potential Reflected Cross-Site Scripting (XSS):**
        - Description:** Attempting to inject an XSS payload into the 'id' parameter resulted in a 500 Internal Server Error. This \*might\* indicate the payload was processed by the server before failing, suggesting a potential XSS vulnerability. Requires further investigation.
        - Evidence:** Injecting <script>alert('XSS')</script> into the 'id' parameter via curl resulted in a 500 error.

Hình 4.10: Các findings của agent trên mục tiêu trong kịch bản thứ hai

Công cụ đã xác minh thành công lỗ hổng mà không cần thực hiện các hành động phá hoại (như drop table), đảm bảo tính an toàn cho website.

### 4.3. Đánh giá hiệu quả

a) Vẽ khả năng phát hiện lỗ hổng

AiPT đã chứng minh khả năng phát hiện chính xác các lỗ hổng nghiêm trọng thuộc nhóm OWASP Top 10:

- SQL Injection: Phát hiện thành công cả trên môi trường lab (SQLite) và thực tế (MSSQL). Khả năng tự động điều chỉnh payload phù hợp với từng loại Database là điểm vượt trội so với các script thủ công.
- Injection trong JSON: Kịch bản Juice Shop cho thấy AiPT xử lý tốt các API RESTful, phân tích sâu vào cấu trúc JSON để tìm lỗi - một điểm yếu mà các crawler truyền thống thường bỏ sót.
- Độ chính xác: Hệ thống thể hiện khả năng phân biệt tốt giữa "Lỗ hổng xác định" và "Cảnh báo tiềm năng", như trong trường hợp XSS tại kịch bản 2, qua đó hỗ trợ Pentester tập trung vào các vấn đề thực sự.

*b) Về khả năng tự động hóa và điều phối*

Framework đã giải quyết tốt bài toán "Tool Orchestration" đặt ra ở Chương 2:

- Không cần can thiệp thủ công: Từ lúc nhập Request thô đến khi có báo cáo, người dùng không cần gõ bất kỳ lệnh nào. Agent tự động chuyển dữ liệu giữa các bước (ví dụ: lấy URL từ bước Recon để nạp vào bước Exploit).
- Giảm tải: Thay vì phải nhớ hàng tá tham số của sqlmap hay nmap, Pentester chỉ cần giám sát quá trình. Thời gian hoàn thành một bài kiểm thử cơ bản giảm xuống chỉ còn vài phút so với hàng giờ làm việc thủ công.

*c) Hạn chế tồn đọng*

Bên cạnh các ưu điểm, thực nghiệm cũng bộc lộ một số hạn chế nhất định:

- Thời gian phản hồi: Do phụ thuộc vào LLM (Gemini), đôi khi có độ trễ trong quá trình Agent "suy nghĩ" (Reasoning phase), đặc biệt khi phân tích các HTTP Response quá dài.
- Phụ thuộc vào công cụ nền: Khả năng tấn công của AiPT bị giới hạn bởi khả năng của các công cụ được tích hợp (sqlmap, nuclei). Nếu công cụ nền không hỗ trợ một kỹ thuật mới, Agent cũng không thể thực hiện được trừ khi được lập trình thêm module code python tùy chỉnh.

#### 4.4. Tổng kết chương

Chương 4 đã hoàn tất kiểm chứng thực nghiệm framework AiPT trên cả môi trường phòng thí nghiệm (OWASP Juice Shop) và thực tế ( website thư viện SGU), qua đó khẳng định tính khả thi của mô hình đề xuất. Kết quả thực nghiệm cho thấy ba ưu điểm cốt lõi: (1) Khả năng phát hiện lỗ hổng khá tốt nhờ phân tích ngữ nghĩa và nhận diện công nghệ chính xác cùng cơ chế tích hợp công cụ PT; (2) Kiến trúc Agent/MCP hoạt động ổn định, khắc phục sự rời rạc của công cụ truyền thống thông qua khả năng tự động lập kế hoạch và suy luận; (3) Giảm thiểu tỷ lệ dương tính giả nhờ cơ chế phân loại rủi ro thận trọng.

