ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

ĐỀ TÀI:

Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu

Người hướng dẫn: **TS. NGUYỄN NĂNG HÙNG VÂN**

Sinh viên thực hiện: **NGÔ ĐỖ NGUYỄN HẢI SƠN**

Số thẻ sinh viên: **102200191**

Lớp**: 20TCLC-DT4**

Đà Nẵng, 01/2026

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: AN TOÀN THÔNG TIN

ĐỀ TÀI:

Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu

Người hướng dẫn: **TS. NGUYỄN NĂNG HÙNG VÂN**

Sinh viên thực hiện: **NGÔ ĐỖ NGUYỄN HẢI SƠN**

Số thẻ sinh viên: **102200191**

Lớp**: 20TCLC-DT4**

Đà Nẵng, 01/2026

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP

1. Thông tin chung:

1. Họ và tên sinh viên: Ngô Đỗ Nguyễn Hải Sơn
2. Lớp: 20TCLC_DT4 Số thẻ SV: 102200191
3. Tên đề tài: Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu.
4. Người hướng dẫn: TS. Nguyễn Năng Hùng Vân Học hàm/ học vị: Tiến sĩ

II. Nhận xét, đánh giá đồ án tốt nghiệp:

1. Về tính cấp thiết, tính mới, khả năng ứng dụng của đề tài: (điểm tối đa là 2đ)

   ……..
   ……..
2. Về kết quả giải quyết các nội dung nhiệm vụ yêu cầu của đồ án: (điểm tối đa là 4đ)

   ……..
   ……..
3. Về hình thức, cấu trúc, bố cục của đồ án tốt nghiệp: (điểm tối đa là 2đ)

   ……..
   ……..
4. Đề tài có giá trị khoa học/ có bài báo/ giải quyết vấn đề đặt ra của doanh nghiệp hoặc nhà trường: (điểm tối đa là 1đ)

   ……..
   ……..
5. Các tồn tại, thiếu sót cần bổ sung, chỉnh sửa:

   ……..
   ……..

III. Tinh thần, thái độ làm việc của sinh viên: (điểm tối đa 1đ)

……..

IV. Đánh giá:

1. Điểm đánh giá: ……../10 (lấy đến 1 số lẻ thập phân)
2. Đề nghị: ☐ Được bảo vệ đồ án ☐ Bổ sung để bảo vệ ☐ Không được bảo vệ

Đà Nẵng, ngày tháng năm 2026

Người hướng dẫn

TÓM TẮT

Tên đề tài: Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu

Sinh viên thực hiện: Ngô Đỗ Nguyễn Hải Sơn

Mã số sinh viên: 102200191

Lớp: 20TCLC_DT4

Trong bối cảnh chuyển đổi số mạnh mẽ hiện nay, các hệ thống công nghệ thông tin và mạng máy tính ngày càng phải đối mặt với nhiều mối đe doạ an ninh mạng phức tạp và tinh vi, diễn ra đồng thời ở nhiều tầng khác nhau như mạng, hệ thống và ứng dụng. Do đó, việc phát hiện sớm và phân tích chính xác các mối đe doạ an ninh mạng đóng vai trò hết sức quan trọng trong việc bảo vệ dữ liệu và đảm bảo an toàn cho hệ thống.

Đề tài "Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu" nhằm nghiên cứu và phát triển một ứng dụng có khả năng tự động thu thập, phân tích và nhận diện các hành vi tấn công mạng thông qua việc áp dụng các mô hình học sâu (Deep Learning). Ứng dụng cho phép xử lý dữ liệu lớn từ nhiều nguồn khác nhau, học được các đặc trưng phức tạp của các mối đe doạ, từ đó nâng cao độ chính xác và hiệu quả so với các phương pháp phát hiện truyền thống.

Bên cạnh chức năng phát hiện, ứng dụng còn hỗ trợ phân tích, phân loại và trực quan hoá các mối đe doạ an ninh mạng theo từng tầng, giúp người quản trị hệ thống dễ dàng theo dõi, đánh giá mức độ nguy hiểm và đưa ra các biện pháp phòng chống kịp thời. Đề tài hướng tới việc xây dựng một giải pháp an ninh mạng thông minh, có khả năng mở rộng và ứng dụng thực tiễn trong các hệ thống mạng hiện đại.

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Ngô Đỗ Nguyễn Hải Sơn Số thẻ sinh viên: 102200191

Lớp: 20TCLC_DT4 Khoa: Công nghệ thông tin Ngành: An toàn thông tin

1. *Tên đề tài đồ án: Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu.*
2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu: Không có.*
4. *Nội dung các phần thuyết minh và tính toán:*
   - *Tổng quan đề tài: Mục tiêu, ý nghĩa, công nghệ, kỹ thuật sử dụng và kết quả kỳ vọng.*
   - *Cơ sở lý thuyết: Các nền tảng, khái niệm và phương pháp áp dụng trong đề tài.*
   - *Phân tích và thiết kế hệ thống: Yêu cầu chức năng/phi chức năng và thiết kế bằng các biểu đồ.*
   - *Triển khai thực tế: Môi trường triển khai và kết quả đạt được khi vận hành.*
   - *Kết luận và hướng phát triển: Đánh giá kết quả và đề xuất cải tiến.*
   - *Tài liệu tham khảo: Danh mục nguồn sử dụng trong đề tài.*
5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ): Không có*
6. *Họ tên người hướng dẫn:* TS. Nguyễn Năng Hùng Vân
7. *Ngày giao nhiệm vụ đồ án: …… /…… / 2025*
8. *Ngày hoàn thành đồ án: …… /…… / 2026*

Đà Nẵng, ngày tháng năm 2026

Trưởng Bộ môn **Người hướng dẫn**

LỜI NÓI ĐẦU VÀ CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc đến các thầy cô trong Khoa Công nghệ thông tin, cũng như tất cả các thầy cô trong trường Đại học Bách khoa – Đại học Đà Nẵng đã dìu dắt, dạy dỗ và truyền đạt kiến thức, kinh nghiệm quý báu của mình trong suốt quá trình em học tập và nghiên cứu tại trường.

Em xin bày tỏ tình cảm và lòng biết ơn chân thành của em tới thầy giáo **TS. Nguyễn Năng Hùng Vân**, người đã từng bước hướng dẫn, giúp đỡ em tận tình trong quá trình thực hiện đồ án tốt nghiệp của mình. Nhờ đó em có thể hoàn thành đồ án đúng tiến độ và tích lũy cho mình nhiều kiến thức quý báu.

Con xin gửi lời cảm ơn to lớn nhất đến cha, mẹ và gia đình. Cha, mẹ và gia đình đã luôn ở bên con, là nguồn động lực không mệt mỏi và là chỗ dựa tinh thần vững chắc giúp con vượt qua những khó khăn để hoàn thành đồ án này.

Mặc dù đã cố gắng hoàn thành đồ án tốt nhất nhưng thời gian và kiến thức còn có hạn nên sẽ không tránh khỏi những thiếu sót nhất định, rất mong được sự cảm thông và tận tình chỉ bảo, góp ý của quý thầy cô giáo cũng như tất cả các bạn để kết quả của em được hoàn thiện hơn.

Một lần nữa em xin chân thành cảm ơn!

Ngô Đỗ Nguyễn Hải Sơn

CAM ĐOAN

Em xin cam đoan:

1. Nội dung trong đồ án này là do em thực hiện dưới sự hướng dẫn trực tiếp của TS Nguyễn Năng Hùng Vân.
2. Các tham khảo dùng trong đồ án đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.
3. Nếu có những sao chép không hợp lệ, vi phạm quy chế đào tạo, em xin chịu hoàn toàn trách nhiệm.

Đà Nẵng, ngày tháng năm 2026

Sinh viên thực hiện

Ngô Đỗ Nguyễn Hải Sơn

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG BIỂU

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Từ	Viết tắt của	Diễn giải
ACCS	Australian Centre for Cyber Security	Đơn vị/nhóm gắn với việc công bố UNSW-NB15.
API	Application Programming Interface	Giao diện truy cập chức năng hệ thống (HTTP endpoints).
AWS	Amazon Web Services	Nền tảng cloud; bối cảnh triển khai.
CORS	Cross-Origin Resource Sharing	Cơ chế kiểm soát truy cập cross-origin trong trình duyệt.
CPU	Central Processing Unit	Bộ xử lý trung tâm của máy chủ.
CSV	Comma-Separated Values	Định dạng tệp bảng, thường dùng xuất/nhập dữ liệu.
DB	Database	Cơ sở dữ liệu dùng lưu cấu hình, kết quả, review…
DNS	Domain Name System	Giao thức phân giải tên miền (dns.log).
EC2	Elastic Compute Cloud	Dịch vụ máy chủ ảo của AWS.
FN	False Negative	Bỏ sót: dự đoán âm nhưng thực tế dương.
FP	False Positive	Báo động nhầm: dự đoán dương nhưng thực tế âm.
FPR	False Positive Rate	Tỷ lệ báo động nhầm theo lớp âm; dùng trong ROC.
GRU	Gated Recurrent Unit	Biến thể RNN có cổng, gọn hơn LSTM.
GUI	Graphical User Interface	Giao diện đồ hoạ cho người dùng vận hành.
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu hiển thị trang web.
HTTP	Hypertext Transfer Protocol	Giao thức web (http.log).
HTTPS	HTTP Secure	HTTP chạy trên TLS (mã hoá).

IDPS	Intrusion Detection and Prevention System	Khái niệm bao trùm IDS/IPS theo NIST; gồm giám sát, phân tích, cảnh báo và phản ứng.
IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập, sinh cảnh báo.
IDS/IPS	Intrusion Detection/Prevention System	Cách viết gộp để chỉ cả IDS và IPS trong cùng ngữ cảnh.
IETF	Internet Engineering Task Force	Tổ chức ban hành chuẩn Internet (ví dụ IPFIX).
IP	Internet Protocol	Giao thức mạng; địa chỉ IP dùng định danh nguồn/đích.
IPFIX	IP Flow Information Export	Chuẩn IETF để xuất bản ghi luồng; RFC 7011 mô tả giao thức IPFIX.
IPS	Intrusion Prevention System	Hệ thống phòng chống xâm nhập, có thể chặn/giảm thiểu tấn công.
JSON	JavaScript Object Notation	Định dạng dữ liệu có cấu trúc; Zeek có thể xuất JSON.
JWT	JSON Web Token	Token xác thực/phiên đăng nhập khi gọi API.
LSTM	Long Short-Term Memory	Biến thể RNN có cổng, học phụ thuộc dài hạn.
LSTM-AE	LSTM Autoencoder	Autoencoder dùng LSTM để tái tạo chuỗi phục vụ phát hiện bất thường.
LSTM/GRU	Long Short-Term Memory / Gated Recurrent Unit	Cách viết gộp hai biến thể RNN có cổng.
MAE	Mean Absolute Error	Độ đo lỗi tuyệt đối trung bình.
MSE	Mean Squared Error	Hàm mất mát/độ đo lỗi bình phương trung bình.
ML	Machine Learning	Học máy; bối cảnh huấn luyện/đánh giá mô hình.
NBA	Network Behaviour Analysis	Phân tích hành vi mạng dựa trên thống kê/đặc trưng lưu lượng.
NIDS	Network-based IDS	IDS triển khai ở tầng mạng, giám sát lưu lượng.

NIPS	Network-based IPS	IPS triển khai ở tầng mạng, can thiệp/ngiăn chặn lưu lượng.
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên; được liên hệ với embedding/token hoá.
OOV	Out-of-Vocabulary	Giá trị/token chưa từng có trong codebook; ánh xạ về UNKNOWN.
PAD	Padding	Token đệm để chuẩn hoá độ dài chuỗi.
PCAP	Packet Capture	Dữ liệu/tệp bắt gói (packet capture) dùng cho phân tích sâu ở mức gói.
PR	Precision–Recall	Đường cong Precision–Recall (hữu ích khi dữ liệu mất cân bằng).
RAM	Random Access Memory	Bộ nhớ chính của hệ thống.
RBAC	Role-Based Access Control	Phân quyền dựa trên vai trò (admin/user…).
REST	Representational State Transfer	Phong cách thiết kế API dựa trên tài nguyên và HTTP method.
RFC	Request for Comments	Tài liệu tiêu chuẩn/đặc tả kỹ thuật do IETF công bố (ví dụ RFC 7011).
RNN	Recurrent Neural Network	Họ mạng nơ-ron hồi tiếp cho dữ liệu chuỗi.
ROC	Receiver Operating Characteristic	Đường cong TPR–FPR theo ngưỡng.
ROC/AUC	ROC curve & AUC	Cách viết gộp đường ROC và diện tích dưới đường cong.
SEQUENCE	Sequence boundary token	Token đánh dấu bắt đầu/kết thúc chuỗi.
SIEM	Security Information and Event Management	Hệ thống thu thập–tương quan–quản trị log/sự kiện an ninh.
SOC	Security Operations Center	Đội/trung tâm vận hành an ninh, xử lý cảnh báo và điều tra.
SSE	Server-Sent Events	Cơ chế server đẩy dữ liệu một chiều qua HTTP để cập nhật gần thời gian thực.
SSL	Secure Sockets Layer	Tên gọi/di sản cho lớp mã hoá; trong Zeek thường dùng ssl.log cho lưu lượng TLS/SSL.

TCP	Transmission Control Protocol	Giao thức vận chuyển hướng kết nối.
TCP/UDP	Transmission Control Protocol / User Datagram Protocol	Cách viết gộp khi nói chung về hai giao thức vận chuyển.
TLS	Transport Layer Security	Cơ chế mã hoá tầng vận chuyển; khiến payload khó phân tích nội dung.
TN	True Negative	Dự đoán đúng lớp âm (bình thường).
TP	True Positive	Dự đoán đúng lớp dương (tấn công/bất thường).
TPR	True Positive Rate	Tỷ lệ bắt đúng dương; thường tương đương Recall/Sensitivity.
TS	Timestamp	Viết tắt thường gặp cho dấu thời gian (ts) trong log.
TTL	Time To Live	Trường TTL (tuổi thọ gói) trong IP; cũng dùng để nói về thời hạn chặn tạm.
UDP	User Datagram Protocol	Giao thức vận chuyển không hướng kết nối.
UID	Unique Identifier	Mã định danh duy nhất (trong Zeek, UID định danh từng kết nối).
UML	Unified Modelling Language	Ngôn ngữ mô hình hoá (use case, sequence, class diagram…).
UNSW-NB15	University of New South Wales network benchmark (2015)	Tập dữ liệu benchmark có nhãn phục vụ IDS/bất thường ở tầng mạng.
UTC	Coordinated Universal Time	Múi giờ chuẩn dùng khi chuẩn hoá thời gian log.
VRAM	Video RAM	Bộ nhớ trên GPU.
XSS	Cross-Site Scripting	Lỗ hổng chèn script khi hiển thị dữ liệu trên web; cần escape khi render.

TÓM TẮT ĐỀ TÀI

Trong bối cảnh các hệ thống mạng và hệ thống máy tính ngày càng phát triển và kết nối rộng rãi, các mối đe doạ an ninh mạng cũng gia tăng cả về số lượng lẫn mức độ phức tạp. Nhiều hình thức tấn công không còn thể hiện qua một hành vi đơn lẻ mà được biểu hiện thông qua chuỗi hành vi liên tiếp, diễn ra đồng thời ở nhiều tầng khác nhau của hệ thống. Do đó, việc phát hiện và phân tích các mối đe doạ an ninh mạng một cách tự động, chính xác và kịp thời là một yêu cầu cấp thiết.

Đề tài "*Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu*" tập trung nghiên cứu và xây dựng một ứng dụng phát hiện bất thường trong an ninh mạng dựa trên dữ liệu chuỗi, áp dụng cho tầng mạng. Trong đề tài, bộ dữ liệu an ninh mạng tiêu biểu là UNSW-NB15 (lưu lượng mạng) được sử dụng để huấn luyện và đánh giá mô hình.

Ứng dụng được xây dựng dựa trên mô hình học sâu DabLog, sử dụng mạng Bi-directional LSTM kết hợp Autoencoder nhằm học cấu trúc hành vi bình thường của các chuỗi sự kiện và phát hiện các chuỗi bất thường thông qua sự sai lệch so với mô hình đã học. Phương pháp này cho phép khai thác mối quan hệ theo thời gian và cấu trúc của chuỗi sự kiện, từ đó nâng cao hiệu quả phát hiện các mối đe doạ so với các phương pháp truyền thống.

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong bối cảnh công nghệ thông tin và truyền thông phát triển mạnh mẽ, các hệ thống mạng và hệ thống thông tin ngày càng được triển khai với quy mô lớn, tính liên kết cao và độ phức tạp ngày càng tăng. Cùng với những lợi ích mang lại, các hệ thống này cũng phải đối mặt với nhiều mối đe doạ an ninh mạng ngày càng tinh vi, đa dạng và khó kiểm soát. Các cuộc tấn công không chỉ diễn ra tại một điểm đơn lẻ mà thường xuất hiện dưới dạng chuỗi hành vi liên tiếp, ảnh hưởng đồng thời đến nhiều tầng khác nhau của hệ thống như tầng mạng, tầng hệ điều hành và tầng ứng dụng.

Trong quá trình vận hành, hầu hết các hoạt động của hệ thống đều được ghi lại dưới dạng dữ liệu nhật ký (log). Dữ liệu log phản ánh chi tiết trạng thái, hành vi và mối quan hệ giữa các sự kiện xảy ra trong hệ thống theo thời gian. Do đó, log đóng vai trò quan trọng trong việc giám sát, phát hiện bất thường và phân tích các sự cố an ninh mạng. Tuy nhiên, khối lượng log phát sinh trong các hệ thống hiện đại là rất lớn, có tính tuần tự, không đồng nhất và chứa nhiều nhiễu, khiến việc phân tích thủ công hoặc dựa trên các luật định sẵn trở nên kém hiệu quả và khó mở rộng.

Các phương pháp phát hiện xâm nhập truyền thống, dựa trên chữ ký tấn công hoặc tập luật chuyên gia, thường chỉ hiệu quả đối với các mối đe doạ đã biết trước. Khi đối mặt với các hành vi tấn công mới hoặc các bất thường mang tính cấu trúc, các phương pháp này dễ bỏ sót hoặc đưa ra nhiều cảnh báo sai. Trước những hạn chế đó, trí tuệ nhân tạo, đặc biệt là các phương pháp học máy và học sâu, đã được nghiên cứu và ứng dụng rộng rãi trong lĩnh vực an ninh mạng.

Học sâu cho phép mô hình tự động học các đặc trưng tiềm ẩn từ dữ liệu lớn và phức tạp, đặc biệt phù hợp với dữ liệu log dạng chuỗi sự kiện theo thời gian. Bằng cách học hành vi bình thường của hệ thống, các mô hình học sâu có thể phát hiện các sai lệch bất thường mà không cần phụ thuộc hoàn toàn vào tri thức chuyên gia hay chữ ký tấn công. Điều này mở ra hướng tiếp cận hiệu quả cho bài toán phát hiện và phân tích các mối đe doạ an ninh mạng trong môi trường thực tế.

Xuất phát từ yêu cầu thực tiễn nêu trên, em quyết định lựa chọn đề tài "Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu", việc nghiên cứu và xây dựng một ứng dụng phát hiện và phân tích các mối đe doạ an ninh

mạng dựa trên trí tuệ nhân tạo và học sâu là cần thiết. Ứng dụng không chỉ hỗ trợ phát hiện sớm các dấu hiệu bất thường từ dữ liệu log mà còn giúp người quản trị hệ thống hiểu rõ hơn mối quan hệ giữa các sự kiện, từ đó nâng cao hiệu quả giám sát và xử lý sự cố an ninh mạng.

2. Mục tiêu và ý nghĩa của đề tài

2.1 Mục tiêu

Mục tiêu của đề tài "Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu" là nghiên cứu và xây dựng một ứng dụng có khả năng phát hiện và phân tích các mối đe doạ an ninh mạng dựa trên dữ liệu chuỗi, thông qua việc áp dụng các kỹ thuật học sâu hiện đại. Cụ thể, đề tài hướng tới các mục tiêu sau:

- Nghiên cứu cơ sở lý thuyết về an ninh mạng, phát hiện bất thường và học sâu trên dữ liệu chuỗi, làm nền tảng cho việc xây dựng hệ thống phát hiện mối đe doạ.
- Xây dựng quy trình xử lý và chuẩn hoá dữ liệu an ninh mạng cho tầng mạng và tầng hệ thống, nhằm chuyển đổi dữ liệu thô thành dạng chuỗi sự kiện phù hợp cho mô hình học sâu.
- Áp dụng mô hình học sâu DabLog dựa trên mạng Bi-directional LSTM và Autoencoder để học cấu trúc hành vi bình thường và phát hiện các chuỗi bất thường.
- Thiết kế và triển khai một ứng dụng cho phép huấn luyện mô hình, phát hiện bất thường, đánh giá kết quả và trực quan hoá dữ liệu phục vụ phân tích an ninh mạng.
- Đánh giá hiệu quả của hệ thống thông qua các chỉ số phù hợp, từ đó phân tích ưu điểm và hạn chế của phương pháp được đề xuất.

2.2 Ý nghĩa

Đề tài có ý nghĩa cả về mặt học thuật và thực tiễn trong lĩnh vực an ninh mạng.Về mặt học thuật, đề tài giúp củng cố và mở rộng kiến thức về an ninh mạng, phát hiện bất thường và học sâu, đặc biệt là các mô hình xử lý dữ liệu chuỗi. Việc nghiên cứu và áp dụng mô hình DabLog giúp minh hoạ cách kết hợp giữa học sâu và dữ liệu an ninh mạng, góp phần làm rõ tiềm năng của các phương pháp học sâu trong việc nâng cao hiệu quả phát hiện mối đe doạ.

Về mặt thực tiễn, ứng dụng được xây dựng trong đề tài có khả năng hỗ trợ người quản trị và người phân tích an ninh trong việc giám sát và phát hiện sớm các hành vi bất

thường ở tầng mạng và tầng hệ thống. Mặc dù được triển khai trong phạm vi một đồ án học tập, hệ thống vẫn có thể làm nền tảng tham khảo cho việc phát triển các giải pháp an ninh mạng thông minh trong thực tế.

Ngoài ra, đề tài còn giúp người thực hiện rèn luyện kỹ năng nghiên cứu, phân tích bài toán, thiết kế hệ thống và triển khai mô hình học sâu trên dữ liệu thực tế, tạo tiền đề cho việc học tập và nghiên cứu chuyên sâu hơn trong lĩnh vực an ninh mạng và trí tuệ nhân tạo trong tương lai.

3. Đối tượng và phạm vi của đề tài

3.1 Đối tượng

Đối tượng của đề tài là các mối đe doạ an ninh mạng được biểu hiện thông qua dữ liệu chuỗi phát sinh trong quá trình vận hành của hệ thống mạng và hệ thống máy tính. Cụ thể, đề tài tập trung nghiên cứu và xử lý các loại dữ liệu lưu lượng mạng và các bản ghi kết nối, phản ánh hành vi truyền thông giữa các thực thể trong hệ thống. Dữ liệu này được khai thác thông qua bộ dữ liệu UNSW-NB15, bao gồm cả lưu lượng bình thường và lưu lượng tấn công, được sử dụng để nghiên cứu và phát hiện các hành vi bất thường ở tầng mạng.

3.2 Phạm vi đề tài

Đề tài "Xây dựng ứng dụng phát hiện và phân tích các mối đe doạ an ninh mạng dựa vào học sâu" được thực hiện trong phạm vi nghiên cứu và ứng dụng ở mức cơ bản đến trung bình, phù hợp với quy mô một đồ án học tập. Cụ thể, phạm vi sử dụng của đề tài bao gồm:

- Ứng dụng tập trung vào việc phát hiện và phân tích một số mối đe doạ an ninh mạng phổ biến, chẳng hạn như tấn công mạng, truy cập trái phép, hành vi bất thường trong lưu lượng mạng hoặc hệ thống, dựa trên dữ liệu thu thập được.
- Hệ thống được xây dựng nhằm mô phỏng và minh hoạ nguyên lý hoạt động của một ứng dụng an ninh mạng sử dụng học sâu, chưa hướng tới việc triển khai trên các hệ thống lớn hoặc môi trường sản xuất thực tế.
- Đề tài sử dụng các tập dữ liệu mẫu hoặc dữ liệu công khai phục vụ cho việc huấn luyện và đánh giá mô hình học sâu, không xử lý dữ liệu nhạy cảm hoặc dữ liệu bảo mật thực tế.
- Phạm vi ứng dụng chủ yếu dành cho mục đích học tập, nghiên cứu và tham khảo, giúp sinh viên hiểu rõ hơn về cách áp dụng học sâu trong lĩnh vực an ninh mạng.

- Chưa đi sâu vào các vấn đề nâng cao như tối ưu hoá hiệu năng ở quy mô lớn, tích hợp với các hệ thống bảo mật thương mại hay triển khai thời gian thực trong môi trường doanh nghiệp.

Thông qua phạm vi sử dụng này, đề tài nhằm cung cấp một nền tảng cơ bản để nghiên cứu và phát triển các giải pháp an ninh mạng thông minh, đồng thời có thể làm cơ sở cho việc mở rộng và nâng cấp trong các nghiên cứu hoặc ứng dụng thực tiễn sau này.