## PHẦN BA - KẾT LUẬN

### 1. Kết quả đạt được

Đề tài nghiên cứu "Xây dựng Framework tự động hóa Penetration Testing dựa trên AI Agent" đã hoàn thành các mục tiêu đề ra ban đầu, giải quyết thành công bài toán kết hợp sức mạnh suy luận của AI với độ tin cậy của các công cụ kiểm thử truyền thống. Các kết quả cụ thể đạt được bao gồm:

- Xây dựng thành công kiến trúc hệ thống Multi-Agent: nghiên cứu đã thiết kế và triển khai một hệ thống phối hợp đa tác nhân bao gồm Recon Agents và Exploit Agents. Kiến trúc này cho phép chuyển đổi quy trình kiểm thử từ rà quét tuyến tính, cứng nhắc sang mô hình định hướng mục tiêu linh hoạt, có khả năng tự lập kế hoạch tấn công dựa trên ngữ cảnh.
- Giải quyết bài toán ảo giác của LLM thông qua MCP: framework AiPT đã tích hợp thành công giao thức MCP, đóng vai trò lớp trùu tượng hóa giúp LLM điều khiển chính xác các công cụ scanner như Burp Suite, Nmap hay Sqlmap và sử dụng trình duyệt thông qua Playwright. Cách tiếp cận này loại bỏ rủi ro AI sinh ra các câu lệnh sai cú pháp hoặc ảo giác, đảm bảo tính thực thi an toàn và hiệu quả.
- Hiện thực hóa mô hình Human-in-the-loop: hệ thống đã phát triển giao diện tương tác trực quan, cho phép chuyên viên kiểm thử giám sát thời gian thực, phê duyệt các hành động nhạy cảm và định hướng chiến lược cho AI. Điều này giúp giảm tải nhận thức cho con người ở các tác vụ lặp lại, đồng thời giữ lại quyền kiểm soát tối cao ở các quyết định quan trọng.
- Hiệu quả thực nghiệm đã được kiểm chứng: qua thử nghiệm trên môi trường Lab và môi trường thực tế, AiPT đã chứng minh khả năng phát hiện chính xác các lỗ hổng nghiêm trọng (Critical/High) như SQL Injection, Cross-Site

Scripting (XSS) và các lỗi cấu hình bảo mật. Hệ thống thể hiện khả năng "hiểu" ngữ cảnh ứng dụng (như nhận diện công nghệ backend, cấu trúc API) vượt trội so với các máy quét DAST truyền thống, đồng thời giảm thiểu đáng kể tỷ lệ dương tính giả.

## 2. Hướng phát triển

Mặc dù đã đạt được những kết quả khả quan, framework AiPT vẫn còn một số hạn chế cần được khắc phục và mở rộng trong tương lai để trở thành một sản phẩm thương mại hoàn chỉnh:

- Tối ưu hóa độ trễ và hiệu năng: hiện tại, thời gian phản hồi của hệ thống phụ thuộc lớn vào tốc độ xử lý của LLM thông qua API, gây ra độ trễ trong quá trình suy luận. Hướng phát triển tiếp theo sẽ tập trung vào việc tinh chỉnh các mô hình ngôn ngữ nhỏ hơn chuyên biệt cho an ninh mạng để triển khai cục bộ (on-premise), giúp tăng tốc độ và bảo mật dữ liệu.
- Tăng cường khả năng phát hiện lỗ hổng Logic nghiệp vụ: hiện tại AiPT làm tốt các lỗi kỹ thuật bề mặt nhưng chưa thực sự có thể đảm nhận hoàn toàn cho các lỗ hổng logic nghiệp vụ (dù vẫn xử lý được các testcase cơ bản). Nghiên cứu tương lai sẽ tập trung nâng cấp cho khả năng của Exploit Agents trong việc học và hiểu các quy trình nghiệp vụ phức tạp (như quy trình thanh toán, đặt hàng) để phát hiện các lỗ hổng Business Logic – nơi mà các công cụ tự động hiện nay vẫn còn yếu.
- Nâng cấp khả năng scripting: mở rộng khả năng của module thực thi mã để Agent có thể tự động viết và chạy các đoạn script khai thác (Python/Bash) tùy chỉnh phức tạp hơn, thay vì quá phụ thuộc vào các công cụ có sẵn như sqlmap hay nuclei.

- Tích hợp báo cáo tuân thủ tiêu chuẩn: phát triển module Agent có khả năng tự động ánh xạ các lỗ hổng phát hiện được với các tiêu chuẩn bảo mật quốc tế như PCI-DSS, ISO 27001 hay HIPAA, hỗ trợ doanh nghiệp trong việc đánh giá tuân thủ tự động.