4. Bố cục của báo cáo

Ngoài phần mở đầu trình bày lý do chọn đề tài, mục tiêu nghiên cứu, ý nghĩa khoa học và thực tiễn của đề tài, cùng với phạm vi và đối tượng nghiên cứu, nội dung chính của báo cáo được tổ chức thành 04 chương như sau:

- Chương I: chương này trình bày các khái niệm và nền tảng lý thuyết liên quan đến đề tài. Nội dung tập trung làm rõ các vấn đề cơ bản trong lĩnh vực an ninh mạng, phát hiện mối đe doạ và bất thường, cũng như các kiến thức nền về học sâu được sử dụng trong bài toán an ninh mạng. Chương I đóng vai trò làm cơ sở khoa học để hiểu và triển khai các phương pháp được đề xuất trong các chương tiếp theo.
- Chương II: chương này trình bày chi tiết các phương pháp được sử dụng trong đồ án để xây dựng và huấn luyện mô hình phát hiện mối đe doạ an ninh mạng. Nội dung bao gồm các phương pháp xử lý dữ liệu đầu vào ở tầng mạng, kỹ thuật trích xuất và biểu diễn đặc trưng, mô hình học sâu được lựa chọn, cũng như quy trình huấn luyện và đánh giá mô hình. Bên cạnh đó, chương này cũng mô tả cách thức triển khai mô hình đã huấn luyện lên môi trường máy chủ nhằm phục vụ cho việc vận hành hệ thống trong thực tế.
- Chương III: chương này tập trung vào việc phân tích yêu cầu hệ thống và thiết kế kiến trúc tổng thể của ứng dụng phát hiện và phân tích mối đe doạ an ninh mạng. Nội dung bao gồm xác định các thành phần chính của hệ thống, luồng xử lý dữ liệu, mối quan hệ giữa các mô-đun, cũng như thiết kế kiến trúc phần mềm và các thành phần triển khai nhằm đảm bảo tính mở rộng, hiệu năng và độ ổn định của hệ thống.
- Chương IV: chương này trình bày quá trình hiện thực hoá hệ thống dựa trên thiết kế đã đề xuất. Nội dung bao gồm việc xây dựng các mô-đun xử lý dữ liệu, huấn luyện và tích hợp mô hình học sâu, triển khai hệ thống trên máy chủ, cũng như mô tả các chức năng chính của ứng dụng. Chương này cũng trình bày kết quả

thực nghiệm, quá trình chạy thử và đánh giá hiệu quả hoạt động của hệ thống trong việc phát hiện và phân tích các mối đe doạ an ninh mạng.

- Phần kết luận và hướng phát triển: phần cuối của báo cáo tổng kết các kết quả đạt được trong quá trình nghiên cứu và triển khai đề tài, đánh giá mức độ hoàn thành so với mục tiêu ban đầu đã đề ra. Đồng thời, phần này cũng nêu bật những đóng góp của đề tài về mặt lý thuyết và ứng dụng thực tiễn trong lĩnh vực an ninh mạng. Bên cạnh đó, các hạn chế còn tồn tại và hướng phát triển trong tương lai của hệ thống cũng được trình bày nhằm mở rộng và nâng cao hiệu quả của đề tài trong các nghiên cứu tiếp theo.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 Tổng quan an ninh mạng và các mối đe doạ phổ biến

Trong bối cảnh chuyển đổi số và kết nối liên thông ngày càng sâu rộng, hạ tầng công nghệ thông tin của cơ quan, doanh nghiệp thường xuyên phải đối mặt với nhiều rủi ro từ không gian mạng. An ninh mạng có thể hiểu là tập hợp các giải pháp kỹ thuật và quản trị nhằm bảo vệ hệ thống thông tin trước các hành vi truy cập trái phép, phá hoại hoặc lạm dụng tài nguyên, qua đó đảm bảo ba thuộc tính cơ bản: tính bảo mật (thông tin không bị lộ), tính toàn vẹn (dữ liệu không bị sửa đổi trái phép) và tính sẵn sàng (dịch vụ hoạt động ổn định, liên tục). Về mặt khái niệm, cần phân biệt mối đe doạ (threat) và tấn công (attack). Mối đe doạ là khả năng tiềm ẩn gây hại cho hệ thống (ví dụ: sự tồn tại của botnet trên Internet, lỗ hổng phần mềm chưa vá, hoặc kẻ tấn công có động cơ), còn tấn công là hành vi cụ thể khai thác điểm yếu để đạt mục tiêu như chiếm quyền, đánh cắp dữ liệu hoặc làm gián đoạn dịch vụ. Trong thực tế vận hành, cùng một mối đe doạ có thể dẫn tới nhiều kiểu tấn công khác nhau tuỳ theo mục tiêu và phương thức của đối tượng tấn công.

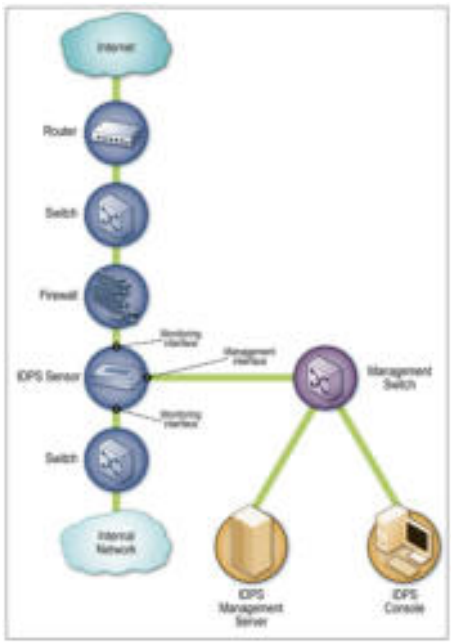
Các mối đe doạ phổ biến trong môi trường mạng thường có thể quan sát qua hành vi lưu lượng và nhật ký (log). Thứ nhất là tấn công từ chối dịch vụ (DoS/DDoS), trong đó đối tượng tấn công tạo lượng truy cập lớn bất thường nhằm làm cạn kiệt tài nguyên của máy chủ hoặc đường truyền, dấu hiệu điển hình là số lượng kết nối/tốc độ gói tin tăng đột biến trong một khoảng thời gian ngắn, gây suy giảm chất lượng dịch vụ. Thứ hai là dò quét và trinh sát (scanning/reconnaissance), tức kẻ tấn công thăm dò cổng dịch vụ, phiên bản ứng dụng, hoặc các điểm mở để chuẩn bị khai thác, hành vi này thường biểu hiện qua nhiều kết nối ngắn tới nhiều cổng/đích khác nhau, lặp lại theo mẫu. Thứ ba là tấn công dò mật khẩu (brute force/credential abuse), thường xuất hiện dưới dạng nhiều lần đăng nhập thất bại hoặc thử nhiều tài khoản trong thời gian ngắn. Bên cạnh đó là các nhóm tấn công nghiêm trọng hơn như khai thác lỗ hổng (exploitation) để thực thi mã, mã độc và điều khiển từ xa (malware/C2) để duy trì hiện diện trong hệ thống, và rò rỉ dữ liệu (data exfiltration) qua các kênh hợp lệ hoặc được nguỵ trang.

Một đặc điểm đáng chú ý của an ninh mạng hiện nay là tính động và khó dự đoán của hành vi tấn công. Nhiều chiến dịch tấn công không còn diễn ra theo một hành vi đơn lẻ mà theo chuỗi bước liên tiếp: trinh sát, xâm nhập, leo thang, di chuyển ngang, đánh cắp dữ liệu. Do đó, nếu chỉ dựa vào một dấu hiệu tại một thời điểm, hệ thống có

thể bỏ sót những cuộc tấn công tinh vi hoặc phát sinh nhiều cảnh báo sai. Đây chính là lý do các hướng tiếp cận hiện đại ngày càng chú trọng đến phân tích dữ liệu theo chuỗi sự kiện theo thời gian (sequential events), kết hợp học máy/học sâu để nhận diện các mẫu bất thường. Trong phạm vi đồ án này, nguồn dữ liệu được khai thác là nhật ký mạng từ Zeek (và các đặc trưng/flow trích xuất tương ứng). Zeek cung cấp thông tin có cấu trúc về kết nối, giao thức và sự kiện mạng, nhờ đó cho phép mô hình hoá hành vi mạng dưới dạng chuỗi sự kiện. Từ nền tảng lý thuyết nêu trên, các phần tiếp theo sẽ trình bày các hướng phát hiện xâm nhập, đặc biệt là tiếp cận dựa trên bất thường và học sâu cho dữ liệu chuỗi, làm cơ sở cho phương pháp đề xuất ở Chương II.

1.2 Giới thiệu hệ thống IDS/IPS và phát hiện tấn công dựa trên bất thường

Trong các giải pháp phòng thủ mạng, IDS (Intrusion Detection System) và IPS (Intrusion Prevention System) là hai thành phần quan trọng nhằm phát hiện (và có thể ngăn chặn) các hành vi xâm nhập [1]. Theo cách tiếp cận chuẩn hoá của NIST, khái niệm thường được dùng ở mức tổng quát là IDPS (Intrusion Detection and Prevention System), bao hàm cả chức năng giám sát, phân tích, cảnh báo và thực thi phản ứng như chặn lưu lượng hoặc cô lập nguồn tấn công. Về bản chất, IDS tập trung vào việc phát hiện và sinh cảnh báo để người quản trị/nhân sự SOC đánh giá, trong khi đó IPS được triển khai "trên đường truyền" (in-line) hoặc ở vị trí có quyền can thiệp, nhằm ngăn chặn theo chính sách (ví dụ: drop gói tin, reset kết nối, đưa địa chỉ vào danh sách chặn). Việc chuyển từ IDS sang IPS giúp giảm thời gian phản ứng, tuy nhiên cũng làm tăng yêu cầu về độ chính xác vì chặn nhầm có thể ảnh hưởng dịch vụ.



Hình 1.1: Ví dụ kiến trúc cảm biến IDPS mạng kiểu inline [1].

Xét theo vị trí triển khai và loại sự kiện giám sát, IDPS thường được phân thành các nhóm chính: Network-based (NIDS/NIPS) giám sát lưu lượng mạng, Host-based (HIDS/HIPS) giám sát nhật ký và hành vi trên máy chủ, ngoài ra còn có nhóm Wireless và Network Behaviour Analysis (NBA) tập trung vào hành vi, đặc trưng lưu lượng ở mức thống kê [2]. Cách phân loại này giúp lựa chọn đúng nguồn dữ liệu và cơ chế triển khai phù hợp mục tiêu bảo vệ.

Về kỹ thuật phát hiện, có hai hướng tiếp cận tiêu biểu. Thứ nhất là phát hiện dựa trên chữ ký (signature-based): hệ thống so khớp sự kiện/lưu lượng với các mẫu đã biết (rules/patterns). Cách làm này hiệu quả với các tấn công phổ biến và có hành vi đặc trưng, đồng thời dễ giải thích kết quả vì có "luật" tương ứng. Ví dụ điển hình là Snort sử dụng tập luật để nhận diện hoạt động độc hại và sinh cảnh báo, đồng thời có thể vận hành theo chế độ IDS/IPS tuỳ cấu hình.

Thứ hai là phát hiện dựa trên bất thường (anomaly-based detection): hệ thống xây dựng "hồ sơ hành vi bình thường" (normal profile) từ dữ liệu quan sát theo thời gian (người dùng, máy chủ, kết nối, ứng dụng…), sau đó dùng các phương pháp thống kê/học máy để phát hiện những sai lệch đáng kể so với hành vi thường lệ. NIST mô tả hướng tiếp cận này như việc tạo profile từ hoạt động điển hình rồi so sánh hoạt động hiện tại với profile để nhận ra bất thường.

So sánh hai hướng tiếp cận, signature-based thường có tỉ lệ cảnh báo sai thấp đối với các mẫu đã biết nhưng hạn chế trong việc phát hiện biến thể mới/zero-day, ngược lại, anomaly-based có khả năng nhận diện hành vi mới lạ nhưng dễ phát sinh false positive nếu ngữ cảnh thay đổi (giờ cao điểm, sự kiện đặc biệt, thay đổi cấu hình hệ thống) [1]. Vì vậy, trong môi trường thực tế, hệ thống thường cần cơ chế ngưỡng, xếp hạng mức độ bất thường, và quy trình review để cân bằng giữa độ nhạy và tính ổn định vận hành.

Trong đồ án, việc sử dụng Zeek đóng vai trò như một thành phần NIDS/NBA: Zeek phân tích lưu lượng và xuất ra các nhật ký có cấu trúc (ví dụ conn.log ghi lại thông tin kết nối, ngoài ra còn có dns.log, http.log, ssl.log…), tạo nguồn dữ liệu đầu vào thuận lợi cho bước trích xuất đặc trưng và mô hình hoá chuỗi sự kiện. [3]

Về dữ liệu phục vụ nghiên cứu/đánh giá mô hình phát hiện xâm nhập, một lựa chọn phổ biến trong học thuật là các bộ dữ liệu benchmark. Ví dụ, UNSW-NB15 được UNSW Canberra công bố, tạo từ lưu lượng bình thường kết hợp hành vi tấn công tổng hợp, có nhiều nhóm tấn công và kèm nhãn phục vụ huấn luyện/đánh giá mô hình IDS. [4]

1.3 Tổng quan dữ liệu phục vụ phát hiện xâm nhập

Trong các hệ thống phát hiện/phòng chống xâm nhập (IDS/IPS), dữ liệu quan sát là nền tảng quyết định khả năng nhận diện tấn công. Về nguyên tắc, IDPS có thể thu thập và lưu vết nhiều loại thông tin khác nhau (mạng, máy chủ, ứng dụng), nhằm phục vụ đồng thời hai mục tiêu: phát hiện (dựa trên mẫu chữ ký hoặc sai lệch hành vi), điều tra và truy vết khi có sự cố [1]. NIST cũng nhấn mạnh việc IDPS thường ghi log chi tiết cho các sự kiện/phiên kết nối và dùng chúng để xác minh cảnh báo, điều tra và tương quan với các nguồn log khác.

1.3.1. Dữ liệu mức gói tin

Dữ liệu mức gói tin (packet/PCAP) ghi lại toàn bộ gói đi qua điểm giám sát (header và có thể cả payload) [1]. Ưu điểm của dạng này là độ "giàu thông tin", cho phép phân tích sâu (ví dụ tái dựng phiên, kiểm tra nội dung ứng dụng). Tuy nhiên, nhược điểm là:

- Dung lượng rất lớn, khó lưu trữ dài hạn ở mạng tốc độ cao.
- Payload ngày càng bị mã hoá (TLS), khiến phân tích nội dung hạn chế, nhiều hệ thống thực tế chuyển hướng sang khai thác metadata và hành vi.

Trong phạm vi đồ án triển khai theo hướng vận hành, dữ liệu gói tin thường phù hợp cho điều tra chuyên sâu theo mẫu, còn hệ thống giám sát liên tục thường ưu tiên flow/log để tối ưu tài nguyên.

1.3.2. Dữ liệu mức luồng

Flow records (NetFlow/IPFIX) có thể hiểu là bản ghi tóm tắt thống kê cho một "luồng" lưu lượng (thường được định danh bằng bộ khoá 5-tuple như IP nguồn/đích, port nguồn/đích, giao thức), kèm các chỉ số như tổng số gói, tổng byte, thời điểm bắt đầu/kết thúc [4]. Các chuẩn/giải pháp phổ biến gồm NetFlow (Cisco) và IPFIX (chuẩn IETF hoá cơ chế xuất thông tin flow). Cisco mô tả NetFlow như công nghệ cung cấp thống kê về các gói "chảy qua" thiết bị, phục vụ giám sát và phân tích mạng/an ninh. Về mặt chuẩn hoá, RFC 7011 xác định IPFIX là giao thức truyền thông tin flow từ bộ xuất (exporter) tới bộ thu (collector) dựa trên bản ghi dữ liệu và bản ghi khuôn mẫu (template). Dữ liệu flow có ưu điểm nổi bật:

- Nhẹ hơn rất nhiều so với PCAP, phù hợp lưu trữ và xử lý thời gian dài.
- Hữu ích để phát hiện bất thường theo hành vi thống kê (tăng đột biến lưu lượng, quét cổng, beaconing…).

Tuy vậy, hạn chế là flow chủ yếu phản ánh tổng hợp, không chứa nội dung ứng dụng, do đó cần kết hợp thêm log sự kiện hoặc đặc trưng ngữ cảnh để tăng khả năng phân biệt.

1.3.3. Dữ liệu mức sự kiện

Trong triển khai NIDS hiện đại, một hướng hiệu quả là sử dụng log sự kiện (event logs) có cấu trúc do hệ thống giám sát sinh ra [4]. Với Zeek, dữ liệu đầu ra là các log theo từng loại giao thức/sự kiện, trong đó:

- conn.log được xem là một trong các log quan trọng nhất, mô tả thông tin kết nối cho cả TCP và UDP.
- Bên cạnh đó, Zeek còn cung cấp nhiều log cốt lõi khác như dns.log, http.log…, phản ánh tương tác ở mức ứng dụng và giúp tăng tính giải thích của cảnh báo.

Ưu điểm của log Zeek là giàu ngữ nghĩa hơn flow (vì có trường theo giao thức), nhưng vẫn nhẹ hơn PCAP và thuận lợi cho thiết kế hệ thống vận hành: dễ lưu theo ngày, dễ truy vấn, dễ dựng dashboard và tương quan giữa nguồn/đích/thời điểm.

Log File	Description	Field Descriptions
conn.log	TCP/UDP/ICMP connections	Conn::Info
dce_rpc.log	Distributed Computing Environment/RPC	DCE_RPC::Info
dhcp.log	DHCP leases	DHCP::Info
dnp3.log	DNP3 requests and replies	DNP3::Info
dns.log	DNS activity	DNS::Info
ftp.log	FTP activity	FTP::Info
http.log	HTTP requests and replies	HTTP::Info
irc.log	IRC commands and responses	IRC::Info
kerberos.log	Kerberos	KRB::Info
ldap.log	LDAP Messages	LDAP::MessageInfo
ldap_search.log	LDAP Searches	LDAP::SearchInfo
modbus.log	Modbus commands and responses	Modbus::Info
modbus_register_change.log	Tracks changes to Modbus holding registers	Modbus::MemmapInfo
mysql.log	MySQL	MySQL::Info

Hình 1.2: Một phần danh sách các file log giao thức mạng do Zeek sinh ra. [4]

1.3.4. Dữ liệu theo chuỗi thời gian và động cơ "sequential"

Một đặc điểm quan trọng của dữ liệu an ninh mạng là tính tuần tự theo thời gian. Nhiều hành vi tấn công không thể hiện rõ ở một bản ghi đơn lẻ, mà lộ diện khi quan sát chuỗi sự kiện (ví dụ: nhiều kết nối ngắn tới nhiều cổng → thử đăng nhập thất bại lặp lại → kết nối dài bất thường → tăng lưu lượng ra ngoài). Vì vậy, trong thực hành phát hiện bất thường, dữ liệu thường được tổ chức lại theo:

- Cửa sổ thời gian (time window): gom sự kiện theo khoảng (ví dụ 1 phút/5 phút/30 phút).
- Chuỗi sự kiện theo thực thể: theo IP nguồn, theo cặp kết nối, theo host, hoặc theo "phiên" logic.

Cách tổ chức này giúp mô hình học được "ngữ cảnh trước–sau", giảm bỏ sót các mẫu tấn công có tính giai đoạn.

1.3.5. Các thách thức chất lượng dữ liệu trong bài toán IDS

Khi sử dụng flow/log cho phát hiện bất thường, một số khó khăn thường gặp gồm:

- Mất cân bằng lớp: dữ liệu bình thường chiếm đa số, khiến mô hình dễ thiên lệch.
- Trôi khái niệm (concept drift): hành vi "bình thường" thay đổi theo thời gian (giờ cao điểm, cập nhật hệ thống, thay đổi chính sách), gây tăng cảnh báo sai nếu không có cơ chế hiệu chỉnh.
- Đồng bộ thời gian và tương quan nguồn: log từ nhiều thành phần có thể lệch timestamp, cần thống nhất chuẩn thời gian để xâu chuỗi sự kiện.
- Giới hạn quan sát do mã hoá: giảm thông tin payload, làm tăng vai trò của metadata (flow + log giao thức).

Các vấn đề này là cơ sở để ở Chương II lựa chọn chiến lược biểu diễn dữ liệu và tiêu chí cảnh báo (ví dụ ngưỡng/xếp hạng bất thường), đồng thời ở Chương III–IV cần có quy trình review và phản ứng phù hợp.

1.4 Giới thiệu học sâu cho dữ liệu chuỗi

Dữ liệu trong an ninh mạng (đặc biệt là log/flow) thường mang tính chuỗi theo thời gian, trong đó ý nghĩa của một sự kiện phụ thuộc vào ngữ cảnh trước – sau. Vì vậy, thay vì chỉ xem từng bản ghi độc lập, các mô hình học sâu cho dữ liệu chuỗi hướng tới việc học một biểu diễn "trạng thái" tích luỹ theo thời gian để nắm bắt quy luật hành vi. Cách tiếp cận này được trình bày hệ thống trong chương về mô hình hoá chuỗi của sách Deep Learning (Goodfellow, Bengio, Courville) [5], trong đó RNN là lớp mô hình nền

tảng và các biến thể có cơ chế "cổng" như LSTM/GRU được giới thiệu để khắc phục hạn chế khi học phụ thuộc dài.

Về nguyên lý, Recurrent Neural Network (RNN) là mạng nơ-ron có kết nối hồi tiếp, cho phép mô hình duy trì một trạng thái ẩn (hidden state) được cập nhật tuần tự khi đọc từng phần tử của chuỗi [5]. Khi "trải" theo thời gian (unroll), RNN tương đương một chuỗi các khối tính toán giống nhau dùng chung tham số, trong đó đầu ra ở thời điểm t t phụ thuộc vào đầu vào hiện tại và trạng thái ở thời điểm $t-1$. Nhờ cấu trúc này, RNN phù hợp với các bài toán như dự đoán phần tử tiếp theo, phân loại chuỗi, hoặc học biểu diễn của hành vi theo tiến trình thời gian.

Để khắc phục vấn đề trên, Long Short-Term Memory (LSTM) được Hochreiter và Schmidhuber đề xuất với ý tưởng cốt lõi là bổ sung một ô nhớ (cell state) và các cổng (gates) nhằm kiểm soát việc ghi–quên–đọc thông tin [6]. Thay vì để trạng thái ẩn biến đổi tự do theo mọi nhiễu động của chuỗi, LSTM học cách "mở/đóng" các cổng để giữ lại thông tin quan trọng trong thời gian dài và chỉ cập nhật khi cần thiết. Công trình gốc nhấn mạnh LSTM được thiết kế để giải quyết bài toán học phụ thuộc dài vốn làm cho các phương pháp dựa trên lan truyền ngược trong RNN truyền thống huấn luyện rất chậm hoặc kém hiệu quả.

Bên cạnh LSTM, Gated Recurrent Unit (GRU) là một biến thể có cơ chế "cổng" nhưng kiến trúc gọn hơn, thường gộp/đơn giản hoá một số thành phần so với LSTM. GRU được giới thiệu trong dòng nghiên cứu RNN Encoder–Decoder và sau đó được đánh giá thực nghiệm rộng rãi, cho thấy GRU có thể đạt hiệu năng tương đương LSTM trên một số bài toán mô hình hoá chuỗi, đồng thời có ưu điểm về đơn giản hoá tham số và triển khai.

Bảng 1.1: So sánh tổng quan nhanh RNN, LSTM và GRU.

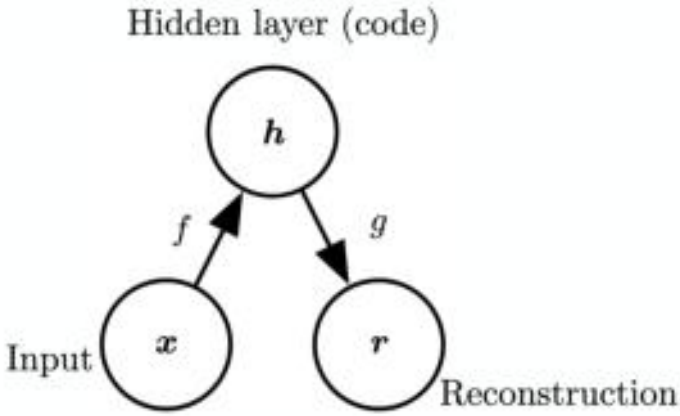
Tiêu chí	**RNN (truyền thống)**	**LSTM**	**GRU**
Mức độ phức tạp	Đơn giản nhất	Phức tạp hơn	Đơn giản hơn LSTM
Số cổng (gates)	Không có cổng	3 (Forget, Input, Output)	2 (Update, Reset)
Trạng thái bộ nhớ	Chỉ Hidden state	Hidden state + Cell state	Chỉ Hidden state

Khả năng ghi nhớ dài hạn	Kém	Rất tốt	Tốt
Vấn đề gradient	Dễ bị vanishing/exploding	Giảm đáng kể	Giảm đáng kể
Số tham số	Ít nhất	Nhiều nhất	Ít hơn LSTM
Tốc độ huấn luyện	Nhanh	Chậm hơn	Nhanh hơn LSTM
Nguy cơ overfitting	Thấp	Cao hơn	Thấp hơn LSTM
Hiệu quả với chuỗi dài	Không hiệu quả	Rất hiệu quả	Hiệu quả
Ứng dụng điển hình	Chuỗi ngắn, bài toán đơn giản	NLP, log analysis, time-series dài	NLP, anomaly detection, real-time

Trong bối cảnh phát hiện bất thường/an ninh mạng, các mô hình RNN có cổng (LSTM/GRU) đặc biệt phù hợp khi dữ liệu được tổ chức thành chuỗi sự kiện theo thực thể (ví dụ theo IP nguồn, theo phiên, theo cửa sổ thời gian). Khi đó, mô hình có thể học "nhịp" hành vi bình thường (tần suất, trật tự sự kiện, sự chuyển trạng thái) và phát hiện các sai lệch theo tiến trình—điều khó đạt được nếu chỉ dùng đặc trưng tĩnh từng bản ghi. Phần này là cơ sở lý thuyết để Chương II trình bày cách chuyển đổi log/flow thành chuỗi và lựa chọn kiến trúc học sâu cho bài toán phát hiện bất thường.

1.5 Autoencoder cho phát hiện bất thường

Autoencoder là một dạng mạng nơ-ron học biểu diễn (representation learning) theo cơ chế mã hoá – giải mã (encoder–decoder). Mục tiêu của autoencoder là học một hàm nén dữ liệu đầu vào x sang không gian tiềm ẩn $\mathrm{z} = \mathrm{f}(\mathrm{x})$ và sau đó tái tạo lại $\mathrm{x}' = \mathrm{g}(\mathrm{z})$ sao cho x' gần với x nhất theo một hàm mất mát tái tạo (reconstruction loss). Khái niệm và các biến thể (undercomplete, sparse, denoising, …) được trình bày có hệ thống trong Deep Learning (Goodfellow, Bengio, Courville) [5], nơi autoencoder được xem như một công cụ học đặc trưng hiệu quả khi dữ liệu có cấu trúc và nhiễu. Trong thực tế, autoencoder thường được dùng để giảm chiều và trích xuất đặc trưng từ dữ liệu chiều cao. Công trình kinh điển của Hinton & Salakhutdinov cho thấy deep autoencoder có thể học mã hoá chiều thấp hữu ích cho dữ liệu phức tạp, tạo nền tảng cho việc dùng tầng tiềm ẩn như một "đặc trưng học được" thay vì đặc trưng thủ công.



Hình 1.3: Cấu trúc tổng quát của một autoencoder [5].

Ánh xạ một đầu vào x thành một đầu ra (gọi là bản tái tạo) r thông qua một biểu diễn nội bộ hay mã hoá tiềm ẩn h. Autoencoder gồm hai thành phần: bộ mã hoá f (ánh xạ x → h) và bộ giải mã g (ánh xạ h → r). [5]

Từ cơ chế "tái tạo", autoencoder có thể được khai thác cho phát hiện bất thường theo hướng không/ít giám sát. Lập luận cơ bản là: nếu mô hình được huấn luyện chủ yếu trên dữ liệu bình thường, nó sẽ học tốt cấu trúc và quy luật của dữ liệu bình thường, từ đó tái tạo tốt các mẫu bình thường. Ngược lại, với các mẫu bất thường (khác phân phối, khác cấu trúc), mô hình thường tái tạo kém, dẫn tới sai số tái tạo (reconstruction error) lớn hơn. Khi đó, ta định nghĩa một điểm bất thường (anomaly score) dựa trên sai số tái tạo (ví dụ MSE/MAE hoặc tổng sai khác theo từng đặc trưng), và so sánh với một ngưỡng để quyết định cảnh báo. Cách tư duy "reconstruction-based" này là nền tảng phổ biến trong các hệ thống anomaly detection dùng autoencoder.

Một vấn đề quan trọng khi áp dụng autoencoder cho phát hiện bất thường là xác định ngưỡng. Trong triển khai thực tế, ngưỡng thường được hiệu chỉnh dựa trên tập dữ liệu kiểm định/validation (ưu tiên dữ liệu bình thường), chẳng hạn chọn theo percentile của phân phối sai số tái tạo hoặc tối ưu theo mục tiêu vận hành (giảm false positive nhưng vẫn giữ độ nhạy). Với dữ liệu chuỗi và môi trường biến thiên theo thời gian, việc theo dõi phân phối sai số và cập nhật ngưỡng theo từng giai đoạn cũng là một hướng thường được cân nhắc để giảm ảnh hưởng của concept drift.

Đối với dữ liệu chuỗi (log/flow theo thời gian), autoencoder thường được hiện thực dưới dạng sequence-to-sequence autoencoder sử dụng RNN có cổng (điển hình là LSTM Autoencoder). Khi đó, encoder đọc một chuỗi độ dài cố định và nén thành biểu diễn tiềm ẩn, decoder cố gắng tái tạo lại toàn bộ chuỗi ban đầu. Nếu chuỗi phản ánh hành vi bình thường, sai số tái tạo sẽ thấp, còn chuỗi chứa hành vi bất thường (chẳng

hạn trình tự sự kiện hiếm gặp hoặc biến đổi đột ngột), sai số sẽ tăng và trở thành tín hiệu cảnh báo. Hướng tiếp cận LSTM-AE kiểu "reconstruction-based threshold" đã được sử dụng trong nhiều bài toán anomaly detection theo time-series, bao gồm cả ngữ cảnh an ninh (ví dụ phát hiện DDoS theo chuỗi đặc trưng đa biến).

Về ưu điểm, autoencoder phù hợp với bài toán an ninh mạng khi dữ liệu nhãn hạn chế: mô hình có thể học từ dữ liệu bình thường nhiều hơn dữ liệu tấn công, và cho phép xây dựng một thước đo "mức độ khác thường" liên tục thay vì phân loại cứng ngay từ đầu. Tuy nhiên, hạn chế đáng chú ý là nếu dữ liệu huấn luyện đã "lẫn" nhiều bất thường hoặc hành vi bình thường thay đổi mạnh theo thời gian, autoencoder có thể học tái tạo cả các mẫu không mong muốn, làm giảm khả năng phân biệt. Do đó, trong thiết kế hệ thống thực tế, autoencoder thường được kết hợp với tiền xử lý/chuẩn hoá tốt, quy trình review, và cơ chế giám sát drift để quyết định thời điểm tái huấn luyện.

1.6 Khái niệm biểu diễn sự kiện/log

Trong bài toán phát hiện bất thường dựa trên học sâu, chất lượng của bước biểu diễn dữ liệu có ảnh hưởng trực tiếp đến hiệu quả mô hình. Dữ liệu thô (raw logs) thường bao gồm nhiều trường dạng chuỗi, số, địa chỉ mạng, trạng thái phiên, … và có thể biến thiên mạnh theo thời gian. Do đó, trước khi đưa vào mô hình, cần chuyển log về dạng chuỗi sự kiện rời rạc (discrete events) và sau đó ánh xạ sang biểu diễn số học (numeric representation) sao cho vừa giữ được thông tin hành vi, vừa giảm nhiễu và độ thưa.

1.6.1. Khái niệm log có cấu trúc và dữ liệu đầu vào từ Zeek

Trong hệ thống của đồ án, nguồn dữ liệu chính là log Zeek. Ưu điểm của Zeek là xuất log theo schema rõ ràng, có thể ở định dạng TSV truyền thống hoặc JSON, giúp việc đọc – chuẩn hoá – trích xuất trường dữ liệu ổn định hơn so với log dạng tự do. Tài liệu Zeek mô tả cơ chế log format, header metadata và schema hỗ trợ việc hiểu ý nghĩa từng trường. Ví dụ, conn.log là log nền tảng phản ánh các kết nối TCP/UDP, Zeek cũng cung cấp tài liệu hướng dẫn cách sử dụng thông tin trong conn.log và cách tra cứu đầy đủ trường dữ liệu qua Conn::Info.

Tuy nhiên, dù đã có cấu trúc, log Zeek vẫn có nhiều trường "độ phân giải cao" (như IP/port/uid) khiến số lượng giá trị khác nhau rất lớn. Nếu đưa trực tiếp các giá trị này vào mô hình như nhãn rời rạc, không gian đặc trưng sẽ bị rất thưa và khó học. Vì vậy, bước tiếp theo thường là chuẩn hoá thành sự kiện thông qua log key.

1.6.2. Khái niệm log key và mục tiêu của việc tạo sự kiện

Log key (có thể hiểu như event key hoặc template/event type) là một nhãn đại diện cho "loại sự kiện", nhằm gom các bản ghi có cùng bản chất hành vi về cùng một mã sự kiện, đồng thời tách phần biến thiên (tham số) khỏi phần ổn định:

- Với log có cấu trúc như Zeek, log key thường được xây dựng bằng cách chọn một tập trường có tính khái quát (ví dụ giao thức, dịch vụ, trạng thái kết nối, nhóm hành vi), và (tuỳ mục tiêu) rời rạc hoá một số trường số. Cách này giúp giảm số lượng "loại sự kiện" và làm rõ hình thái hành vi theo thời gian.
- Với log dạng tự do (free-form logs), log key thường được tạo bằng log parsing/template extraction: biến thông điệp log thành một "mẫu" (template) bằng cách giữ lại phần hằng (constant tokens) và thay các tham số biến thiên (IDs, số, đường dẫn…) bằng ký hiệu đại diện. Thuật toán Drain là một phương pháp tiêu biểu, thiết kế để parse log theo luồng (streaming) bằng cây phân tích độ sâu cố định.

Trong nghiên cứu phát hiện bất thường từ log, một hướng rất phổ biến là:

- trích template.
- ánh xạ template thành event ID.
- mô hình hoá chuỗi event IDs.

DeepLog là một ví dụ kinh điển theo hướng coi log như "chuỗi ngôn ngữ" và học mô hình chuỗi để phát hiện bất thường.

1.6.3. Khái niệm codebook và tính nhất quán

Sau khi có log key, hệ thống cần một codebook (từ điển mã hoá sự kiện) để ánh xạ mỗi log key sang một chỉ số nguyên (event ID). Codebook có vai trò đặc biệt quan trọng trong triển khai thực tế vì:

- Đảm bảo nhất quán giữa giai đoạn huấn luyện và giai đoạn vận hành (inference).
- Cho phép xử lý các sự kiện mới bằng cơ chế unknown/out-of-vocabulary (OOV), tránh lỗi khi gặp log chưa từng xuất hiện.

Trong huấn luyện mô hình, quy trình này được thực hiện dưới dạng xây dựng codebook và sử dụng các token đặc biệt (ví dụ pad/begin/end/unknown) để chuẩn hoá chuỗi và phục vụ huấn luyện/suy luận ổn định.

1.6.4. Khái niệm Embedding

Nếu chỉ dùng one-hot cho event ID, vector đầu vào sẽ rất lớn và thưa. Thay vào đó, mô hình thường dùng embedding (từ event ID sang vector đặc trưng liên tục), một ma trận tham số học được, biến mỗi event ID thành một vector dày (dense vector) có số chiều nhỏ hơn nhiều. Ý tưởng "biểu diễn phân tán" (distributed representation) đã được khai thác rộng rãi trong NLP, các công trình như Word2Vec cho thấy việc học vector liên tục giúp nắm bắt quan hệ ngữ nghĩa giữa các token.

Ở góc độ triển khai, lớp Embedding trong Keras/TensorFlow là cơ chế chuẩn để ánh xạ chỉ số nguyên sang vector đặc trưng cố định, thông qua tham số input_dim (kích thước từ vựng) và output_dim (số chiều embedding). Trong bối cảnh log an ninh mạng, embedding giúp mô hình:

- Học quan hệ "gần nhau" giữa các loại sự kiện có hành vi tương tự.
- Giảm nhiễu do số lượng loại sự kiện lớn.
- Tạo đầu vào phù hợp cho các mô hình chuỗi (như LSTM autoencoder).

1.7 Khái niệm độ đo của mô hình

Để đánh giá chất lượng một mô hình phân loại/phát hiện bất thường trong an ninh mạng, báo cáo thường sử dụng các chỉ số định lượng dựa trên so sánh giữa nhãn thực và kết quả dự đoán. Nền tảng của các chỉ số này là ma trận nhầm lẫn (confusion matrix), trong đó phần tử $C_{i,j}$ biểu diễn số mẫu thuộc lớp thật i nhưng được dự đoán thành lớp j [7]. Với bài toán nhị phân (bình thường/bất thường hoặc bình thường/tấn công), ma trận nhầm lẫn quy về bốn lượng cơ bản:

- TP (True Positive): dự đoán tấn công và thực tế tấn công.
- FP (False Positive): dự đoán tấn công nhưng thực tế bình thường (báo động giả).
- TN (True Negative): dự đoán bình thường và thực tế bình thường.
- FN (False Negative): dự đoán bình thường nhưng thực tế tấn công (lọt tấn công).

Từ các đại lượng trên, Accuracy (độ chính xác tổng thể) được hiểu là tỷ lệ dự đoán đúng trên toàn bộ mẫu, tức:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1.1}$$

Accuracy dễ diễn giải và phù hợp khi dữ liệu cân bằng, tuy nhiên trong an ninh mạng, dữ liệu thường mất cân bằng mạnh (bình thường chiếm đa số), nên một mô hình có thể đạt Accuracy cao nhưng vẫn bỏ sót phần lớn tấn công (FN cao). Vì vậy, Accuracy thường chỉ được dùng như chỉ số tham khảo, cần đi kèm các chỉ số nhấn mạnh lớp

"dương" (tấn công/bất thường). Hai chỉ số quan trọng nhất trong bối cảnh này là Precision và Recall. Theo định nghĩa chuẩn trong tài liệu scikit-learn:

- Precision đo mức "đáng tin" của cảnh báo, trong tất cả những gì hệ thống báo là tấn công, có bao nhiêu phần trăm là tấn công thật, là tỷ lệ mẫu dự đoán dương thực sự là dương:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.2}$$

- Recall (còn gọi là sensitivity/TPR) đo khả năng "bắt đủ" tấn công, trong tất cả các tấn công thật, hệ thống bắt được bao nhiêu phần trăm, là tỷ lệ mẫu dương thật được phát hiện:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.3}$$

Trong vận hành an ninh mạng, Precision cao giúp giảm tải cho người trực (ít cảnh báo nhầm), còn Recall cao giúp giảm rủi ro bỏ lọt tấn công. Tuỳ mục tiêu hệ thống (ưu tiên không bỏ sót hay ưu tiên giảm báo động nhầm), điểm làm việc (operating point) có thể được lựa chọn bằng ngưỡng quyết định khác nhau [7]. Để tổng hợp đồng thời Precision và Recall, F1- score được dùng rộng rãi như trung bình điều hoà, là một điểm cân bằng giữa bắt được tấn công (Recall) và độ tin cậy cảnh báo (Precision):

$$\text{F1} - \text{score} = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{1.4}$$

F1-score tăng khi cả Precision và Recall đều tốt, và giảm mạnh nếu một trong hai yếu. Đây là chỉ số phù hợp để so sánh mô hình trong điều kiện mất cân bằng khi ta cần một thước đo gọn, nhưng vẫn phản ánh "đánh đổi" giữa báo động nhầm và bỏ sót.

Với các mô hình xuất ra điểm số (probability/score) thay vì nhãn cứng, đánh giá theo nhiều ngưỡng là cần thiết. ROC curve biểu diễn quan hệ giữa TPR (Recall) và FPR qua các ngưỡng, AUC (Area Under ROC Curve) tóm tắt khả năng phân biệt của mô hình trên toàn bộ dải ngưỡng. ROC/AUC được trình bày kinh điển trong bài "An introduction to ROC analysis" của Fawcett, đồng thời được hỗ trợ trực tiếp trong scikit-learn (roc_curve, roc_auc_score). Tuy nhiên, trong bài toán mất cân bằng, nhiều nghiên cứu khuyến nghị sử dụng thêm Precision–Recall curve (PR curve) và các biến thể diện tích dưới đường cong, vì PR tập trung trực tiếp vào chất lượng lớp dương (tấn công/bất thường). Saito & Rehmsmeier chỉ ra PR plot thường "thông tin hơn" ROC plot trong trường hợp dữ liệu mất cân bằng, do ROC có thể cho cảm giác "đẹp" ngay cả khi Precision ở mức thấp. [7]

Trong thực hành, scikit-learn thường dùng Average Precision (AP) để tóm tắt PR curve (gần tương ứng PR–AUC theo cách xấp xỉ chuẩn hoá), và cung cấp hàm average_precision_score cùng ví dụ minh hoạ PR curve. Ngoài ra, khi dữ liệu mất cân bằng mạnh, báo cáo cũng có thể bổ sung Balanced Accuracy (trung bình Recall theo các lớp) nhằm tránh việc lớp đa số chi phối kết quả. Trường hợp phương pháp của hệ thống dựa trên xếp hạng Top-N (dự đoán đúng nếu nhãn thật nằm trong N nhãn có điểm cao nhất), có thể ghi thêm Top-k accuracy như một chỉ số phù hợp với cơ chế ra quyết định theo thứ hạng. [7]

1.8 Nghiên cứu liên quan về phát hiện bất thường từ log/chuỗi sự kiện

1.8.1. Bối cảnh dữ liệu

Trong bài toán phát hiện mối đe doạ dựa trên hành vi, dữ liệu mạng thường được quan sát dưới dạng log/flow có cấu trúc, cho phép mô hình hoá trình tự sự kiện theo thời gian. Hệ sinh thái Zeek cung cấp nhiều loại log theo tầng giao thức; trong đó conn.log phản ánh tổng quan kết nối lớp 3–4 (ai kết nối tới ai, trạng thái kết nối, thời lượng, lưu lượng…), http.log bổ sung ngữ nghĩa tầng ứng dụng (phương thức, host, URI, mã phản hồi…), và ftp.log hỗ trợ nhận diện các mẫu truyền tệp/đăng nhập đặc thù của FTP. Việc kết hợp ba nguồn log này giúp vừa mô hình hoá hành vi nền (baseline connectivity) vừa tăng khả năng "giải thích" cảnh báo khi cần truy vết theo ngữ cảnh ứng dụng.

Trong hướng tiếp cận dựa trên chuỗi (sequence-based), log thô thường được trừu tượng hoá thành các "token" rời rạc (ví dụ log key/event id), rồi ghép thành chuỗi theo subject (định danh phiên/định danh thực thể theo thiết kế của hệ thống) và theo cửa sổ thời gian. Từ đó, bài toán phát hiện bất thường chuyển thành bài toán học "ngữ pháp" của chuỗi sự kiện: chuỗi bình thường có mẫu chuyển tiếp ổn định, còn chuỗi tấn công/đột biến có xu hướng phá vỡ mẫu.

1.8.2. Nhóm học máy truyền thống dựa trên ngoại lai

Các phương pháp như One-Class SVM (OC-SVM) và Local Outlier Factor (LOF) là lựa chọn phổ biến khi cần baseline mạnh và dễ triển khai

- OC-SVM học "vùng" của dữ liệu bình thường, điểm nằm ngoài vùng bị xem là bất thường.
- LOF đánh giá mức "lạ" bằng cách so sánh mật độ lân cận; điểm có mật độ thấp bất thường so với hàng xóm thường bị coi là ngoại lai.

Điểm chung của nhóm này là cần biến dữ liệu đầu vào thành vector đặc trưng (feature vector), ví dụ: tần suất xuất hiện log key trong một cửa sổ, thống kê bytes/duration, hoặc đặc trưng n-gram đơn giản. Ưu điểm và hạn chế:

- Ưu điểm: chạy nhanh, ít phụ thuộc kiến trúc DL, phù hợp làm baseline và hữu ích khi đặc trưng thống kê đã đủ mạnh.
- Hạn chế: nếu chỉ dùng đặc trưng tổng hợp/đếm, mô hình dễ mất thông tin thứ tự (ordering) của chuỗi, trong khi nhiều hành vi tấn công thể hiện rõ qua mẫu trình tự (ví dụ: chuỗi thăm dò → truy cập dịch vụ → tải xuống).

1.8.3. Nhóm dựa trên tái tạo với Autoencoder, GRU-AE, Transformer-AE

Nhóm reconstruction-based giả định mô hình có thể học cấu trúc của dữ liệu bình thường để tái tạo lại đầu vào; khi gặp dữ liệu bất thường, sai số tái tạo tăng lên và được dùng làm điểm bất thường (anomaly score). Với dữ liệu chuỗi sự kiện, các biến thể thường gặp:

- GRU-AE / LSTM-AE (Seq2Seq Autoencoder): mã hoá chuỗi thành biểu diễn ẩn rồi giải mã để tái tạo chuỗi.
- Transformer-AE: dùng attention để học phụ thuộc xa trong chuỗi và tái tạo lại chuỗi/biểu diễn.

Ưu điểm: trực quan, có thể "giải thích" ở mức sai số theo bước thời gian, phù hợp khi mẫu bình thường ổn định.

Hạn chế quan trọng trong vận hành: phụ thuộc mạnh vào việc chọn ngưỡng (threshold) trên phân phối score, khi môi trường thay đổi theo ngày/ca hoặc khi dịch vụ thay đổi, phân phối score có thể drift khiến tăng cảnh báo giả nếu vẫn giữ ngưỡng tĩnh.

1.8.4. Nhóm dự đoán, xếp hạng theo chuỗi

Khác với tái tạo toàn chuỗi, nhóm dự đoán (prediction), xếp hạng (ranking) theo chuỗi xem chuỗi sự kiện như "ngôn ngữ rời rạc": từ ngữ cảnh trước đó, mô hình dự đoán sự kiện tiếp theo. Nếu sự kiện quan sát có xác suất thấp hoặc nằm ngoài top-k dự đoán, chuỗi có xu hướng bất thường. DeepLog là một hướng tiêu biểu theo cơ chế dự đoán chuỗi log [7].

Trong đồ án, hướng tiếp cận theo DabLog 1 chiều tập trung học biểu diễn chuỗi sự kiện (từ log key/event id) và đánh giá bất thường theo mức "khớp" của từng bước trong chuỗi. Cốt lõi của hướng này là tận dụng tính thứ tự và "mẫu chuyển tiếp" (transition patterns), điều đặc biệt phù hợp khi dữ liệu đã được gom theo subject và cửa

sổ thời gian. Một yêu cầu vận hành quan trọng của mô hình chuỗi rời rạc là quản lý codebook (từ điển ánh xạ log key → event id) và xử lý OOV/UNKNOWN để tránh sai lệch giữa giai đoạn huấn luyện và triển khai.

1.8.5. Nhóm Transformer, BERT-like và các hướng nâng cao cho log

Các mô hình BERT-like cho log (ví dụ LogBERT) thường dùng cơ chế attention và bài toán tiền huấn luyện kiểu "dự đoán token bị che" để học ngữ cảnh dài, phù hợp khi log đa dạng và có nhiều phụ thuộc xa. Tuy nhiên, chi phí huấn luyện/suy luận thường cao hơn và cần dữ liệu đủ lớn để ổn định. Trong phạm vi triển khai hệ thống phát hiện gần thời gian thực, các mô hình BERT-like thường được cân nhắc như một hướng mở rộng hoặc dùng cho tầng phân tích sâu (triage) hơn là tầng cảnh báo nhanh.

1.8.6. Tóm tắt so sánh và lý do lựa chọn hướng của đồ án

Tổng hợp các hướng trên cho thấy:

- OC-SVM/LOF là baseline hợp lý do đơn giản và hiệu quả trong nhiều thiết lập, nhưng dễ mất thông tin thứ tự nếu vector hoá thô.
- GRU-AE/Transformer-AE có khả năng học cấu trúc chuỗi theo tái tạo, nhưng nhạy với drift và phụ thuộc chọn ngưỡng.
- prediction/ranking (DeepLog, DabLog) khai thác trực tiếp tính trình tự và phù hợp với dữ liệu chuỗi sự kiện đã trừu tượng hoá thành log key. Do đó, đồ án lựa chọn hướng DabLog 1 chiều để cân bằng giữa năng lực học mẫu chuỗi và khả năng triển khai, đồng thời cần bổ sung baseline (OC-SVM/LOF) và các mô hình AE (GRU-AE/Transformer-AE) trong thực nghiệm để đối chiếu công bằng và làm rõ lợi thế/giới hạn của mô hình đề xuất.

Trong vận hành an ninh mạng, dữ liệu quan sát thường ở dạng log/flow có cấu trúc (ví dụ Zeek hoặc các flow đã chuẩn hoá), phù hợp để mô hình hoá trật tự sự kiện theo thời gian. Từ góc nhìn hành vi, nhiều pha của tấn công thể hiện thành chuỗi mẫu (sequence patterns): thăm dò → kết nối bất thường → thử đăng nhập/khai thác → tăng lưu lượng/di chuyển ngang. Vì vậy, các phương pháp khai thác mẫu chuyển tiếp giữa sự kiện (transition patterns) thường có lợi thế hơn các mô hình chỉ nhìn thống kê tổng hợp, đặc biệt khi mục tiêu là phát hiện sớm các diễn biến lệch chuẩn trong một cửa sổ thời gian ngắn.

Bảng 1.2 tổng hợp các hướng tiếp cận chính trong phát hiện bất thường từ log/chuỗi sự kiện theo 3 trục:

Bbiểu diễn đầu vào (token/chuỗi hay vector đặc trưng).

- Cơ chế chấm điểm (reconstruction/prediction/ranking/density).
- Điều kiện áp dụng (ổn định môi trường, drift, near-real-time).

Bảng cũng nêu lý do đồ án ưu tiên hướng prediction/ranking theo chuỗi (DabLog 1 chiều), đồng thời chỉ ra các baseline nên bổ sung để đối chiếu công bằng.

Bảng 1.2: So sánh các hướng phát hiện bất thường từ log/chuỗi sự kiện.