## **TÀI LIỆU THAM KHẢO**

- [1] The PTES Team. (2012). The Penetration Testing Execution Standard. <http://www.pentest-standard.org/>
- [2] Grance, T., Scarfone, K., Souppaya, M., & Stevens, M. (2008, September). Technical Guide to Information Security Testing and Assessment (NIST Special Publication 800-115). National Institute of Standards and Technology. <https://csrc.nist.gov/publications/detail/sp/800-115/final>
- [3] Herzog, P. (2010). Open Source Security Testing Methodology. Institute for Security and Open Methodologies. <https://www.isecom.org/OSSTMM.3.pdf>
- [4] Open Information Systems Security Group. (2006). Information Systems Security Assessment Framework (ISSAF). <https://www.oissg.org/issaf/>
- [5] T. Yarphel and D. Rani, “Comparative Performance Analysis of Web Vulnerability Scanners,” International Journal of Information Security, vol. 4, no. 1, pp. 98–110, Jun. 2025. DOI: 10.34218/IJIS\_04\_01\_005.
- [6] A. Postolache, “Benchmarking our Website Vulnerability Scanner and 5 others,” Pentes-Tools.com Blog, Jun. 26, 2024. [Online]. Available: <https://pentest-tools.com/blog/web-app-vulnerability-scanner-benchmark-2024>.
- [7] N. Jones, “Business Logic Testing: Why Your Scanner Can’t Find What It Doesn’t Understand,” StackHawk Blog, Oct. 17, 2025. [Online]. Available: <https://www.stackhawk.com/blog/testing-for-business-logic-vulnerabilities>.
- [8] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," Information, vol. 11, no. 1, p. 6, Jan. 2020, doi: 10.3390/info11010006.

- [9] Clintswood, D. G. Lie, L. Kuswandana, Nadia, S. Achmad, and D. Suhartono, "The Usage of Machine Learning on Penetration Testing Automation," in 2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), 2023, pp. 322-327, doi: 10.1109/ICE3IS59323.2023.10335188.
- [10] H. Liu, C. Liu, X. Wu, Y. Qu, "An Automated Penetration Testing Framework Based on Hierarchical Reinforcement Learning," *Electronics*, vol. 13, no. 21, art. no. 4311, 2024. DOI: 10.3390/electronics13214311.
- [11] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng and Y. Zhong, "VulDeePecker: A Deep Learning-Based System for Vulnerability Detection," arXiv:1801.01681 [cs.CR], Jan. 2018. Available: <https://arxiv.org/abs/1801.01681>
- [12] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu and Z. Chen, "SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities," arXiv:1807.06756 [cs.LG], Jul. 2018. [Online]. Available: <https://arxiv.org/abs/1807.06756>
- [13] Y. Zhou, S. Liu, J. Siow, X. Du and Y. Liu, "Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks," in \*Advances in Neural Information Processing Systems 32 (NeurIPS 2019)\*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett (Eds.), Curran Associates, Inc., 2019. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/49265d2447bc3bbfe9e76306ce40a31f-Paper.pdf>
- [14] J. Hao and Y.-W. Kwon, "Enhancing Graph-Based Vulnerability Detection through Standardized Deep Learning Pipelines," in \*Proc. 2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)\*, Sanya, China, Dec. 17–21, 2024, doi: 10.1109/TrustCom63139.2024.00174.

- [15] M. Y. Tanko, A. B. M. Sultan, M. H. Osman and H. Zulzalil, “An Approach for Vulnerability Detection in Web Applications Using Graph Neural Networks and Transformers”, Journal of Theoretical and Applied Information Technology, Jan. 2025. Available: <https://www.jatit.org/volumes/Vol103No1/22Vol103No1.pdf>
- [16] N. Risse and M. Böhme, “Uncovering the Limits of Machine Learning for Automatic Vulnerability Detection,” in Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, Aug. 2024. Available: <https://doi.org/10.48550/arXiv.2306.17193>
- [17] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, “PentestGPT: An LLM-empowered automatic penetration testing tool”, Jun. 2024. Available: <https://doi.org/10.48550/arXiv.2308.06782>
- [18] Y. Ginige, A. Niroshan, S. Jain, and S. Seneviratne, “AutoPentester: An LLM Agent-based Framework for Automated Pentesting,” in Proc. IEEE TrustCom, 2025.
- [19] Zytech Digital, "XBOW AI Review", Zytech Digital, November 21, 2024. [Online]. Available: <https://zytechdigital.com/xbow-ai-review/>.
- [20] Chen, C.-H., Liu, H.-T., Su, T.-C., & Lai, J.-Y. (2025). High-Efficiency and High-Power-Density Inverter Design for Electric Vehicle Applications. *Applied Sciences*, 15(16), 9096. <https://doi.org/10.3390/app15169096>