Ưu điểm trong vận hành	**Hạn chế chính**	**Điều kiện áp dụng phù hợp**
Nhanh, dễ triển khai, baseline tốt	Dễ mất thông tin thứ tự nếu vector hoá thô, nhạy chọn đặc trưng	Khi đã có feature mạnh; cần baseline "nhẹ" và ổn định
Trực quan; có thể xem "bước nào sai"	Phụ thuộc ngưỡng; drift làm sai phân phối score	Khi hành vi nền ổn định, có cơ chế hiệu chỉnh ngưỡng
Khai thác mẫu chuyển tiếp; phù hợp dữ liệu rời rạc theo chuỗi	Cần quản lý codebook/OOV, vẫn cần ngưỡng vận hành	Khi có pipeline token hoá ổn định; cần phát hiện theo chuỗi
Bắt phụ thuộc xa tốt; giàu ngữ cảnh	Chi phí huấn luyện cao, cần dữ liệu đủ lớn	Tầng phân tích sâu (triage), hoặc khi đủ tài nguyên/khối lượng dữ liệu
Mạnh khi đầu vào là chuỗi số (telemetry)	Cần thiết kế feature time-series phù hợp	Khi dữ liệu đã là time-series liên tục ổn định
Linh hoạt; dễ thêm "guardrail"	Thiết kế phức tạp; cần kiểm soát giao thoa	Khi cần hệ thống vận hành thực tế, có lớp an toàn

Nhóm phương pháp	Đại diện tiêu biểu	Biểu diễn đầu vào	Điểm bất thường (anomaly score)
Ngoại lai truyền thống (density/outlier)	Isolation Forest, OC-SVM, LOF	Vector đặc trưng theo subject/bucket (đếm log key, thống kê bytes/duration, entropy…)	Khoảng cách/vùng biên/density thấp
Reconstruction-based	AE/VAE, GRU-AE/LSTM-AE, Transformer-AE, USAD	Chuỗi/ma trận đặc trưng theo thời gian	Sai số tái tạo (reconstruction error)
Prediction/ranking theo chuỗi	DeepLog, DabLog	Chuỗi token (event id/log key) theo subject	Xác suất thấp / không nằm trong top-k / rank score cao
Transformer/BERT-like cho log	LogBERT và biến thể	Chuỗi token, ngữ cảnh dài	Loss/score theo masked modelling hoặc scoring head
Transformer cho time-series	TranAD (time-series AD)	Chuỗi đặc trưng liên tục theo thời gian	Sai số dự đoán/tái tạo theo attention
Hybrid/đa tầng	Rule + ML + DL	Kết hợp chuỗi + thống kê	Score tổng hợp/ensemble

Từ bảng 1.2, hướng prediction/ranking theo chuỗi phù hợp với dữ liệu đã trừu tượng thành log key/event id và được nhóm theo subject + time bucket như trong đồ án. Đồ án lựa chọn DabLog 1 chiều nhằm cân bằng giữa:

- Năng lực học mẫu chuyển tiếp của chuỗi sự kiện.
- Khả năng triển khai ổn định (mô hình gọn, suy luận nhanh, dễ đóng gói theo batch).

Tuy nhiên, để chứng minh lợi thế một cách thuyết phục, thực nghiệm cần bổ sung tối thiểu các baseline khác họ như Isolation Forest và OC-SVM/LOF trên cùng cách chia train/test và cùng quy tắc tạo subject.

CHƯƠNG 2: PHƯƠNG PHÁP ĐỀ XUẤT TRONG PHÁT HIỆN CÁC MỐI ĐE DOẠ AN NINH MẠNG DỰA VÀO HỌC SÂU

2.1 Phát biểu bài toán và giả định phát hiện bất thường theo chuỗi

Trong đồ án này, bài toán được đặt ra trong bối cảnh giám sát hoạt động mạng theo thời gian nhằm phát hiện sớm các hành vi bất thường có khả năng liên quan đến tấn công. Dữ liệu quan sát (từ log/flow) được mô hình hoá dưới dạng chuỗi sự kiện theo thứ tự thời gian. Cách phát biểu này phù hợp với mô hình phát hiện bất thường trên dữ liệu tuần tự: mỗi hoạt động mạng tại thời điểm t được xem như một sự kiện và một khoảng thời gian liên tiếp tạo thành một chuỗi sự kiện. Gọi e_t là sự kiện thứ t (được trích từ log Zeek sau tiền xử lý và mã hoá), khi đó toàn bộ hoạt động trong một giai đoạn dài được biểu diễn bởi:

$$E = \{e_1, e_2, \dots, e_n\} \tag{2.1}$$

Từ E, ta xây dựng các chuỗi con (sub-sequences) theo cửa sổ trượt độ dài L:

$$S_i = \{e_i, e_{i+1}, \dots, e_{i+L-1}\}, \qquad \mathrm{i} = 1, \dots, \mathrm{n} - \mathrm{L} + 1 \tag{2.2}$$

Mục tiêu của hệ thống là xác định các chuỗi con S_i tương ứng các khoảng thời gian/phiên/nhóm theo thực thể như IP) có hành vi bất thường so với phần còn lại của dữ liệu, đồng thời (khi cần) chỉ ra những sự kiện bất thường bên trong chuỗi để phục vụ phân tích và phản ứng. Để bài toán có thể giải được trong thực tế vận hành IDS, phương pháp phát hiện bất thường thường dựa trên hai giả thiết cơ bản:

- Các chuỗi con bất thường xuất hiện hiếm trong hành vi bình thường.
- Đặc trưng/hình thái của chuỗi bất thường khác biệt đáng kể so với chuỗi bình thường.

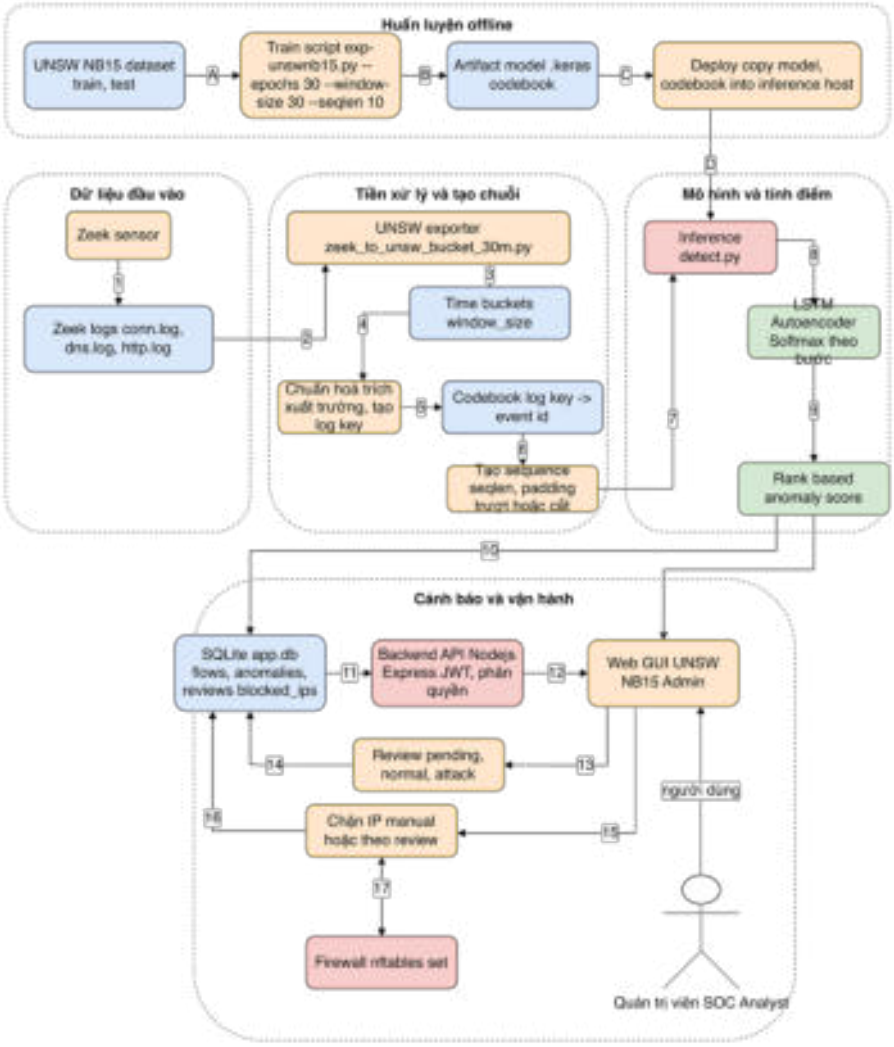
Từ đó, bài toán phát hiện bất thường trong đồ án được phát biểu ở dạng quyết định:

- Xây dựng một hàm chấm điểm/đánh giá mức độ bất thường $A(S_i)$ cho mỗi chuỗi S_i.
- Xác định nhãn:

$$y_i = \begin{cases} 1, & nếu\ A(S_i) \geq \tau\ (\text{bất thường}) \\ 0, & nếu\ A(S_i) < \tau\ (\text{bình thường}) \end{cases} \tag{2.3}$$

trong đó τ là ngưỡng cảnh báo được lựa chọn theo mục tiêu vận hành (giảm báo động nhầm hoặc giảm bỏ sót).

2.2 Mô hình đề xuất



Hình 2.1: Tổng quan quy trình phương pháp đề xuất.

Phương pháp đề xuất của đồ án được thiết kế theo hướng phát hiện bất thường dựa trên chuỗi sự kiện và phục vụ triển khai vận hành (có bước review và phản ứng chặn IP). Quy trình tổng quát gồm hai pha: pha học mô hình và pha phát hiện sinh cảnh báo.

A. Thu thập và chuẩn hoá dữ liệu đầu vào.

Ở hệ thống triển khai, nguồn dữ liệu chính là log Zeek (đặc biệt các log phản ánh kết nối và sự kiện giao thức), có cấu trúc rõ ràng và thuận lợi cho việc trích xuất trường thông tin. Zeek mô tả conn.log như log ghi lại các yếu tố lớp 3–4 "ai nói chuyện với ai, khi nào, trong bao lâu, dùng giao thức nào…", phù hợp để mô hình hoá hành vi theo thời gian. Ở pha huấn luyện/đánh giá trong môi trường nghiên cứu, có thể sử dụng bộ dữ liệu benchmark UNSW-NB15 (flow được gán nhãn) để kiểm chứng ý tưởng và tinh

chỉnh tham số. Trong triển khai, các trường thời gian/IP/port được chuẩn hoá định dạng và múi giờ, đồng thời loại bỏ/điền giá trị thiếu để đảm bảo pipeline tiền xử lý ổn định.

Hình 2.2: Minh hoạ nguồn dữ liệu conn.log từ Zeek.

B. Trừu tượng hoá bản ghi thành "sự kiện".

Từ mỗi bản ghi log/flow, hệ thống tạo ra một log key đại diện cho "loại sự kiện" (ví dụ dựa trên một số trường then chốt và/hoặc rời rạc hoá), nhằm giảm độ thưa và làm nổi bật cấu trúc hành vi. Các log key này sau đó được ánh xạ sang event ID thông qua một codebook (từ điển mã hoá), kèm các token đặc biệt như BEGIN, END, PAD, UNKNOWN để đảm bảo nhất quán giữa huấn luyện và suy luận. Việc trừu tượng hoá giúp giảm nhiễu do biến thiên giá trị liên tục (ví dụ duration/bytes) và làm tăng khả năng tổng quát hoá khi gặp hành vi mới. Trong thực tế, log key thường được xây dựng theo dạng "mẫu hành vi" của một sự kiện, ví dụ ghép các trường như protocol, service, state và một số thuộc tính rời rạc hoá để giữ lại thông tin phân biệt nhưng vẫn tránh bùng nổ số lượng giá trị. Việc cố định cấu hình tạo key (tập trường, cách rời rạc hoá) giúp event ID ổn định giữa các lần chạy và giảm nguy cơ lệch biểu diễn khi môi trường thay đổi nhẹ. Ngoài ra, cơ chế UNKNOWN đóng vai trò "van an toàn" để tiếp nhận các kiểu sự kiện hiếm hoặc mới mà không làm hỏng chuỗi đầu vào.

C. Tổ chức dữ liệu thành chuỗi theo cửa sổ thời gian và theo "đối tượng theo dõi".

Các sự kiện được nhóm theo một "subject" (thực thể theo dõi) và theo time bucket (ví dụ theo ngày–giờ–khoảng phút), tạo thành các chuỗi S_i như đã phát biểu ở 2.1. Cách nhóm này giúp hệ thống nhìn thấy trật tự sự kiện trong một khoảng hoạt động cụ thể (ví dụ theo IP trong một khung thời gian), thay vì chỉ nhìn các bản ghi rời rạc. Ngoài subject theo IP, hệ thống có thể mở rộng theo các subject khác (ví dụ 5-tuple hoặc UID của Zeek) tuỳ mục tiêu phát hiện và đặc thù mạng. Để hạn chế việc trộn lẫn các phiên hoạt động không liên quan, hệ thống có thể đặt thêm điều kiện cắt chuỗi khi khoảng cách thời gian giữa hai sự kiện vượt một ngưỡng (session gap), hoặc giới hạn số sự kiện tối đa trong một bucket. Khi số sự kiện vượt seqlen, áp dụng chiến lược cắt (truncate) theo "mới nhất" để ưu tiên tín hiệu gần thời điểm cảnh báo, hoặc dùng cửa sổ trượt để giữ

ngữ cảnh liên tục. Việc tổ chức theo subject và bucket cũng thuận lợi cho bước tổng hợp cảnh báo, vì có thể thống kê miss/score theo từng đối tượng trong từng khoảng thời gian cố định để so sánh theo ngày/giờ.

```
# define how to read sequences from file
def readSequences(ip, filename):
    sequence = {}
    label = {}
    with open(os.path.join(args.input, ip, filename), 'rt') as fin:
        csvfin = csv.reader(fin, delimiter=',')
        for line in csvfin:
            datetime = data.unswnb15.key.getDateTimeFromLine(line)
            srcip = line[data.unswnb15.key.srcip]
            dstip = line[data.unswnb15.key.dstip]
            dsport = line[data.unswnb15.key.dsport]
            subject = '-'.join(['from', srcip, 'to', dstip, 'on', str(datetime.day), str(datetime.hour),
                                str(datetime.minute // args.window_size)])
            slabel = data.unswnb15.key.getLabelFromLine(line)
            skeystr = data.unswnb15.key.getKeyFromLine(line, args.logkeys, args.key_divisor)
            if subject not in sequence:
                sequence[subject] = list()
            sequence[subject].append(skeystr)
            if subject not in label:
                label[subject] = list()
            label[subject].append(slabel)
    ret = []
    for subject in sequence:
        notNoneLabels = [l for l in label[subject] if l is not None]
        ret.append((sequence[subject], subject, ','.join(sorted(set(notNoneLabels))), len(notNoneLabels)))
    return ret
```

Hình 2.3: Hàm nhóm các sự kiện theo một "subject" theo windown_size.

D. Học mô hình chuỗi bằng LSTM Autoencoder.

Mô hình trung tâm là LSTM Autoencoder dạng encoder–decoder: encoder nén chuỗi sự kiện, decoder tái tạo lại chuỗi mục tiêu, hàm mất mát dùng categorical cross-entropy trên phân phối softmax theo từng bước thời gian. Cách học này nhằm mô hình hoá "mẫu chuỗi bình thường", từ đó nhận biết chuỗi lệch chuẩn ở pha suy luận. (Trong huấn luyện/triển khai, mô hình dùng nhiều tầng LSTM và có tuỳ chọn RepeatVector để tạo kiến trúc encoder–decoder.)

E. Chấm điểm bất thường và quyết định cảnh báo theo tiêu chí Top-N/rank.

Khi suy luận, mô hình xuất ra xác suất dự đoán cho từng bước thời gian. Một chuỗi được coi là bất thường nếu tồn tại bước mà nhãn/sự kiện thực tế không nằm trong nhóm dự đoán có xác suất cao (định lượng bằng rank chuẩn hoá so với kích thước từ vựng), vượt quá ngưỡng cấu hình. Đây là cách ra quyết định theo "xếp hạng" thường dùng trong phát hiện bất thường chuỗi log, tương thích với giả thiết "bất thường là hiếm và khác biệt đáng kể". Ngoài điều kiện rank, có thể bổ sung điều kiện về khoảng cách embedding để giảm nhầm lẫn khi mô hình dự đoán sai nhưng "gần nghĩa".

F. Tổng hợp kết quả thành bản ghi cảnh báo phục vụ vận hành.

Các "prediction-miss" được tổng hợp theo subject/time bucket để tạo chỉ số như: số miss, tổng số bước, tỷ lệ miss (%), các thống kê điểm (min/avg/max), và dải chỉ số

trong chuỗi… Sau đó cảnh báo được ghi vào CSDL (SQLite) và cung cấp qua API cho GUI, cho phép người dùng lọc theo ngày/trạng thái review, xem chi tiết, và thực hiện chặn IP (đúng định hướng IDPS: phát hiện + phản ứng).

2.3 Dữ liệu và nhãn

Trong đồ án này, dữ liệu được khai thác theo hai hướng song song: dữ liệu vận hành thực tế để triển khai hệ thống cảnh báo và dữ liệu chuẩn/benchmark để kiểm chứng mô hình trong môi trường có nhãn. Cách tổ chức này phù hợp với đặc thù bài toán an ninh mạng: dữ liệu thực tế thường thiếu "nhãn chuẩn", trong khi dữ liệu benchmark giúp đánh giá định lượng rõ ràng.

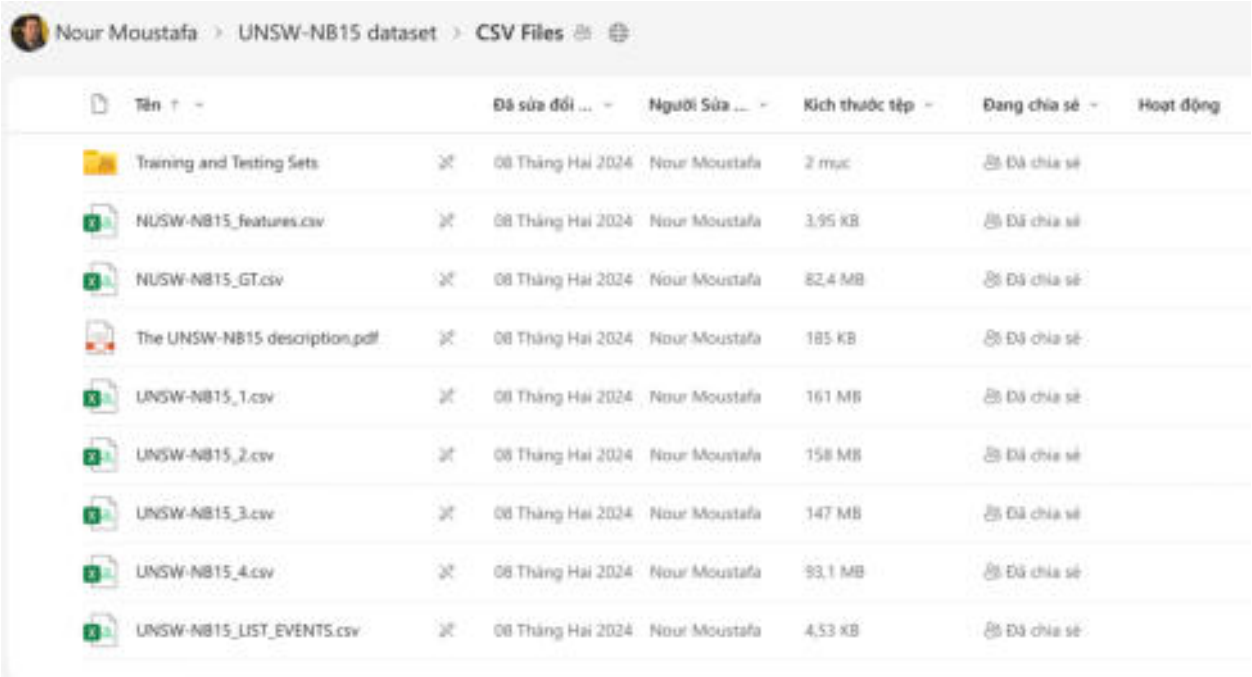
2.3.1. Nguồn dữ liệu vận hành từ Zeek

Ở mức triển khai, hệ thống thu thập log Zeek (network security monitor) và xem đây là nguồn dữ liệu đầu vào chính cho quá trình phát hiện. Zeek sinh ra nhiều loại log theo giao thức/sự kiện, trong đó conn.log phản ánh hoạt động lớp 3–4 ("ai kết nối tới ai, lúc nào, trong bao lâu, dùng giao thức nào…") và thường được dùng làm "xương sống" để mô hình hoá hành vi theo thời gian. Danh mục các log chuẩn mà Zeek có thể sinh ra (như conn.log, dns.log, http.log, ssl.log, files.log, …) được liệt kê rõ trong tài liệu chính thức [4], giúp quá trình lựa chọn nguồn trường dữ liệu ổn định và có thể mở rộng.

Trong hệ thống, các bản ghi sau xử lý được nhóm theo subject (ví dụ theo IP và time bucket) để tạo chuỗi sự kiện phục vụ suy luận. Kết quả phát hiện được lưu xuống CSDL và hiển thị trên giao diện dưới dạng bản ghi cảnh báo, ví dụ các trường ip, miss/total, miss_pct, thống kê score_*, và trạng thái xử lý review_status. (Phần cấu trúc bản ghi này sẽ được mô tả chi tiết ở Chương III khi trình bày thiết kế CSDL và luồng nghiệp vụ review/chặn IP.)

2.3.2. Dữ liệu benchmark có nhãn phục vụ huấn luyện và đánh giá

Tập dữ liệu UNSW-NB15 được công bố vào năm 2015 bởi Cyber Range Lab thuộc Australian Centre for Cyber Security (ACCS) [3]. Đây là một trong những bộ dữ liệu được sử dụng rộng rãi trong nghiên cứu phát hiện xâm nhập và phát hiện bất thường trong mạng máy tính, do kết hợp giữa lưu lượng mạng hợp lệ (benign traffic) và các hành vi tấn công được sinh tổng hợp bằng các công cụ tạo lưu lượng và kịch bản tấn công (ví dụ như IXIA PerfectStorm). Trong tập dữ liệu này, mỗi bản ghi tương ứng với một luồng mạng (network flow), đại diện cho một chuỗi các gói tin được quan sát giữa một địa chỉ IP nguồn và một địa chỉ IP đích trong một khoảng thời gian nhất định.



Hình 2.4: Bộ dữ liệu UNSW-NB15 [3].

Bảng 2.1: Giải thích chức năng của các tệp và thư mục trong bộ dữ liệu UNSW-NB15.

Tệp/Thư mục	Giải thích chức năng
Training and Testing Sets/ (thư mục)	Chứa 2 tập con đã chia sẵn để huấn luyện và kiểm thử mô hình dùng để chạy thí nghiệm ML nhanh mà không cần tự chia dữ liệu.
NUSW-NB15_features.csv	"Data dictionary" cho bộ dữ liệu: liệt kê các trường/đặc trưng (features) của UNSW-NB15 (tên feature, mô tả/nhóm/kiểu tuỳ phiên bản). Dùng để hiểu ý nghĩa từng cột, chọn feature, tiền xử lý, viết báo cáo.
NUSW-NB15_GT.csv	Ground truth table**:** bảng nhãn chuẩn/đối chiếu nhãn theo kịch bản sinh tấn công. Thường dùng khi cần xác thực nhãn, hoặc khi bạn xử lý lại dữ liệu từ mức sự kiện/kịch bản để gán nhãn cho bản ghi.
The UNSW-NB15 description.pdf	Tài liệu mô tả bộ dữ liệu: bối cảnh tạo dữ liệu, nhóm tấn công, cách sinh feature, ý nghĩa nhãn/định dạng. Dùng để trích dẫn và mô tả dataset trong chương dữ liệu/thiết kế thực nghiệm.
UNSW-NB15_1.csv	1 trong 4 file chứa toàn bộ bản ghi dữ liệu dạng CSV (các feature + nhãn), dùng làm dữ liệu gốc để huấn luyện/đánh giá khi muốn dùng full dataset.

UNSW-NB15_2.csv	Tương tự _1.csv: phần dữ liệu thứ 2 của tập tổng.
UNSW-NB15_3.csv	Tương tự _1.csv: phần dữ liệu thứ 3 của tập tổng.
UNSW-NB15_4.csv	Tương tự _1.csv: phần dữ liệu thứ 4 của tập tổng.

UNSW-NB15 bao gồm tổng cộng 2.540.044 luồng mạng, trong đó có 2.218.761 luồng bình thường (87,35%) và 321.283 luồng tấn công (12,65%). Các luồng tấn công được phân thành chín loại gồm: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode và Worms. Lưu lượng mạng được thu thập thông qua các công cụ bắt gói (ví dụ: tcpdump), sau đó được xử lý để trích xuất 49 đặc trưng ở mức luồng, được xây dựng dựa trên các công cụ phân tích lưu lượng như Argus và Bro/Zeek, kết hợp với các thuật toán sinh đặc trưng bổ sung. Để đảm bảo khả năng tái lập kết quả và thuận tiện cho việc so sánh với các nghiên cứu trước, nhiều công trình sử dụng bộ chia dữ liệu chuẩn do nhóm tác giả UNSW-NB15 công bố công khai, bao gồm 175.341 luồng cho tập huấn luyện và 82.332 luồng cho tập kiểm thử. Bảng 2.1 trình bày phân bố các lớp trong bộ chia này, cho thấy sự mất cân bằng lớp nghiêm trọng giữa các loại tấn công, đây là một thách thức quan trọng trong quá trình huấn luyện và đánh giá các mô hình phát hiện xâm nhập và bất thường.

Bảng 2.2: Phân bố lớp trong bộ chia huấn luyện/kiểm thử thường dùng của UNSW-NB15

Lớp	Số mẫu huấn luyện	Tỷ lệ (%)	Số mẫu kiểm thử	Tỷ lệ (%)
Normal	56.000	31,94	37.000	44,94
Analysis	2.000	1,14	677	0,82
Backdoors	1.746	1,00	583	0,71
DoS	12.264	6,99	4.089	4,97
Exploits	33.393	19,05	11.132	13,52
Fuzzers	18.184	10,37	6.062	7,36
Generic	40.000	22,81	18.871	22,92
Reconnaissance	10.491	5,98	3.496	4,25
Shellcode	1.133	0,65	378	0,46
Worms	130	0,07	44	0,05
Tổng:	175.341	100	82.332	100

Dưới đây là ý nghĩa các nhãn (label) này trong bộ dữ liệu an ninh mạng kiểu UNSW-NB15 (thường dùng để phân loại lưu lượng mạng thành "bình thường" và các nhóm tấn công):

- Normal: lưu lượng hợp lệ,bình thường như duyệt web, DNS, SSH quản trị hợp lệ, tải file hợp pháp, truy cập dịch vụ đúng cách.
- Analysis: nhóm hành vi "phân tích/thăm dò sâu" để hiểu hệ thống, ví dụ: port scan nâng cao, service enumeration, OS fingerprinting, kiểm tra banner/version để chuẩn bị khai thác.
- Backdoors: tấn công cài "cửa hậu" để quay lại sau này, ví dụ: mở một cổng bí mật/tiến trình ẩn, tạo tài khoản lạ, cài trojan để có thể truy cập trái phép về sau.
- DoS (Denial of Service): làm dịch vụ quá tải hoặc không phục vụ được người dùng thật. Ví dụ: gửi cực nhiều request/packet (SYN flood, UDP flood, HTTP flood) khiến web/app bị treo hoặc chậm.
- Exploits: khai thác lỗ hổng để chiếm quyền hoặc thực thi lệnh. Ví dụ: khai thác lỗi buffer overflow, SQL injection, RCE, khai thác CVE của dịch vụ để chạy mã trên máy nạn nhân.
- Fuzzers: "Fuzzing" = bắn dữ liệu ngẫu nhiên/biến dị vào dịch vụ để làm nó lỗi/rụng, hoặc tìm lỗ hổng. Ví dụ: gửi chuỗi input kỳ lạ/độ dài bất thường vào server để gây crash hoặc lộ lỗi xử lí.
- Generic: nhóm tấn công "chung chung" (thường liên quan mật mã) mà không phụ thuộc nhiều vào ứng dụng cụ thể. Ví dụ: tấn công vào thuật toán/handshake (như brute-force/đụng độ/khai thác điểm yếu mật mã), hoặc hành vi có mẫu chung khó gắn vào một dịch vụ cụ thể.
- Reconnaissance: trinh sát/thu thập thông tin trước khi tấn công. Ví dụ: ping sweep, scan dải IP, dò cổng, tìm host sống, tra cứu DNS, kiểm tra service/đường dẫn phổ biến.
- Shellcode: mã nhỏ dùng để mở shell hoặc thực thi lệnh sau khi khai thác thành công. Ví dụ: payload tạo reverse shell, bind shell, hoặc đoạn mã chạy lệnh hệ thống sau khi vượt qua bảo vệ.
- Worms: sâu máy tính tự lây lan qua mạng. Ví dụ: malware tự quét mạng để tìm máy dễ bị khai thác rồi tự nhân bản sang máy khác (lan nhanh, không cần người dùng bấm).

Bộ dữ liệu UNSW-NB15 là một trong những tập dữ liệu chuẩn được sử dụng rộng rãi trong nghiên cứu phát hiện xâm nhập và phát hiện bất thường ở tầng mạng. Trong đồ án này, bộ dữ liệu được sử dụng để mô hình hóa hành vi lưu lượng mạng, trong

đó mỗi bản ghi đại diện cho một kết nối hoặc luồng mạng được mô tả bằng tập các thuộc tính đo lường được. Mỗi bản ghi trong UNSW-NB15 phản ánh một tương tác mạng cụ thể, bao gồm thông tin kết nối cơ bản như địa chỉ nguồn, địa chỉ đích, cổng, giao thức và thời lượng, cùng với các thống kê lưu lượng như số lượng gói tin, số byte truyền, tốc độ và các chỉ số liên quan đến hướng truyền. Ngoài ra, bộ dữ liệu còn cung cấp các đặc trưng hành vi được thiết kế để phản ánh các mẫu đáng ngờ, chẳng hạn như hành vi quét cổng, tần suất kết nối bất thường hoặc sự mất cân đối trong luồng truyền dữ liệu.

Hình 2.5: Cấu trúc các dòng log trong bộ dữ liệu UNSW-NB15.

Bộ dữ liệu này đặc biệt phù hợp với tầng mạng do nó chứa sự pha trộn giữa hành vi hợp lệ và hành vi tấn công, đồng thời các đặc trưng chủ yếu ở dạng có cấu trúc. Điều này cho phép áp dụng các phương pháp học máy truyền thống, các mô hình học sâu xử lý vector đặc trưng, cũng như các pipeline kết hợp tiền xử lý đặc trưng và mô hình hóa theo chuỗi khi các bản ghi được gom nhóm theo cửa sổ thời gian hoặc phiên giao tiếp. Trong đồ án, các bản ghi mạng được xử lý thông qua bước tiền xử lý để chuyển đổi thành dạng đầu vào phù hợp cho mô hình. Các thuộc tính dạng phân loại được mã hóa, trong khi các thuộc tính số được chuẩn hóa hoặc tiêu chuẩn hóa.

Bảng 2.3: Thống kê số lượng mẫu trong các tệp dữ liệu UNSW-NB15 được sử dụng.

Tệp dữ liệu	Tổng số luồng	Luồng bình thường	Luồng tấn công	Tỷ lệ tấn công (%)
UNSW-NB15_1.csv	700.000	677.785	22.215	3,17
UNSW-NB15_2.csv	700.000	647.251	52.749	7,54
UNSW-NB15_3.csv	700.000	542.575	157.425	22,49
UNSW-NB15_4.csv	440.044	351.150	88.894	12,70

Tổng cộng:	2.540.044	2.218.761	321.283	12,65

2.3.3. Khái niệm nhãn trong đồ án và cách sử dụng

Trong đồ án, cần làm rõ "nhãn" theo hai cấp độ:

A. Nhãn chuẩn trong dữ liệu benchmark.

Với UNSW-NB15, nhãn thường bao gồm: nhãn nhị phân (normal/attack) và nhãn đa lớp theo nhóm tấn công. Nhãn này được sử dụng để:

- Đánh giá định lượng các chỉ số như Precision/Recall/F1, ROC–AUC/PR–AUC.
- So sánh tham số và lựa chọn cấu hình mô hình phù hợp trước khi đưa vào triển khai.

B. Nhãn vận hành (thực tế) trong hệ thống triển khai.

Với log Zeek trong môi trường thật, nhãn đúng không sẵn có. Vì vậy, đồ án sử dụng cơ chế phát hiện bất thường để sinh cảnh báo, sau đó dựa vào quy trình review của người vận hành để tạo ra nhãn vận hành theo thời gian. Cụ thể:

- eview_status (ví dụ pending, và các trạng thái sau khi duyệt) được xem như nhãn phản hồi của chuyên viên/ quản trị.
- review_by, review_at là siêu dữ liệu phục vụ truy vết và đánh giá chất lượng quy trình vận hành.

Cách thiết kế này có hai lợi ích: hỗ trợ ra quyết định thực tế (lọc, ưu tiên, chặn IP) và tạo ra tập dữ liệu phản hồi để cải thiện mô hình về sau (theo hướng human-in-the-loop hoặc bán giám sát).

2.4 Tiền xử lý và tạo chuỗi

Quy trình phát hiện bất thường theo chuỗi phụ thuộc mạnh vào cách biến đổi log thô thành chuỗi sự kiện rời rạc có ý nghĩa. Với nguồn log Zeek, lợi thế là dữ liệu đã có cấu trúc và schema rõ ràng, thuận lợi cho chuẩn hoá và tạo "sự kiện" một cách nhất quán.

2.4.1. Lựa chọn nguồn log và trường thông tin

Bước đầu tiên là chọn các log phản ánh hành vi mạng cốt lõi và các trường có giá trị mô tả tiến trình hoạt động. Thông thường:

- conn.log cung cấp bức tranh kết nối nền tảng (thời điểm, định danh kết nối, hai đầu giao tiếp, giao thức, thời lượng, bytes, trạng thái kết nối…), phù hợp để mô hình hoá hành vi theo thời gian.

- Khi cần tăng ngữ cảnh, có thể bổ sung các log giao thức như DNS/HTTP/SSL tuỳ mục tiêu giám sát, Zeek liệt kê danh mục log và mô tả trường theo schema để tham chiếu thống nhất.

Nguyên tắc lựa chọn trường là ưu tiên tín hiệu hành vi ổn định (proto/service/conn_state, nhóm kết quả giao thức, hướng byte…) thay vì các trường có độ biến thiên quá lớn gây bùng nổ số lượng giá trị (ví dụ UID duy nhất cho từng kết nối).

2.4.2. Làm sạch và chuẩn hoá dữ liệu

Tiền xử lý nhằm đảm bảo dữ liệu đầu vào "sạch" và có thể so sánh theo thời gian:

- Chuẩn hoá thời gian: chuyển ts về định dạng thống nhất và đảm bảo sắp xếp đúng thứ tự thời gian.
- Xử lý thiếu dữ liệu: Zeek có cơ chế trường rỗng/không tồn tại tuỳ định dạng TSV/JSON và schema, cần quy ước giá trị mặc định (NA/0/UNKNOWN) nhất quán.
- Chuẩn hoá kiểu dữ liệu: số (duration/bytes) về dạng số, trường phân loại (proto/service/conn_state) về chuỗi chuẩn (lowercase/trim) để tránh "trùng nghĩa khác dạng".
- Giảm nhiễu và cực trị: có thể rời rạc hoá (binning) một số trường liên tục (duration, bytes) theo các khoảng, nhằm giảm nhạy cảm với dao động nhỏ nhưng vẫn giữ được tín hiệu "tăng/giảm bất thường".

2.4.3. Tạo log key từ bản ghi

Mục tiêu của log key (event key) là ánh xạ một bản ghi log/flow thành một token rời rạc đại diện cho "loại sự kiện", giúp chuỗi sự kiện có không gian rời rạc ổn định và tránh bùng nổ số lượng giá trị do các trường liên tục (bytes, duration) hoặc định danh duy nhất (UID). Với log có cấu trúc như Zeek hoặc dữ liệu flow đã chuẩn hoá, log key được xây dựng bằng cách kết hợp nhóm trường phân loại có ý nghĩa hành vi (proto/service/state/…) và một số trường số sau khi rời rạc hoá (binning) theo khoảng hoặc theo phép chia nguyên (key_divisor) để giảm nhạy cảm với dao động nhỏ.

Về mặt hình thức, với một bản ghi r, log key được tạo theo dạng chuỗi ghép có phân tách rõ ràng:

$$skey(r) = proto \mid service \mid state \mid http\_mthd \mid ftp\_login \mid pkt\_bin \mid \cdots \quad (2.4)$$

trong đó các thành phần pkt_bin , byte_bin là giá trị đã rời rạc hoá (ví dụ chia nguyên theo key_divisor, hoặc phân bin theo quantile). Nguyên tắc chung là giữ tín hiệu hành

vi ổn định (giao thức, dịch vụ, trạng thái kết nối, hành vi HTTP/FTP ở mức thô) và loại bỏ/giảm độ phân giải các trường biến thiên mạnh.

Bảng 2.4: Trường sử dụng để tạo log key và quy tắc chuẩn hoá/rời rạc hoá.

Nhóm trường	**Trường**	**Xử lý trước khi ghép vào key**	**Lý do chọn**
Phân loại cốt lõi	proto, service	lowercase/trim; nếu rỗng → unknown_service	Phản ánh loại giao tiếp và dịch vụ
Trạng thái phiên	conn_state/state	chuẩn hoá về tập giá trị hữu hạn; nếu rỗng → state_unk	Tín hiệu mạnh để phân biệt phiên bất thường
HTTP (nếu có)	phương thức/độ sâu giao dịch	phương thức đưa về nhãn rời rạc; trans_depth có thể bin theo {0,1,2+}	Tăng ngữ cảnh ứng dụng web
FTP (nếu có)	is_ftp_login, ct_ftp_cmd	đưa về nhãn {0/1} hoặc bin lệnh	Bắt mẫu đăng nhập/truyền tệp bất thường
Cường độ lưu lượng	tổng gói/tổng bytes (ví dụ spkts+dpkts, sbytes+dbytes)	rời rạc hoá theo key_divisor (chia nguyên) hoặc theo quantile	Giữ tín hiệu "mức độ" ở độ phân giải thô
Trường loại bỏ	UID, timestamp thô, port nguồn thay đổi nhanh	không đưa vào key	Tránh tạo key quá thưa, khó tổng quát

Để đảm bảo tính tái lập, cấu hình tạo key được đóng băng theo tham số (ví dụ logkeys và key_divisor) và chỉ thay đổi có kiểm soát giữa các phiên bản. Ví dụ, với key_divisor = 100, đặc trưng tổng gói được rời rạc hoá theo dạng:

$$\text{pkt\_}bin = \frac{spkts + dpkts}{100} \tag{2.5}$$

nhờ vậy token ổn định hơn trước biến thiên nhỏ và giúp mô hình tập trung vào cấu trúc chuỗi thay vì nhiễu.

```
# tạo logkey cho mô hình học.
def getKeyFromLine(line, keyset=0, divisor=100):
    # correction
    line[is_ftp_login] = str(int(line[is_ftp_login]) & 1) if len(line[is_ftp_login].strip()) else '0'
    line[ct_ftp_cmd] = str(int(line[ct_ftp_cmd])) if len(line[ct_ftp_cmd].strip()) else '0'
    line[ct_flw_http_mthd] = str(int(line[ct_flw_http_mthd])) if len(line[ct_flw_http_mthd].strip()) else '0'
    line[trans_depth] = str(int(line[trans_depth])) if len(line[trans_depth].strip()) else '0'
    # add-on
    pktsum = str((int(line[spkts]) + int(line[dpkts])) // divisor)
    pktsum_idx = len(line)
    line.append(pktsum)
    # key
    keys = [
        # 706/366 keys when divisor is 100/1000
        [proto, state, service, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, trans_depth, is_sm_ips_ports, pktsum_idx],
        [proto, state, service, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, trans_depth, is_sm_ips_ports],  # 284 keys
        [proto, state, service, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, trans_depth],  # 280 keys
        [proto, state, service, is_ftp_login, ct_ftp_cmd],  # 206 keys
        [proto, state, service, is_ftp_login],  # 182 keys
    ]
    keystr = ','.join([line[k] for k in keys[keyset]])
    # print('sbyte:', line[sbytes], ', dbytes:', line[dbytes], end=' , ')
    # print('spkts:', line[spkts], ', dpkts:', line[dpkts], end=' , ')
    # print('smeansz:', line[smeansz], ', dmeansz:', line[dmeansz])
    # print('sum/' + str(divisor) + ':', str(pktsum))
    return keystr
```

Hình 2.6: Đoạn mã tạo log key.

2.4.4. Xây dựng codebook và mã hoá sự kiện

Sau khi tạo log key, hệ thống ánh xạ mỗi log key sang một event id thông qua codebook (từ điển mã hoá). Codebook được xây dựng chỉ từ tập huấn luyện để tránh rò rỉ thông tin, sau đó được đóng băng khi đưa vào vận hành. Trong triển khai thực tế, codebook cần có cơ chế xử lý sự kiện mới/hiếm thông qua token <UNKNOWN> nhằm đảm bảo suy luận không bị lỗi khi môi trường thay đổi (dịch vụ mới, hành vi mới, log mới).

Bảng 2.5: Token đặc biệt và ý nghĩa.

Token đặc biệt	Ý nghĩa	Khi xuất hiện
<PAD>	Đệm cho chuỗi ngắn hơn seqlen	Khi đóng gói batch/cửa sổ
<SEQUENCE>	Đánh dấu đầu chuỗi	(Tuỳ chọn) nếu dùng marker
</SEQUENCE>	Đánh dấu cuối chuỗi	(Tuỳ chọn) nếu dùng marker
<UNKNOWN>	Sự kiện ngoài từ vựng (OOV) hoặc quá hiếm	Khi log key không nằm trong codebook

Để giảm nhiễu do các sự kiện cực hiếm, codebook áp dụng ngưỡng tần suất tối thiểu min_freq: các log key có tần suất nhỏ hơn min_freq trong tập huấn luyện sẽ được gộp về <UNKNOWN>. Cách làm này giúp giảm kích thước từ vựng, tăng ổn định embedding và hạn chế việc mô hình "học vẹt" các token hiếm.

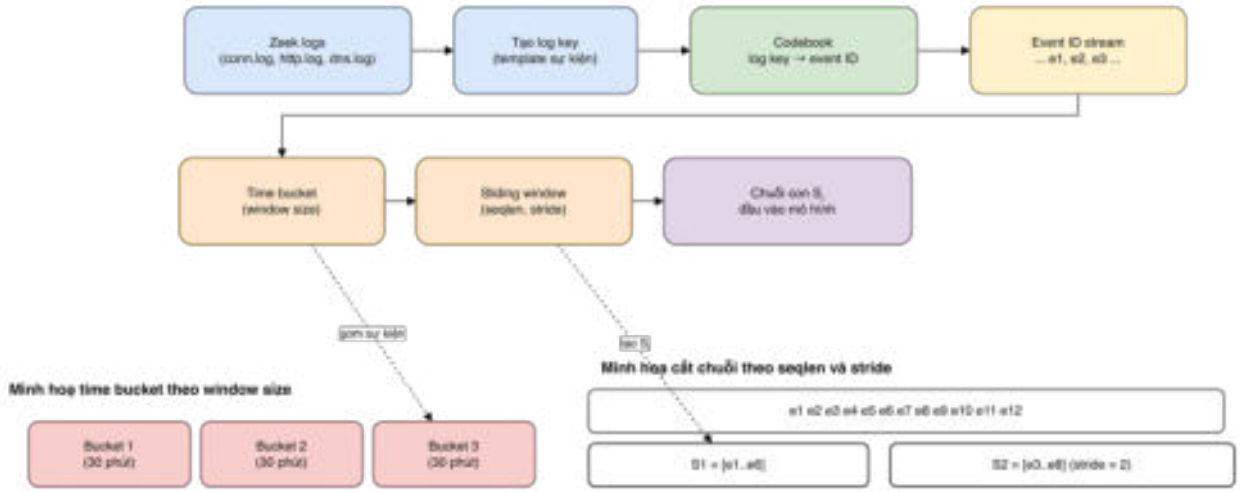
2.4.5. Tạo chuỗi theo time bucket và cửa sổ trượt

Sau khi mỗi bản ghi được mã hoá thành event ID, dữ liệu được tổ chức thành chuỗi theo hai quyết định thiết kế:

- Quy tắc nhóm (grouping) theo thực thể theo dõi: Ví dụ theo IP nguồn, theo cặp IP–dịch vụ, hoặc theo một "subject" đã chuẩn hoá. Quy tắc nhóm cần nhất quán để chuỗi phản ánh hành vi liên tục của cùng một đối tượng.
- Quy tắc chia theo thời gian (time-bucket) và độ dài chuỗi L: Trong mỗi bucket, sắp xếp sự kiện theo thời gian và tạo chuỗi con độ dài cố định L bằng cửa sổ trượt:

$$S_i = \{e_i, e_{i+1}, \dots, e_{i+L-1}\} \tag{2.6}$$

Nếu chuỗi ngắn hơn L thì đệm PAD, nếu dài hơn thì cắt hoặc trượt để sinh nhiều mẫu. Cách chia log theo thư mục/ngày hoặc xoay vòng log theo giờ là đặc tính thường gặp khi vận hành Zeek, thuận lợi cho việc gom dữ liệu theo ngày/bucket và tái hiện lại hành vi theo thời gian.



Hình 2.7: Sơ đồ luồng hoạt động từ log Zeek đến sliding window tạo chuỗi S_i.

2.4.6. Tách tập dữ liệu và tránh rò rỉ theo thời gian

Với dữ liệu chuỗi, cần hạn chế việc trộn ngẫu nhiên nếu mục tiêu là đánh giá gần với vận hành. Thực hành phù hợp là:

- Tách train/validation/test theo thời gian (ví dụ train ở giai đoạn trước, test ở giai đoạn sau) để giảm rò rỉ mẫu tương tự.
- Với dữ liệu theo subject, hạn chế để cùng một đoạn hành vi "rất gần nhau" xuất hiện đồng thời ở train và test.

2.4.7. Định dạng đầu ra của khâu tiền xử lý

Kết quả của mục 2.4 thường gồm:

- Tập chuỗi event ID (đã pad/cắt) và (tuỳ thiết kế) chuỗi mục tiêu cho bài toán tái tạo/dự đoán bước tiếp theo.
- Codebook + cấu hình tiền xử lý (binning, L L, bucket size, quy tắc grouping) để đảm bảo tái lập khi triển khai.

2.5 Xây dựng từ điển mã hoá và cơ chế embedding

2.5.1. Vai trò của codebook trong biểu diễn chuỗi sự kiện

Sau khi mỗi bản ghi log được trừu tượng hoá thành log key (loại sự kiện), cần có một cơ chế ánh xạ thống nhất từ log key sang chỉ số nguyên (event ID). Từ điển ánh xạ này được gọi là codebook. Codebook đảm bảo:

- Cùng một log key luôn được mã hoá thành cùng một event ID.
- Dữ liệu ở các giai đoạn huấn luyện/đánh giá/vận hành có thể được mã hoá nhất quán.
- Hỗ trợ kiểm soát sự kiện "mới" thông qua quy ước OOV (out-of-vocabulary).

2.5.2. Thiết kế tập token đặc biệt và quy ước mã hoá

Trong dữ liệu chuỗi, độ dài chuỗi thực tế thay đổi theo thời gian và theo đối tượng theo dõi, vì vậy cần các token đặc biệt để chuẩn hoá đầu vào mô hình. Thực hành phổ biến là bổ sung tối thiểu các token sau:

- <PAD>: dùng để đệm chuỗi ngắn hơn độ dài chuẩn *L*, giúp tạo tensor đồng kích thước khi huấn luyện/suy luận.
- <UNKNOWN>: đại diện cho sự kiện chưa từng xuất hiện trong codebook, giúp hệ thống không "vỡ" khi gặp log mới.
- <SEQUENCE> / </SEQUENCE>: đánh dấu bắt đầu/kết thúc chuỗi, thuận lợi khi cần mô hình hoá ranh giới chuỗi rõ ràng.

Quy ước token đặc biệt và việc dùng codebook nhằm đảm bảo tính ổn định của pipeline biểu diễn chuỗi và đã được mô tả như một phần của quá trình chuẩn hoá dữ liệu đầu vào trong tài liệu huấn luyện. Các token này cũng hỗ trợ cơ chế masking và giúp mô hình phân biệt rõ "dữ liệu thật" với "vùng đệm", từ đó cải thiện tính nhất quán và độ tin cậy của suy luận.

```python
class Codebook(object):
    """
        Function: to construct a sequence with add-ons: begin-sequence, end-sequence, unknown
    """
    PAD = '<pad>'
    BEGIN = '<sequence>'
    END = '</sequence>'
    UNKNOWN = '<unknown>'

    def __init__(self, codebook=None, threshold=0.01):
        self.encode, self.decode, self.types = None, None, None
        if codebook is not None:
            if isinstance(codebook, dict):
                codebook = set(codebook.keys())
            if isinstance(codebook, list):
                codebook = set(codebook)
            if isinstance(codebook, set):
                for preserved in [self.PAD, self.BEGIN, self.END, self.UNKNOWN]:
                    if preserved in codebook:
                        codebook.remove(preserved)
                self.types = set(codebook)
                self.decode = {i: k for i, k in
                               enumerate([self.PAD, self.BEGIN, self.END, self.UNKNOWN] + sorted(self.types))}
                self.encode = {k: i for i, k in self.decode.items()}
                self.types = codebook | {self.PAD, self.BEGIN, self.END, self.UNKNOWN}
```

Hình 2.8: Thêm các token đặt biệt vào Codebook.

2.5.3. Quy trình xây dựng codebook

Codebook được tạo từ tập log key xuất hiện trong dữ liệu huấn luyện (hoặc giai đoạn "fit" ban đầu) theo quy trình chuẩn:

- Thu thập tập log key sau tiền xử lý và trừu tượng hoá.
- Đếm tần suất và (nếu cần) loại bỏ các log key quá hiếm để giảm nhiễu và kích thước từ vựng.
- Gán chỉ số cho từng log key theo một thứ tự xác định (ví dụ theo tần suất giảm dần hoặc theo thứ tự từ điển) và chèn các token đặc biệt vào các chỉ số đầu tiên để cố định.
- Đóng băng codebook và lưu kèm metadata (kích thước từ vựng, tham số tiền xử lý, phiên bản).

Điểm quan trọng của bước này là tính tái lập: cùng một codebook phải được dùng lại khi mã hoá dữ liệu ở giai đoạn vận hành để tránh sai lệch nghĩa của event ID giữa các lần chạy.

2.5.4. Quản lý OOV và chiến lược cập nhật codebook

Trong vận hành, log key mới có thể xuất hiện do thay đổi dịch vụ, cấu hình, hoặc hành vi người dùng. Khi đó:

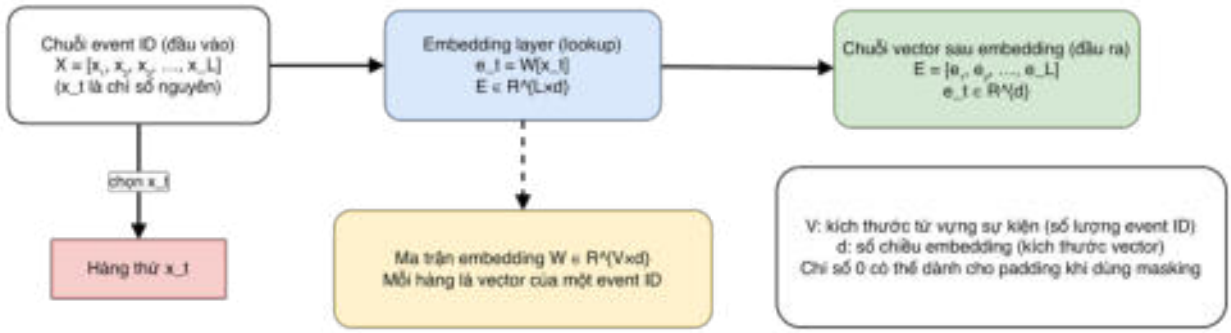
- Nếu log key không có trong codebook, ánh xạ về UNKNOWN để vẫn tạo được chuỗi hợp lệ.
- Việc cập nhật codebook cần được kiểm soát theo phiên bản: khi thay đổi codebook, mô hình và codebook phải được cập nhật đồng bộ, nếu chỉ cập nhật codebook mà không cập nhật mô hình, embedding sẽ không còn tương thích về ý nghĩa và kích thước từ vựng.

2.5.5. Cơ chế Embedding

Sau khi sự kiện đã được mã hoá thành event ID, cần chuyển đổi sang dạng vector số thực để mô hình học sâu xử lý hiệu quả. Embedding thực hiện phép ánh xạ:

$$event\_id \in \{0, \dots, |V| - 1\} \longrightarrow v \in \mathbb{R}^d \quad (2.7)$$

trong đó | V | là kích thước từ vựng (số event ID) và d là số chiều embedding. Embedding được học như một ma trận tham số $E \in \mathbb{R}^{|v| \times d}$, mỗi hàng là vector biểu diễn cho một event ID, khi gặp event ID, mô hình thực hiện phép "tra cứu" (lookup) để lấy ra vector tương ứng. Cơ chế này giúp giảm biểu diễn thưa kiểu one-hot và cho phép mô hình học quan hệ gần nhau giữa các loại sự kiện có hành vi tương tự. Lớp Embedding trong Keras/TensorFlow là hiện thực phổ biến của cơ chế này với các tham số input_dim và output_dim.



Hình 2.9: Ma trận embedding và phép lookup event ID → vector.

2.5.6. Liên hệ giữa codebook và embedding trong triển khai mô hình

Giữa codebook và embedding tồn tại ràng buộc chặt chẽ:

- Kích thước embedding phụ thuộc | V | (kích thước codebook).
- Ý nghĩa của mỗi hàng trong ma trận embedding phụ thuộc đúng thứ tự gán chỉ số trong codebook.

Do đó, khi lưu mô hình để triển khai, cần lưu kèm codebook (và metadata tiền xử lý) như một phần bắt buộc để đảm bảo suy luận nhất quán theo thời gian.

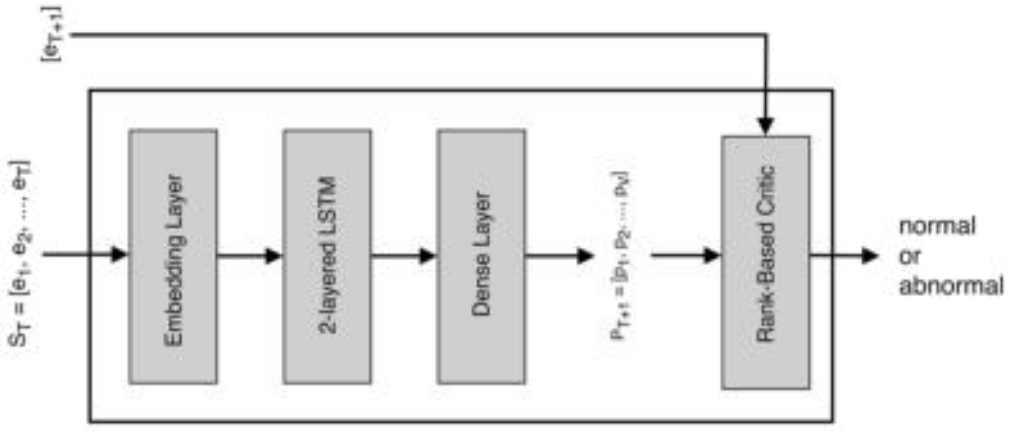
2.6 Kiến trúc mô hình học sâu đề xuất

2.6.1 Giới thiệu mô hình DeepLog và DABLog

Trong những năm gần đây, phát hiện bất thường dựa trên log hệ thống (log-based anomaly detection) đã trở thành một hướng nghiên cứu quan trọng trong lĩnh vực an ninh mạng và quản trị hệ thống. Thay vì phụ thuộc vào các luật thủ công do con người xây dựng, các phương pháp học sâu cho phép mô hình hóa hành vi bình thường của hệ thống một cách tự động và phát hiện các hành vi bất thường dựa trên sự sai lệch so với hành vi đã học. Trong số các phương pháp tiêu biểu, DeepLog và DABLog là hai mô hình đại diện cho hướng tiếp cận học chuỗi log dựa trên deep learning.

DeepLog là một mô hình phát hiện bất thường dựa trên log, sử dụng mạng nơ-ron hồi tiếp LSTM để học các phụ thuộc theo thời gian của chuỗi sự kiện log. Ý tưởng cốt lõi của DeepLog là giả định rằng trong điều kiện vận hành bình thường, các sự kiện log sẽ xuất hiện theo những chuỗi có tính quy luật. Mô hình được huấn luyện để dự đoán sự kiện log tiếp theo dựa trên các sự kiện trước đó; nếu sự kiện quan sát được không nằm trong tập các dự đoán có xác suất cao, hệ thống sẽ coi đây là một dấu hiệu bất thường.

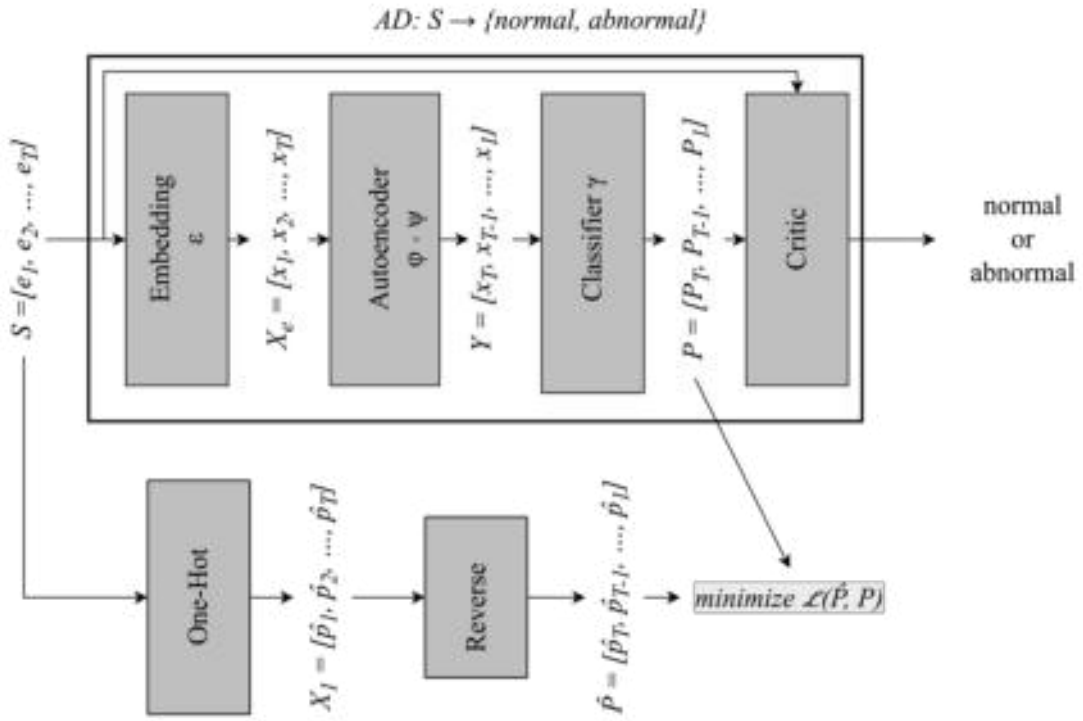
DABLog (Deep Attention-Based Log anomaly detection) được đề xuất như một mô hình mở rộng của DeepLog nhằm khắc phục hạn chế khi xử lý các chuỗi log dài và phức tạp. Thay vì sử dụng cơ chế attention tường minh như trong các mô hình NLP truyền thống, DABLog tích hợp cơ chế attention một cách gián tiếp (implicit attention) thông qua quá trình học biểu diễn có hướng dẫn bởi critic. Cách tiếp cận này cho phép mô hình tập trung nhiều hơn vào các sự kiện log có ảnh hưởng lớn đến quyết định phát hiện bất thường, từ đó nâng cao hiệu quả phát hiện trong các hệ thống log quy mô lớn.



Hình 2.10: Kiến trúc mô hình DeepLog.

Hình 2.10 minh họa kiến trúc tổng quát của mô hình DeepLog. Các log thô trước hết được tiền xử lý và chuyển đổi thành các log template, sau đó được ánh xạ thành các chỉ số (ID). Chuỗi các ID này được đưa vào mạng LSTM nhiều tầng để học mối quan hệ theo chuỗi giữa các sự kiện log. Đầu ra của LSTM được đưa qua lớp dense để sinh ra phân bố xác suất của sự kiện log tiếp theo. Một cơ chế đánh giá dựa trên xếp hạng (rank-based critic) được sử dụng để xác định liệu sự kiện quan sát được có thuộc nhóm dự đoán hợp lệ hay không; nếu không, sự kiện đó sẽ được đánh dấu là bất thường.

Kiến trúc mô hình học sâu đề xuất được xây dựng theo hướng của DabLog, một phương pháp phát hiện bất thường cho chuỗi sự kiện rời rạc bằng cách kết hợp autoencoder theo thời gian và bộ phân loại sự kiện theo từng bước. Hình 2.11 trình bày kiến trúc của mô hình DABLog. Sau bước embedding, chuỗi log được đưa vào một autoencoder để học biểu diễn tiềm ẩn của chuỗi sự kiện. Biểu diễn này sau đó được sử dụng cho nhiệm vụ phân loại và đánh giá bất thường thông qua classifier và critic. Không giống như DeepLog, DABLog sử dụng một cơ chế phản hồi ngược (reverse/feedback loop) từ critic để điều chỉnh quá trình học biểu diễn. Thông qua cơ chế này, các sự kiện log quan trọng sẽ có ảnh hưởng lớn hơn đến hàm mất mát và quá trình cập nhật tham số, từ đó đóng vai trò tương đương với một cơ chế attention gián tiếp. Cách tiếp cận này giúp mô hình tập trung vào các sự kiện log mang tính quyết định mà không cần một lớp attention tường minh.



Hình 2.11: Tổng quan về kiến trúc của DabLog.

Bảng 2.6: So sánh mô hình DeepLog và DABLog.

Tiêu chí	DeepLog	DABLog
Kiến trúc chính	LSTM + Rank-based critic	Autoencoder + Classifier + Critic
Cơ chế attention	Không có	Attention gián tiếp (critic-guided)
Cách học chuỗi log	Học tuần tự, trọng số đồng đều	Tái trọng số sự kiện log quan trọng
Khả năng xử lý chuỗi dài	Tốt	Rất tốt
Độ chính xác phát hiện	Cao	Cao hơn
Tỷ lệ false positive	Tương đối cao trong chuỗi dài	Thấp hơn
Khả năng giải thích	Thấp	Cao
Độ phức tạp mô hình	Trung bình	Cao
Thời gian huấn luyện	Lâu	Lâu hơn
Ứng dụng phù hợp	Hệ thống log quy mô vừa	Hệ thống log lớn, phức tạp

Từ bảng so sánh có thể thấy rằng DeepLog là một mô hình nền tảng hiệu quả trong phát hiện bất thường dựa trên log, đặc biệt phù hợp với các hệ thống có chuỗi log không quá dài và cấu trúc tương đối ổn định. Tuy nhiên, do các sự kiện log trong chuỗi được xử lý với mức độ quan trọng tương đương nhau, mô hình có thể gặp hạn chế khi áp dụng cho các hệ thống phức tạp hoặc chuỗi log dài.

DABLog khắc phục hạn chế này bằng cách sử dụng cơ chế attention gián tiếp thông qua critic-guided representation learning, cho phép mô hình tập trung nhiều hơn vào các sự kiện log mang tính quyết định. Nhờ đó, DABLog cải thiện độ chính xác và giảm tỷ lệ cảnh báo sai trong các môi trường thực tế quy mô lớn. Tuy nhiên, lợi ích này đi kèm với chi phí tính toán và độ phức tạp mô hình cao hơn so với DeepLog.

2.6.2. Embedding layer

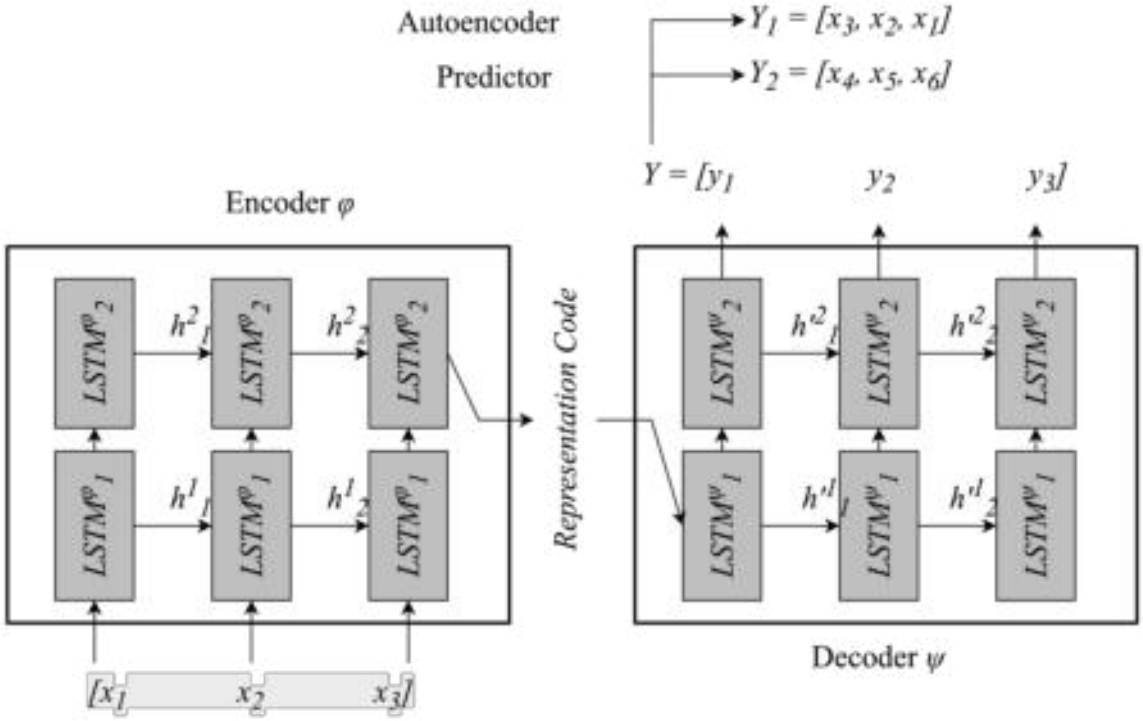
Đầu vào của mô hình là chuỗi chỉ số nguyên $X = [x_1, x_2, \dots, x_L]$, trong đó x_t là event id tại bước thời gian t, và L là độ dài chuỗi sau khi chuẩn hoá bằng padding hoặc cắt. Tập event id được sinh từ codebook, codebook ánh xạ mỗi token sự kiện rời rạc sang một chỉ số duy nhất trong miền $[0, \mathrm{V} - 1]$, với V là kích thước từ vựng sự kiện.

Embedding layer biến mỗi event id thành vector dày có số chiều d thông qua ma trận tham số $W \in \mathbb{R}^{V \times d}$. Khi đó $e_t = W[x_t]$, và đầu ra của embedding là chuỗi vector $\mathrm{E} = [e_1, e_2, \dots, e_L] \in \mathbb{R}^{L \times d}$. Việc chuyển từ biểu diễn rời rạc sang không gian liên tục giúp mô hình học được quan hệ gần xa giữa các sự kiện, thay vì chỉ coi các event id là các nhãn độc lập.

Để xử lý padding, thường quy ước một chỉ số dành riêng cho PAD, ví dụ PAD bằng 0. Khi bật masking, các vị trí có $x_t = 0$ sẽ bị tạo mask để các lớp phía sau có thể bỏ qua trong quá trình tính toán và cập nhật. Quy ước này kéo theo yêu cầu codebook không gán 0 cho sự kiện hợp lệ, và tham số input dim cần tính đúng theo cách đánh số của từ vựng.

2.6.3. Deep LSTM Autoencoder

Khối autoencoder gồm encoder và decoder, dùng LSTM xếp chồng để tăng năng lực biểu diễn. Mục tiêu của khối này là học biểu diễn ngữ cảnh theo thời gian của chuỗi sự kiện, và tạo ra biểu diễn phù hợp để decoder sinh ra chuỗi trạng thái phục vụ phân loại theo từng bước. Encoder nhận chuỗi embedding $\mathbb{R}^{L \times d}$ và biến đổi thành biểu diễn ngữ cảnh. Với cấu hình nhiều tầng, các tầng trung gian thường đặt return sequences bằng True để truyền toàn bộ chuỗi trạng thái ẩn sang tầng kế tiếp, giúp các tầng sâu học quan hệ phức tạp hơn theo thời gian.



Hình 2.12: Deep LSTM Autoencoder Network.

Tại nút thắt, có hai lựa chọn kiến trúc. Lựa chọn thứ nhất là lấy vector trạng thái tóm tắt của toàn chuỗi, gọi là $h \in \mathbb{R}^m$, rồi đưa qua RepeatVector để lặp h thành chuỗi dài L, tạo tensor $\mathbb{R}^{L\times m}$ làm đầu vào cho decoder. RepeatVector nhận tensor 2D và trả ra tensor 3D bằng cách lặp n lần theo trục thời gian.

Lựa chọn thứ hai là giữ dạng chuỗi ở nút thắt, tức encoder trả $\mathbb{R}^{L\times m}$, rồi decoder nhận trực tiếp chuỗi này. Lựa chọn này giữ nhiều thông tin theo thời gian hơn, nhưng phụ thuộc cách thiết kế số tầng và kích thước ẩn để tránh quá khớp. Decoder là một hoặc nhiều tầng LSTM đặt return sequences bằng True để sinh ra chuỗi trạng thái ẩn $\mathbb{R}^{L\times r}$, trong đó r là số chiều trạng thái của decoder. Chuỗi trạng thái này được chuyển sang Event classifier để tạo phân phối xác suất trên từ vựng sự kiện theo từng bước thời gian.

2.6.4. Event classifier, softmax theo từng bước thời gian

Event classifier nhận đầu ra theo chuỗi từ decoder, và tại mỗi bước thời gian t sinh ra một phân phối xác suất trên toàn bộ V sự kiện. Về mặt toán học, với trạng thái decoder $s_t \in \mathbb{R}^r$, bộ phân loại tạo:

$$p_t = \text{softmax}(W_c s_t + b_c) \tag{2.8}$$

Trong đó $p_t \in \mathbb{R}^V$ và tổng các phần tử bằng 1. Trong Keras, triển khai điển hình là TimeDistributed bọc một lớp Dense có softmax, để cùng một lớp Dense được áp dụng cho mọi lát thời gian của tensor đầu vào. TimeDistributed là wrapper áp một layer lên từng lát thời gian, và coi chiều thứ hai là chiều thời gian. Đầu ra của khối này là tensor xác suất $\mathbb{R}^{L\times V}$, được sử dụng trực tiếp bởi Anomaly critic. Cách thiết kế đầu ra theo từng bước thời gian cũng tương thích với hướng mô hình hoá chuỗi sự kiện kiểu dự đoán phân phối cho sự kiện tại mỗi bước. [7]

2.6.5. Anomaly critic

Anomaly critic là khối hậu xử lý, nhận vào hai luồng thông tin. Luồng thứ nhất là chuỗi phân phối softmax $\mathbb{R}^{L\times V}$ từ Event classifier. Luồng thứ hai là chuỗi sự kiện quan sát X cùng các thông tin ngữ cảnh, như subject và time bucket. Nhiệm vụ của critic là biến các phân phối theo thời gian thành các đại lượng phục vụ cảnh báo, gồm điểm bất thường theo từng bước, và các thống kê tổng hợp theo chuỗi. Ví dụ các thống kê thường dùng gồm số bước bất thường, tỉ lệ phần trăm bất thường, và các thống kê min, trung bình, max của điểm theo bước. Các bản ghi tổng hợp này được thiết kế để dễ lưu trữ và dễ lọc trên giao diện vận hành. Critic có thể tổ chức theo hai nhánh tính điểm. Nhánh xếp hạng dựa trên thứ tự xác suất của sự kiện quan sát trong danh sách dự đoán. Nhánh khoảng cách dựa trên quan hệ trong không gian embedding hoặc không gian biểu

diễn ẩn. Công thức tính điểm và quy tắc đặt ngưỡng được trình bày ở mục 2.7, phần này chỉ mô tả vai trò và giao diện vào ra của critic.

2.7 Hàm mất mát và tiêu chí phát hiện bất thường

2.7.1. Hàm mất mát categorical cross entropy

Đầu ra của Event classifier tại mỗi bước thời gian $\boldsymbol{t}$ là một phân phối softmax $\boldsymbol{p_t}$ trên tập từ vựng sự kiện có kích thước $\boldsymbol{V}$. Nhãn mục tiêu tại bước $\boldsymbol{t}$ được biểu diễn dưới dạng one hot $\boldsymbol{y_t}$, trong đó phần tử đúng của sự kiện quan sát bằng 1 và các phần tử còn lại bằng 0. Hàm mất mát tại một bước thời gian được định nghĩa bằng categorical cross entropy.

$$\mathcal{L}_t = -\sum_{k=1}^{V} y_{t,k} \log(p_{t,k}) \tag{2.9}$$

Với một chuỗi độ dài L, mất mát của một mẫu thường được lấy trung bình hoặc lấy tổng theo thời gian.

$$\mathcal{L} = \frac{1}{L}\sum_{t=1}^{L} \mathcal{L}_t \tag{2.10}$$

Khi có padding, các bước tương ứng token PAD cần được loại khỏi tính toán mất mát bằng masking, nhằm tránh việc mô hình học theo các phần đệm không mang ý nghĩa hành vi.

2.7.2. Tiêu chí bất thường theo xếp hạng

Tại mỗi bước thời gian t, từ phân phối p_t có thể lấy ra tập $\mathrm{TopN}(p_t)$, là N sự kiện có xác suất cao nhất. Gọi x_t là sự kiện quan sát thực tế tại bước t. Khi đó một bước được xem là dự đoán trượt, nếu x_t không thuộc tập Top N.

$$miss_t = \begin{cases} 1, & x_t \notin TopN(p_t) \\ 0, & x_t \in TopN(p_t) \end{cases} \tag{2.11}$$

Cách kiểm tra Top N là một tiêu chí điển hình trong các mô hình phát hiện bất thường từ chuỗi log theo hướng dự đoán sự kiện tiếp theo, tiêu biểu như DeepLog. Ngoài cách đếm trượt nhị phân, có thể sử dụng thước đo mềm dựa trên thứ hạng. Gọi $rank_t$ là thứ hạng của x_t khi sắp xếp $p_{t,k}$, giảm dần, khi đó có thể chuẩn hoá.

$$score_t = \frac{rank_t - 1}{V - 1} \tag{2.12}$$

Giá trị càng gần 0 thì càng phù hợp với dự đoán của mô hình, giá trị càng gần 1 thì càng bất thường. Ở mức chuỗi, một tiêu chí đơn giản là tổng số trượt và tỉ lệ trượt.

$$miss = \sum_{t=1}^{L} miss_t \ , \qquad miss\_pct = \frac{miss}{L} \times 100 \tag{2.13}$$

Sau đó đặt ngưỡng τ để kết luận chuỗi bất thường, ví dụ cảnh báo khi $\text{miss}_{\text{pct}} \geq \tau$, hoặc khi trung bình $score_t$ vượt ngưỡng. Việc chọn N và τ phản ánh đánh đổi giữa bỏ sót và cảnh báo nhầm, phần lựa chọn tham số được trình bày ở mục 2.8.

2.7.3. Tiêu chí bất thường theo ngưỡng khoảng cách

Tiêu chí Top N đôi khi đánh dấu bất thường ngay cả khi dự đoán sai nhưng các ứng viên dự đoán vẫn gần về mặt ngữ nghĩa sự kiện. Để tăng độ tin cậy, có thể bổ sung một tiêu chí khoảng cách trong không gian biểu diễn. Một cách phổ biến là dùng khoảng cách trong không gian embedding. Gọi $V(k)$ là vector embedding của sự kiện k. Tại bước t, lấy tập Top N như trên và định nghĩa khoảng cách nhỏ nhất giữa sự kiện quan sát và các ứng viên dự đoán.

$$d_t = \min_{k \in TopN(p_t)} \| V(x_t) - V(k) \| \tag{2.14}$$

Khi d_t lớn hơn một ngưỡng δ, bước t được xem là bất thường theo nghĩa khoảng cách, kể cả khi mô hình đã dự đoán một số ứng viên có xác suất cao. Ngưỡng δ thường được thiết lập theo phân phối d_t trên dữ liệu bình thường, ví dụ chọn theo percentile để kiểm soát tỉ lệ cảnh báo nhầm.

2.7.4. Kết hợp tiêu chí xếp hạng và khoảng cách

Hai tiêu chí có thể kết hợp theo quy tắc logic hoặc theo điểm tổng hợp. Một cách dễ triển khai là cảnh báo bước t khi đồng thời xảy ra trượt Top N và khoảng cách vượt ngưỡng.

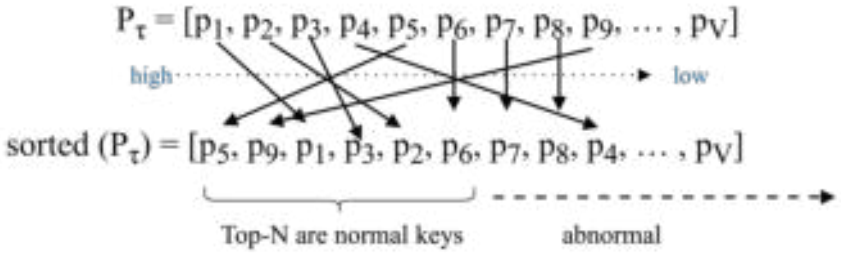
$$abnormal_t = 1[x_t \notin TopN(p_t)] \cdot 1[d_t \geq \delta] \tag{2.15}$$

Cách kết hợp này có xu hướng giảm cảnh báo nhầm trong các trường hợp dự đoán sai nhưng vẫn gần về ý nghĩa, đồng thời giữ khả năng bắt các sai lệch rõ rệt về cấu trúc chuỗi.

Trong thiết lập tham chiếu, baseline predictor-based được xem như một biến thể theo hướng DeepLog/nLSALog, triển khai bởi mạng LSTM 2 tầng, một bộ phân loại đa lớp, và rank-based critic để kết luận chuỗi là bình thường/bất thường. Baseline sử dụng softmax cho lớp phân loại và (tương tự DabLog) dùng embedding học được thay vì one-hot trực tiếp. Đánh giá được trình bày dưới dạng xu hướng F1-score theo ngưỡng xếp hạng chuẩn hoá:

$$\rho_N = \frac{N}{|K|} \times 100\% \tag{2.16}$$

tương đương chọn Top-N trên tổng số log keys | K | .



Hình 2.13: Một ví dụ về tiêu chí dựa trên thứ hạng.

2.8 Thiết lập huấn luyện và siêu tham số

2.8.1. Môi trường tính toán và phần cứng huấn luyện

Quá trình huấn luyện được thực hiện trên máy có một GPU NVIDIA RTX 4090 với 24GB VRAM, CPU AMD EPYC 7H12 64 Core, RAM 126GB, và dung lượng đĩa 927GB. Cấu hình này cho phép huấn luyện mô hình chuỗi với batch size tương đối lớn, đồng thời giảm đáng kể thời gian huấn luyện so với cấu hình chỉ dùng CPU.

Bảng 2.7: Bảng cấu hình phần cứng huấn luyện, gồm GPU, VRAM, CPU, RAM, Disk.

Hạng mục	Cấu hình
GPU	NVIDIA GeForce RTX 4090
Số lượng GPU	1
VRAM	24 GB
CPU	AMD EPYC 7H12, 64-Core Processor
RAM	126 GB
Dung lượng đĩa	927 GB
Hệ điều hành	Linux

2.8.2. Nhóm siêu tham số dữ liệu, cửa sổ thời gian và độ dài chuỗi

Nhóm tham số này quyết định mô hình nhìn thấy dữ liệu dưới dạng chuỗi như thế nào, và ảnh hưởng trực tiếp tới mức ngữ cảnh mà mô hình học được.

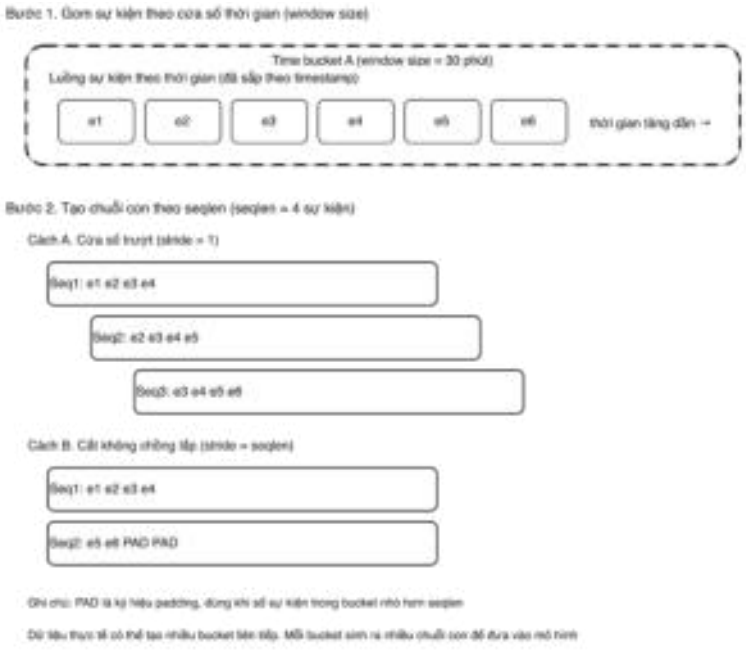
Window size là độ rộng cửa sổ thời gian dùng để gom sự kiện thành một đoạn hoạt động. Khi window size nhỏ, chuỗi phản ánh hành vi rất cục bộ theo thời gian, mô hình nhạy với biến động ngắn nhưng có thể bỏ sót mẫu bất thường cần quan sát lâu hơn.

Khi window size lớn, chuỗi bao quát hành vi dài hơn, tăng khả năng nắm bắt mẫu tấn công theo giai đoạn, nhưng dễ trộn nhiều hoạt động khác nhau vào cùng chuỗi và làm tăng nhiễu.

Seqlen là độ dài chuỗi theo số sự kiện được dùng để đưa vào mô hình. Seqlen nhỏ làm mô hình tập trung vào mẫu ngắn và giảm chi phí tính toán, nhưng giảm khả năng học phụ thuộc dài. Seqlen lớn giúp mô hình học được ngữ cảnh dài hơn, nhưng tăng bộ nhớ và thời gian huấn luyện, đồng thời làm gradient khó ổn định hơn trong mạng hồi tiếp.

Các tham số window size và seqlen có quan hệ với nhau. Window size quyết định phạm vi thời gian của chuỗi, seqlen quyết định số lượng sự kiện mô hình xử lý trong chuỗi đó. Khi window size tăng nhưng seqlen không tăng, chuỗi sẽ thưa hơn theo thời gian. Khi seqlen tăng nhưng window size không tăng, chuỗi sẽ dày hơn và tập trung hơn theo thời gian.

Ngoài ra, lựa chọn hai tham số này nên dựa trên đặc trưng lưu lượng thực tế (tốc độ sinh sự kiện, chu kỳ hoạt động, và loại tấn công mục tiêu). Trong triển khai vận hành, có thể dò tham số bằng cách so sánh phân bố số sự kiện mỗi cửa sổ và tỷ lệ chuỗi bị cắt (truncation) hoặc phải đệm (padding). Một chiến lược thực dụng là thử một vài cấu hình theo "lưới nhỏ" (grid) rồi chọn điểm cân bằng giữa chất lượng phát hiện và chi phí tài nguyên khi chạy liên tục.



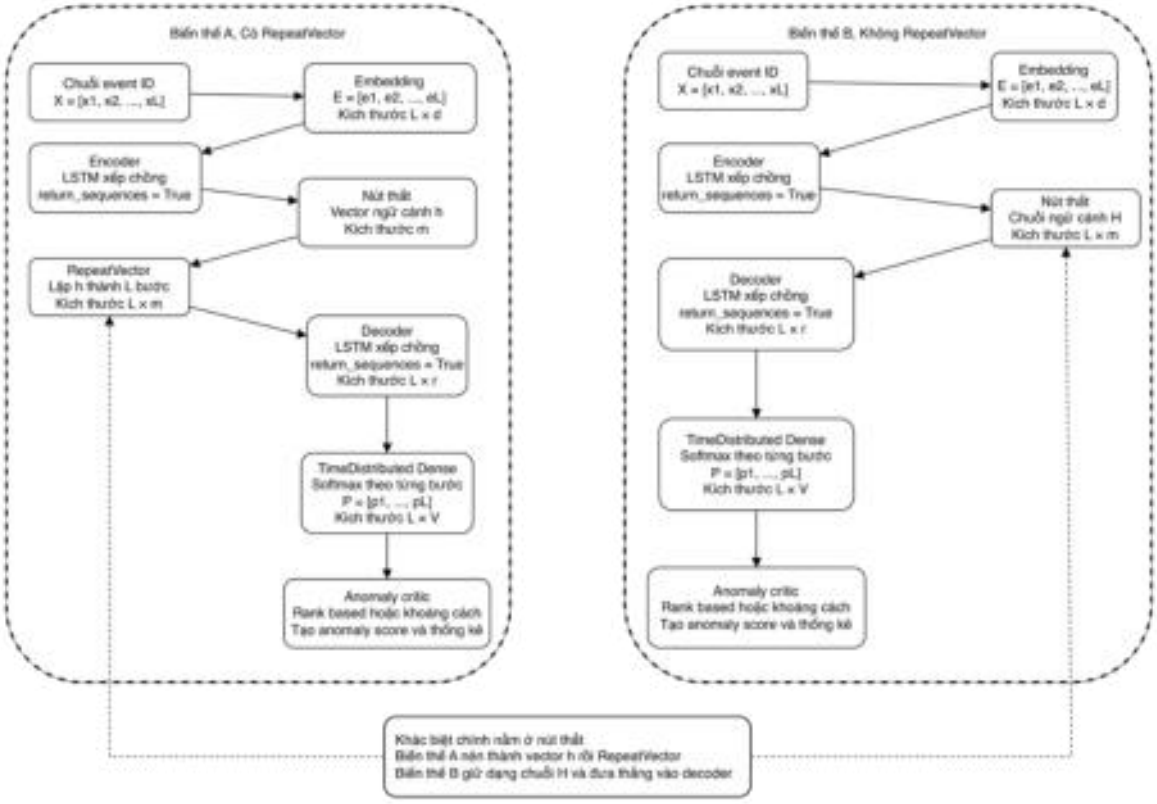
Hình 2.14: Minh hoạ cách window size tạo time bucket và cách seqlen cắt hoặc trượt để tạo các chuỗi con.

2.8.3. Nhóm siêu tham số biểu diễn, kích thước từ vựng và embedding

Nhóm tham số này quyết định mức độ chi tiết của không gian sự kiện, và cách sự kiện được đưa vào LSTM dưới dạng vector liên tục. Kích thước từ vựng V phụ thuộc cách tạo log key và cách lọc các sự kiện hiếm, V càng lớn thì mô hình phân biệt được nhiều loại hành vi hơn, nhưng lớp softmax sẽ nặng hơn và dữ liệu sẽ thưa hơn, dẫn đến nguy cơ học kém cho các sự kiện ít xuất hiện. V nhỏ giúp mô hình học ổn định hơn do dữ liệu ít thưa, nhưng dễ mất khả năng phân biệt các tình huống khác nhau nếu log key bị gom quá mạnh.

Kích thước embedding d là số chiều vector biểu diễn cho mỗi sự kiện, d nhỏ làm biểu diễn bị nén mạnh, dễ underfit nếu số loại hành vi đa dạng, d lớn tăng khả năng biểu diễn và giúp mô hình học quan hệ phức tạp giữa các sự kiện, nhưng tăng tham số, tăng nguy cơ quá khớp, và tăng chi phí tính toán. Từ góc độ vận hành, V và d còn ảnh hưởng trực tiếp tới bộ nhớ GPU. Lớp embedding có ma trận kích thước $V \times d$, và lớp softmax có ma trận trọng số tỷ lệ với $r \times V$, trong đó r là số chiều trạng thái decoder. Vì vậy, việc chọn V và d cần cân bằng giữa độ chi tiết, tài nguyên, và khả năng tổng quát.

2.8.4. Nhóm siêu tham số kiến trúc LSTM autoencoder



Hình 2.15: Hai biến thể kiến trúc LSTM Autoencoder, có RepeatVector và không RepeatVector.

Nhóm tham số này quyết định năng lực biểu diễn của mô hình theo thời gian, và ảnh hưởng tới khả năng tái tạo chuỗi theo nghĩa xác suất ở đầu ra. Hidden units là số chiều trạng thái ẩn của mỗi tầng LSTM. Giá trị này càng lớn thì mô hình càng có khả năng giữ thông tin và học quan hệ phi tuyến phức tạp, nhưng tăng tham số và tăng nguy cơ quá khớp nếu dữ liệu không đủ đa dạng. Hidden layers là số tầng LSTM xếp chồng trong encoder và decoder. Tăng số tầng giúp mô hình học biểu diễn phân cấp, tầng dưới học mẫu cục bộ, tầng trên học quan hệ trừu tượng hơn. Tuy nhiên khi số tầng tăng, việc huấn luyện khó ổn định hơn và dễ xảy ra quá khớp nếu không có điều chuẩn phù hợp.

Tuỳ chọn RepeatVector quyết định dạng nút thắt. Khi bật RepeatVector, encoder nén chuỗi thành một vector và decoder nhận chuỗi được tạo bằng cách lặp vector này theo thời gian. Thiết kế này làm "biểu diễn nén" có vai trò rõ ràng và thường giúp mô hình học một dạng tóm tắt ổn định. Khi không dùng RepeatVector, nút thắt giữ dạng chuỗi và decoder nhận trực tiếp chuỗi biểu diễn, điều này giữ nhiều thông tin theo thời gian hơn nhưng thường đòi hỏi điều chuẩn tốt để tránh mô hình ghi nhớ.

2.8.5. Nhóm siêu tham số tối ưu hoá

Nhóm tham số này quyết định cách mô hình được tối ưu và ảnh hưởng tới tốc độ hội tụ, độ ổn định, và nguy cơ quá khớp.

Epochs là số vòng lặp qua toàn bộ tập huấn luyện. Khi epochs quá ít, mô hình chưa kịp học cấu trúc chuỗi, dẫn đến underfit. Khi epochs quá nhiều, mô hình có thể học quá sát dữ liệu huấn luyện, dẫn đến giảm khả năng tổng quát, đặc biệt trong dữ liệu có drift theo thời gian.

Batch size là số mẫu trong mỗi lần cập nhật gradient. Batch size lớn giúp gradient ổn định hơn và tận dụng GPU tốt hơn, nhưng yêu cầu VRAM cao. Batch size nhỏ làm cập nhật nhiễu hơn, có thể giúp thoát khỏi nghiệm kém nhưng dễ làm loss dao động.

Early stopping là cơ chế dừng huấn luyện khi chỉ số theo dõi không còn cải thiện, thường theo validation loss. Cơ chế này giúp tránh chạy quá nhiều epoch dẫn đến quá khớp, đồng thời chọn checkpoint tốt nhất theo khả năng tổng quát. Patience quyết định số epoch chờ trước khi dừng, patience nhỏ dừng sớm hơn và giảm thời gian, patience lớn cho mô hình thêm cơ hội cải thiện nhưng tăng thời gian và nguy cơ overfit.

2.8.6. Nhóm tham số liên quan tiêu chí phát hiện và hiệu chỉnh sau huấn luyện

Mặc dù tiêu chí phát hiện đã trình bày ở mục 2.7, trong thực nghiệm huấn luyện cần coi các tham số như Top N, rank threshold, và ngưỡng khoảng cách là các tham số hiệu chỉnh để chọn điểm làm việc của hệ thống. Những tham số này không trực tiếp

tham gia tối ưu loss, nhưng quyết định mức nhạy của cảnh báo và tác động tới Precision và Recall trong đánh giá. Do dữ liệu bất thường thường mất cân bằng, việc chọn tham số nên ưu tiên các thước đo như PR-AUC/F1 thay vì chỉ nhìn Accuracy. Ngoài chất lượng, tham số còn cần phù hợp năng lực vận hành (số cảnh báo SOC có thể review mỗi ngày).

Top N càng lớn thì xác suất sự kiện thật nằm trong top dự đoán càng cao, do đó giảm miss, giảm cảnh báo nhầm, nhưng tăng nguy cơ bỏ sót bất thường tinh vi. Rank threshold càng nhỏ thì tiêu chí càng nghiêm, cảnh báo nhiều hơn. Ngưỡng khoảng cách càng nhỏ thì nhánh khoảng cách càng "khó thoả" và làm giảm cảnh báo nhầm, nhưng có thể bỏ sót nếu embedding chưa học đủ tốt. Vì vậy, nên hiệu chỉnh theo một dải giá trị hợp lý trên tập validation và quan sát PR curve để chọn điểm cân bằng. Khi triển khai, cần theo dõi drift (ví dụ tỷ lệ miss nền tăng) để kịp điều chỉnh tham số khi hành vi mạng thay đổi.

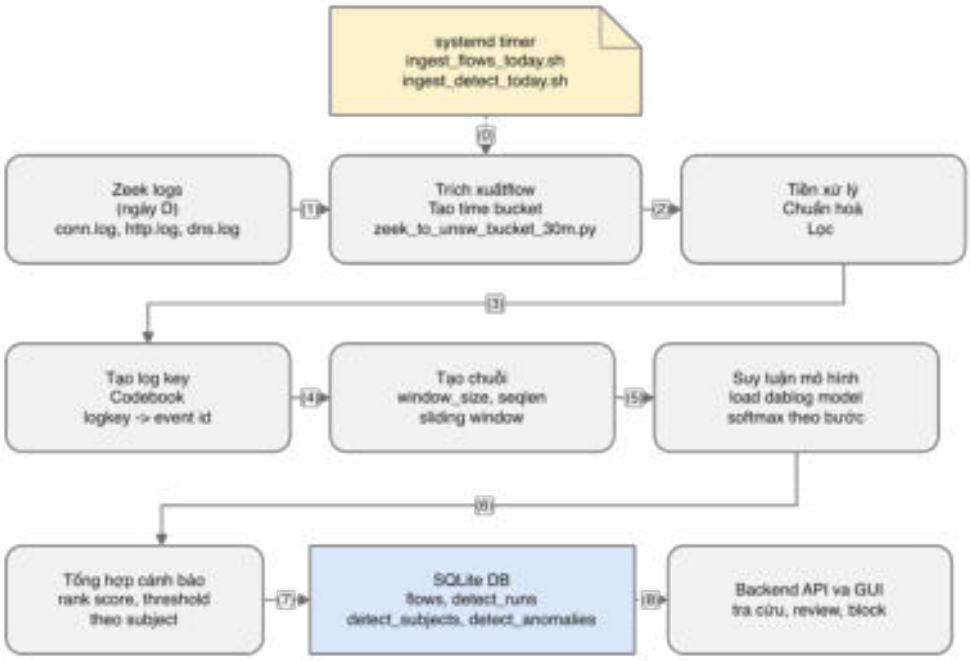
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Khảo sát bài toán thực tế và yêu cầu hệ thống

3.1.1 Bối cảnh vận hành, phạm vi hệ thống, giả định thiết kế, dữ liệu vào ra

Hệ thống được xây dựng trong bối cảnh mạng có lưu lượng ra vào liên tục, khó kiểm soát hoàn toàn bằng luật tĩnh. Việc phát hiện dựa trên chữ ký có thể bỏ sót các hành vi mới hoặc biến thể. Vì vậy hệ thống tập trung vào phát hiện bất thường dựa trên chuỗi sự kiện, sau đó hỗ trợ quy trình vận hành gồm quan sát, duyệt cảnh báo, và phản ứng chặn IP khi cần. Phạm vi hệ thống tập trung vào lớp giám sát mạng và lớp ứng dụng phục vụ vận hành. Nguồn dữ liệu chính là log Zeek, được thu thập theo thời gian và được xử lý theo lịch. Hệ thống không thay thế toàn bộ giải pháp SIEM, mà đóng vai trò như một thành phần phát hiện bất thường và hỗ trợ phản ứng nhanh, với dữ liệu và giao diện đủ để truy vết lại sự kiện gốc.

Giả định thiết kế gồm. Máy giám sát có thể chạy Zeek ổn định và xuất log theo cấu trúc chuẩn. Quá trình suy luận chạy theo lịch theo ngày hoặc theo các mốc định kỳ, và kết quả được lưu lại để truy vấn nhanh. Người vận hành có tài khoản đăng nhập và có quyền duyệt cảnh báo. Chức năng chặn IP được thực hiện có kiểm soát, và mọi thao tác được ghi nhận để kiểm toán.



Hình 3.1: Sơ đồ luồng hoạt động của Detect.

Dữ liệu đầu vào gồm:

- Log Zeek theo ngày, ví dụ conn log và các log giao thức khác nếu sử dụng.
- Codebook và mô hình đã được huấn luyện, dùng để mã hoá sự kiện và suy luận nhất quán.
- Cấu hình xử lý như window size, seqlen, và các tham số phục vụ phát hiện, được lưu theo phiên bản để tái lập.

Dữ liệu đầu ra gồm:

- danh sách cảnh báo bất thường theo ngày và theo subject, kèm các chỉ số như miss, total, miss pct, score min, score avg, score max, và trạng thái xử lý review.
- Nhật ký thao tác vận hành, gồm duyệt cảnh báo và thao tác chặn hoặc bỏ chặn IP.
- Danh sách IP bị chặn và trạng thái hiệu lực, để cơ chế firewall áp dụng hoặc gỡ bỏ khi cần.

Bảng 3.1: Tóm tắt dữ liệu đầu vào và đầu ra.

Nhóm dữ liệu	**Nguồn**	**Định dạng điển hình**	**Tần suất**	**Mục đích sử dụng**
Log mạng	Zeek	TSV hoặc JSON theo schema log	Theo phút hoặc theo file xoay vòng, tổng hợp theo ngày	Nguồn sự kiện để tiền xử lý, tạo chuỗi và suy luận
Mô hình suy luận	Mô hình học sâu đã huấn luyện	File mô hình	Cập nhật theo phiên bản	Tạo phân phối dự đoán softmax theo từng bước thời gian
Codebook	Pipeline tiền xử lý	File từ điển	Đồng bộ với mô hình	Mã hoá log key thành event id, xử lý UNK và PAD
Cấu hình xử lý	Cấu hình hệ thống	JSON hoặc biến môi trường	Thay đổi khi hiệu chỉnh	Quy định window size, seqlen, Top N,

				rank threshold, ngưỡng khoảng cách
Cảnh báo bất thường	Khối inference	Bản ghi trong CSDL	Theo lịch chạy	Hiển thị, lọc, duyệt và truy vết
Nhật ký thao tác	Backend	Bản ghi trong CSDL	Theo thao tác	Kiểm toán duyệt cảnh báo và chặn IP
Danh sách chặn IP	Backend và firewall	Bản ghi CSDL và tập luật firewall	Theo thao tác	Thực thi phản ứng chặn và quản trị vòng đời

3.1.2 Yêu cầu chức năng

Hệ thống cần hỗ trợ đầy đủ chu trình vận hành từ thu thập log mạng đến phát hiện bất thường và xử lý sau cảnh báo. Các yêu cầu chức năng được mô tả theo nhóm để thuận lợi cho thiết kế kiến trúc, dữ liệu, API và giao diện. Bao gồm:

- Nhóm chức năng thu thập dữ liệu và xử lý theo lịch: hệ thống tiếp nhận log Zeek theo ngày hoặc theo mốc định kỳ. Hệ thống thực hiện tiền xử lý và chuẩn hoá theo cùng cấu hình đã dùng khi xây dựng mô hình, gồm tạo log key, mã hoá event id bằng codebook, tạo chuỗi theo subject và time bucket, xử lý UNK và PAD, và chuẩn hoá độ dài chuỗi.
- Nhóm chức năng suy luận và sinh cảnh báo: hệ thống chạy suy luận trên các chuỗi đã tạo, sau đó tính điểm bất thường theo tiêu chí cấu hình và tổng hợp thành bản ghi cảnh báo theo subject và theo thời gian. Cảnh báo cần chứa các trường đủ để lọc nhanh và truy vết, gồm thời gian, IP, subject, mức độ bất thường theo các thống kê tổng hợp, và liên kết tới chi tiết phục vụ điều tra.
- Nhóm chức năng tra cứu, lọc và xem chi tiết trên giao diện: Giao diện cung cấp danh sách cảnh báo theo ngày, hỗ trợ lọc theo trạng thái duyệt và tìm kiếm theo từ khoá. Người vận hành có thể mở chi tiết để xem đầy đủ trường dữ liệu, sau đó cập nhật trạng thái duyệt. Trạng thái duyệt được lưu để phản ánh vòng đời xử lý cảnh báo.
- Nhóm chức năng quản lý danh sách chặn IP: hệ thống cho phép thêm, gỡ, bật tắt chặn IP, có lưu lý do, thời điểm và người thao tác. Nếu tích hợp firewall, danh

sách chặn hiệu lực được đồng bộ xuống tầng firewall để thực thi. Mọi thao tác chặn và gỡ chặn đều được ghi nhận để truy vết.

- Nhóm chức năng tra cứu log và hỗ trợ điều tra: hệ thống cung cấp khả năng tra cứu log hoặc flow theo ngày, theo IP và theo một số trường cơ bản, nhằm hỗ trợ xác minh cảnh báo và đối chiếu ngữ cảnh trước khi quyết định phản ứng.
- Nhóm chức năng xác thực, phân quyền và quản trị: hệ thống yêu cầu đăng nhập trước khi truy cập. Các thao tác nhạy cảm như duyệt cảnh báo, chặn IP, quản lý tài khoản cần được phân quyền rõ ràng. Hệ thống lưu nhật ký thao tác tối thiểu cho đăng nhập, duyệt, chặn và gỡ chặn để phục vụ kiểm toán.

3.1.3 Yêu cầu phi chức năng

Yêu cầu phi chức năng được xác định để đảm bảo hệ thống vận hành ổn định trong môi trường thực tế, xử lý được dữ liệu theo lịch, truy vấn nhanh trên giao diện, và kiểm soát được rủi ro bảo mật khi có chức năng duyệt và chặn IP.

A. Hiệu năng

Hệ thống cần đáp ứng hai nhóm tải chính, tải xử lý theo lịch và tải truy vấn từ giao diện. Khối xử lý theo lịch cần hoàn thành suy luận và ghi cảnh báo theo ngày trong khoảng thời gian chấp nhận được để kết quả sẵn sàng cho người vận hành trong ngày làm việc. Backend API cần phản hồi nhanh cho các thao tác phổ biến như tải danh sách cảnh báo theo ngày, lọc theo trạng thái duyệt, mở chi tiết bản ghi, và thao tác cập nhật trạng thái duyệt. Truy vấn theo ngày và theo trạng thái cần được tối ưu bằng chỉ mục để tránh quét toàn bảng khi dữ liệu tăng theo thời gian.

B. Độ tin cậy và tính sẵn sàng

Hệ thống cần đảm bảo không mất dữ liệu cảnh báo khi có lỗi tạm thời ở một thành phần. Các bước ingest và suy luận cần có cơ chế ghi log lỗi và có thể chạy lại theo ngày mà không tạo trùng dữ liệu, hoặc có quy tắc tránh trùng rõ ràng theo khoá định danh. Backend cần xử lý được các tình huống lỗi phổ biến như hết phiên đăng nhập, thiếu tham số, truy vấn ngày không tồn tại dữ liệu, và trả về thông báo rõ ràng để giao diện hiển thị phù hợp.

C. Bảo mật

Hệ thống cần yêu cầu đăng nhập trước khi truy cập các màn hình và API. Các thao tác nhạy cảm như duyệt cảnh báo, chặn IP, gỡ chặn IP, và quản trị tài khoản cần được kiểm tra quyền ở phía backend. Dữ liệu xác thực cần được bảo vệ, không lưu mật khẩu dạng rõ, và giới hạn quyền truy cập vào các endpoint quản trị. Hệ thống cần ghi

nhận các sự kiện bảo mật cơ bản như đăng nhập thất bại, truy cập endpoint không đủ quyền, và thao tác chặn hoặc gỡ chặn IP.

D. Khả năng mở rộng

Hệ thống cần có khả năng mở rộng theo dữ liệu và theo chức năng. Theo dữ liệu, khi số cảnh báo tăng theo ngày, truy vấn danh sách theo ngày vẫn cần giữ được tốc độ chấp nhận được nhờ chỉ mục và phân trang. Theo chức năng, kiến trúc cần cho phép bổ sung thêm nguồn log Zeek khác hoặc bổ sung loại cảnh báo mới mà không phá vỡ cấu trúc dữ liệu hiện có, tối thiểu giữ tương thích ở mức API và giao diện.

E. Khả năng bảo trì và quan sát hệ thống

Hệ thống cần có log đủ để truy vết luồng xử lý, gồm log của ingest, log của backend API, và log thao tác người dùng. Cấu hình cần tách khỏi mã nguồn ở mức có thể, để thay đổi lịch chạy hoặc tham số lọc hiển thị mà không phải sửa nhiều nơi. Cần có cơ chế sao lưu cơ sở dữ liệu theo chu kỳ để giảm rủi ro mất dữ liệu vận hành.

F. Tính dùng được

Giao diện cần hỗ trợ lọc theo ngày và theo trạng thái duyệt, có tìm kiếm nhanh, có xem chi tiết bản ghi, và phản hồi rõ khi không có dữ liệu. Trạng thái bộ lọc và thao tác duyệt cần phản ánh nhất quán sau khi tải lại dữ liệu để tránh nhầm lẫn khi vận hành.

3.2 Mô hình tác nhân và ca sử dụng

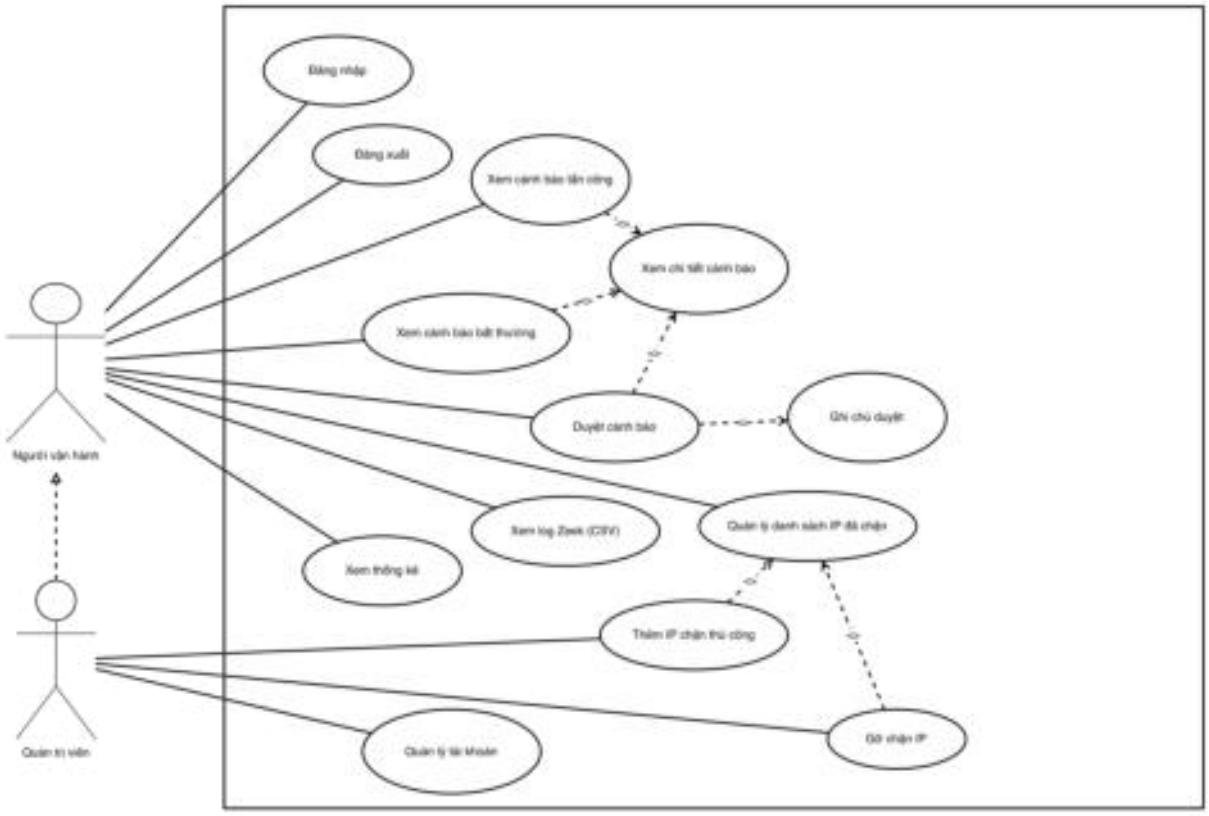
3.2.1 Các tác nhân trong hệ thống

Trong phạm vi sơ đồ ca sử dụng, tác nhân chỉ xét các vai trò có con người tương tác trực tiếp với hệ thống. Do đó hệ thống có hai tác nhân chính

- Người vận hành: người vận hành sử dụng hệ thống để theo dõi cảnh báo theo ngày, lọc theo trạng thái xử lý, xem chi tiết cảnh báo và thực hiện duyệt cảnh báo. Mục tiêu là xác nhận cảnh báo, ưu tiên xử lý và tạo trạng thái xử lý nhất quán.
- Quản trị viên: quản trị viên thực hiện toàn bộ chức năng của người vận hành, đồng thời có thêm quyền thao tác nhạy cảm như quản lý danh sách IP chặn, chặn và gỡ chặn IP, quản lý tài khoản và phân quyền người dùng, theo dõi nhật ký thao tác.

3.2.2 Sơ đồ ca sử dụng tổng quát của hệ thống

Phần này trình bày hai sơ đồ ca sử dụng tổng quát, một cho Người vận hành và một cho Quản trị viên. Quản trị viên kế thừa toàn bộ ca sử dụng của người vận hành và bổ sung các ca quản trị.



Hình 3.2: Sơ đồ ca sử dụng tổng quát.

3.2.3 Đặc tả ngắn các ca sử dụng chính

Phần này đặc tả các ca sử dụng quan trọng theo dạng bảng, tương tự mẫu báo cáo bạn gửi. Các ca được chọn là những ca ảnh hưởng trực tiếp đến quy trình vận hành, gồm đăng nhập, xem danh sách cảnh báo, xem chi tiết và duyệt cảnh báo, quản lý IP chặn, và quản lý tài khoản.

A. Ca sử dụng đăng nhập được mô tả như Bảng 3.2.

Bảng 3.2: Đặc tả ca sử dụng “Đăng nhập”.

Thuộc tính	Nội dung
Tên ca sử dụng	Đăng nhập
Tác nhân	Người vận hành, Quản trị viên

Mô tả	Cho phép người dùng đăng nhập để truy cập hệ thống và thực hiện các chức năng theo quyền
Điều kiện kích hoạt	Người dùng truy cập hệ thống nhưng chưa có phiên hợp lệ, chọn chức năng đăng nhập
Tiền điều kiện	Tài khoản đã được tạo và đang ở trạng thái cho phép sử dụng
Luồng chính	1 Người dùng nhập tên đăng nhập và mật khẩu. 2 Nhấn đăng nhập. 3 Hệ thống kiểm tra thông tin. 4 Nếu hợp lệ, hệ thống tạo phiên và chuyển vào trang chính
Kết quả	Người dùng truy cập được các màn hình tương ứng với vai trò
Trường hợp lỗi	Sai tài khoản hoặc mật khẩu. Tài khoản bị vô hiệu hoá. Phiên hết hạn hoặc lỗi hệ thống, hệ thống thông báo và yêu cầu đăng nhập lại

B. Ca sử dụng xem danh sách cảnh báo được mô tả như Bảng 3.3.

Bảng 3.3: Đặc tả ca sử dụng "Xem danh sách cảnh báo".

Thuộc tính	**Nội dung**
Tên ca sử dụng	Xem danh sách cảnh báo
Tác nhân	Người vận hành, Quản trị viên
Mô tả	Cho phép xem danh sách cảnh báo theo ngày, lọc theo trạng thái, và tìm kiếm để ưu tiên xử lý
Điều kiện kích hoạt	Người dùng đã đăng nhập, chọn màn hình cảnh báo
Tiền điều kiện	Dữ liệu cảnh báo của ngày được xử lý đã có trong cơ sở dữ liệu
Luồng chính	1 Người dùng chọn ngày cần xem. 2 Hệ thống tải danh sách cảnh báo theo ngày. 3 Người dùng lọc theo trạng thái review hoặc tìm kiếm theo từ khoá. 4 Hệ thống cập nhật danh sách theo bộ lọc
Kết quả	Hiển thị danh sách cảnh báo phù hợp bộ lọc, sẵn sàng mở chi tiết
Trường hợp lỗi	Không có dữ liệu cho ngày đã chọn. Lỗi kết nối hoặc hết phiên, hệ thống thông báo và yêu cầu thao tác lại

C. Ca sử dụng xem chi tiết và duyệt cảnh báo được mô tả như Bảng 3.4.

Bảng 3.4: Đặc tả ca sử dụng "Xem chi tiết và duyệt cảnh báo"

Thuộc tính	Nội dung
Tên ca sử dụng	Xem chi tiết và duyệt cảnh báo
Tác nhân	Người vận hành, Quản trị viên
Mô tả	Cho phép mở chi tiết một cảnh báo và cập nhật trạng thái duyệt để phản ánh kết quả xác minh
Điều kiện kích hoạt	Người dùng đang xem danh sách cảnh báo, chọn một bản ghi để xem chi tiết
Tiền điều kiện	Bản ghi cảnh báo tồn tại và chưa bị xoá. Người dùng có quyền cập nhật trạng thái review
Luồng chính	1 Người dùng mở chi tiết cảnh báo. 2 Hệ thống hiển thị đầy đủ trường dữ liệu. 3 Người dùng chọn trạng thái duyệt. 4 Hệ thống ghi nhận trạng thái mới, người duyệt và thời điểm. 5 Danh sách được cập nhật theo trạng thái mới
Kết quả	Cảnh báo chuyển trạng thái và được ghi nhận để kiểm toán
Trường hợp lỗi	Cảnh báo không tồn tại hoặc bị thay đổi. Người dùng không đủ quyền. Lỗi ghi CSDL, hệ thống thông báo thất bại

D. Ca sử dụng quản lý IP chặn được mô tả như Bảng 3.5.

Bảng 3.5: Đặc tả ca sử dụng "Quản lý IP chặn".

Thuộc tính	Nội dung
Tên ca sử dụng	Quản lý IP chặn
Tác nhân	Quản trị viên
Mô tả	Cho phép thêm, gỡ hoặc bật tắt chặn IP, lưu lý do và trạng thái hiệu lực
Điều kiện kích hoạt	Quản trị viên đăng nhập và mở màn hình IP đã chặn
Tiền điều kiện	Quản trị viên có quyền thao tác chặn. Hệ thống cho phép cập nhật trạng thái chặn
Luồng chính	1 Nhập IP và lý do, chọn thêm chặn. 2 Hệ thống kiểm tra định dạng IP và ghi vào CSDL. 3 Khi gỡ hoặc tắt chặn, hệ thống cập nhật trạng thái và ghi nhận thời điểm. 4 Danh sách IP chặn cập nhật

Kết quả	Danh sách chặn phản ánh đúng trạng thái mới và có lịch sử thao tác
Trường hợp lỗi	IP không hợp lệ. IP đã tồn tại. Không ghi được CSDL. Không đồng bộ được xuống firewall khi có tích hợp, hệ thống thông báo lỗi

E. Ca sử dụng quản lý tài khoản được mô tả như Bảng 3.6.

Bảng 3.6: Đặc tả ca sử dụng "Quản lý tài khoản".

Thuộc tính	**Nội dung**
Tên ca sử dụng	Quản lý tài khoản
Tác nhân	Quản trị viên
Mô tả	Cho phép tạo tài khoản, phân vai trò, vô hiệu hoá tài khoản và quản trị quyền truy cập
Điều kiện kích hoạt	Quản trị viên đăng nhập và mở màn hình quản lý tài khoản
Tiền điều kiện	Hệ thống có cơ chế lưu người dùng và vai trò trong cơ sở dữ liệu
Luồng chính	1 Quản trị viên nhập thông tin tài khoản mới hoặc chọn tài khoản cần sửa. 2 Chọn vai trò và trạng thái. 3 Hệ thống kiểm tra hợp lệ và cập nhật CSDL. 4 Danh sách tài khoản được cập nhật
Kết quả	Tài khoản được tạo hoặc cập nhật đúng theo phân quyền
Trường hợp lỗi	Thông tin không hợp lệ. Trùng tài khoản. Không đủ quyền. Lỗi ghi CSDL, hệ thống thông báo thất bại

3.3 Kiến trúc tổng thể hệ thống

Kiến trúc hệ thống được thiết kế theo hướng tách rõ luồng xử lý dữ liệu theo lịch và luồng phục vụ vận hành qua giao diện. Luồng theo lịch chịu trách nhiệm thu thập log, chuẩn hoá, tạo chuỗi và suy luận để sinh cảnh báo. Luồng vận hành chịu trách nhiệm truy vấn dữ liệu, hiển thị, duyệt cảnh báo, và thực hiện các thao tác quản trị như chặn IP. Cách tách này giúp hệ thống dễ bảo trì, giảm ảnh hưởng lẫn nhau giữa xử lý nặng theo lịch và truy cập tương tác của người dùng. Các khối chức năng trong kiến trúc gồm:

- Khối thu thập dữ liệu nhận log Zeek theo ngày hoặc theo lịch định kỳ. Nhiệm vụ chính là xác định đúng phạm vi thời gian cần xử lý, đọc log, và đưa dữ liệu thô sang bước chuẩn hoá. Khối này cần đảm bảo tính liên tục, tránh đọc trùng hoặc bỏ sót khi chạy theo lịch.
- Khối xử lý, chuẩn hoá và tạo chuỗi chuyển dữ liệu thô thành dạng chuỗi sự kiện phù hợp cho mô hình. Các bước điển hình gồm chuẩn hoá thời gian, tạo log key, ánh xạ qua codebook để thành event id, nhóm theo subject và time bucket, rồi tạo chuỗi độ dài chuẩn bằng padding hoặc cửa sổ trượt. Khối này quyết định tính nhất quán giữa huấn luyện và suy luận vì sử dụng cùng quy tắc tạo log key và codebook.
- Khối mô hình inference nhận chuỗi event id và sinh phân phối dự đoán theo từng bước thời gian, sau đó cung cấp dữ liệu trung gian cho bước tổng hợp cảnh báo. Khối này được triển khai tách rời khỏi giao diện để đảm bảo chạy ổn định theo lịch và không bị ảnh hưởng bởi tải truy cập người dùng.
- Khối tổng hợp cảnh báo và lưu trữ thực hiện tính chỉ số bất thường, tổng hợp theo subject, tạo bản ghi cảnh báo, và ghi xuống cơ sở dữ liệu. Đây là khối biến đầu ra mô hình thành dữ liệu vận hành, ưu tiên lưu các thống kê gọn thay vì lưu toàn bộ phân phối dự đoán để tiết kiệm dung lượng.
- Khối lưu trữ, CSDL lưu các bảng cảnh báo, bảng flow hoặc nhật ký liên quan, bảng người dùng và phân quyền, bảng danh sách IP chặn, và bảng audit thao tác. CSDL đóng vai trò trung tâm chia sẻ dữ liệu giữa luồng xử lý theo lịch và luồng truy vấn qua API.
- Backend API Node.js cung cấp endpoint cho giao diện. Các chức năng gồm xác thực, truy vấn danh sách cảnh báo theo ngày và theo bộ lọc, trả chi tiết cảnh báo, cập nhật trạng thái duyệt, quản lý danh sách IP chặn, và quản trị tài khoản khi có quyền. Backend đồng thời ghi audit cho các thao tác nhạy cảm.
- Giao diện Web GUI cung cấp các màn hình điều hướng theo nhóm chức năng, hiển thị danh sách cảnh báo, lọc theo ngày và trạng thái duyệt, xem chi tiết, thao tác duyệt và quản trị. Giao diện không xử lý nặng về suy luận mà chỉ gọi API và hiển thị kết quả.
- Khối phản ứng, chặn IP là tuỳ chọn, thực thi ở tầng firewall. Khi được bật, hệ thống đồng bộ danh sách IP hiệu lực để áp dụng luật chặn. Việc chặn cần có kiểm soát quyền và có ghi nhận lịch sử để giảm rủi ro vận hành.

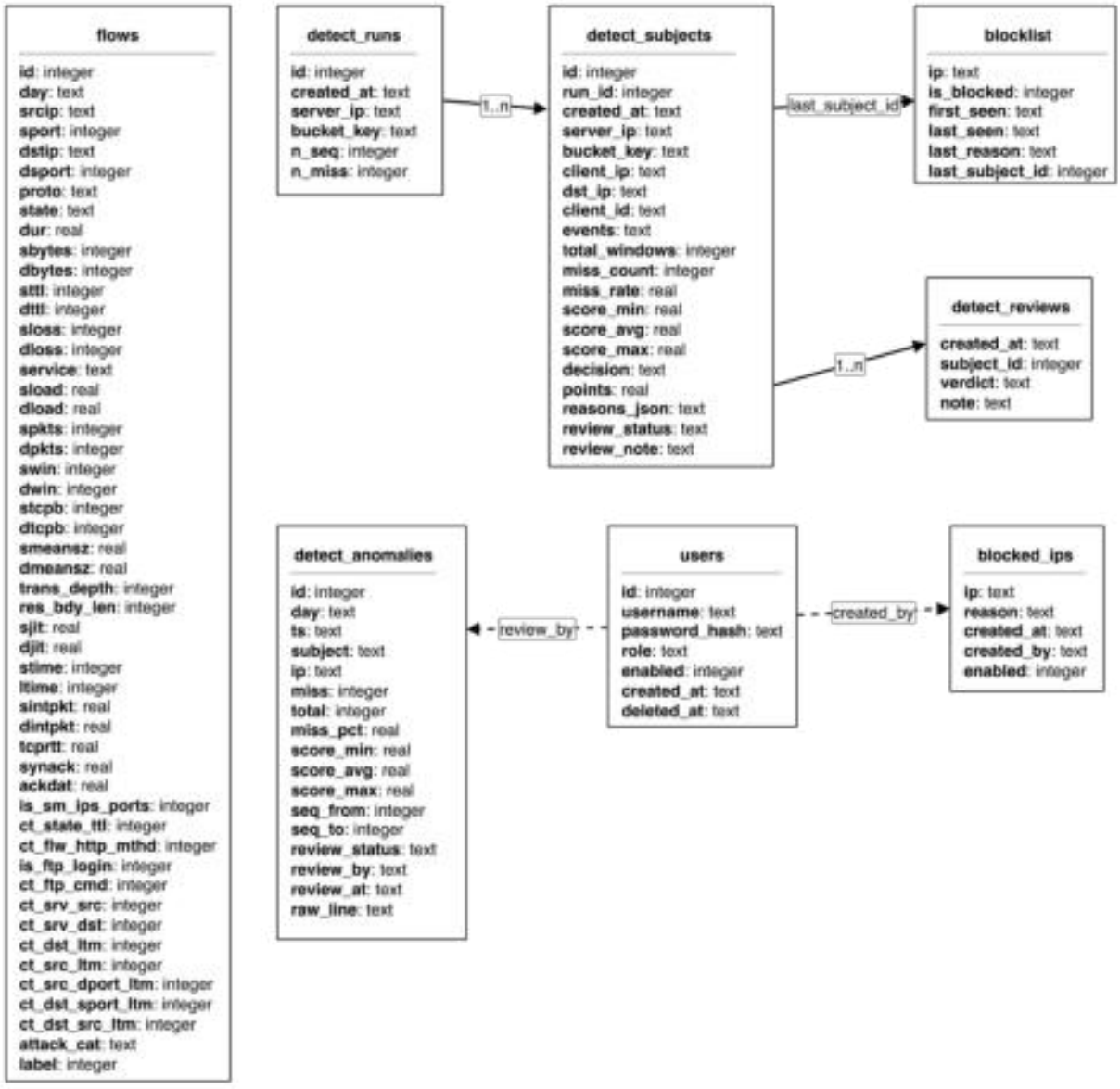
3.4 Thiết kế dữ liệu

3.4.1 Mục tiêu thiết kế dữ liệu

Thiết kế dữ liệu tập trung vào ba mục tiêu chính:

- Thứ nhất, lưu được cảnh báo và trạng thái xử lý để phục vụ vận hành theo ngày và theo người duyệt.
- Thứ hai, hỗ trợ truy vấn nhanh cho giao diện, đặc biệt là lọc theo ngày, theo trạng thái review, theo IP, và mở chi tiết bản ghi.
- Thứ ba, đảm bảo truy vết được nguồn gốc dữ liệu, gồm thời gian, subject, và các chỉ số tổng hợp đã sinh ra cảnh báo.

3.4.2 Chi tiết các bảng cơ sở dữ liệu



Hình 3.3: Chi tiết các bảng trong cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống sẽ gồm các bảng sau:

- Bảng users: quản lý tài khoản đăng nhập và phân quyền.

Bảng 3.7: Bảng Users.

Trường	Kiểu	Ý nghĩa
id	integer	khoá chính
username	text	duy nhất
password_hash	text	mật khẩu đã băm
role	text	admin hoặc user
enabled	integer	1 hoạt động, 0 vô hiệu
created_at	text	thời điểm tạo
deleted_at	text	thời điểm xoá mềm

- Bảng blocked_ips: quản lý danh sách IP chặn thủ công trên giao diện.

Bảng 3.8: Bảng blocked_ips.

Trường	Kiểu	Ý nghĩa
ip	text	khoá chính
reason	text	lý do chặn
created_at	text	thời điểm thêm vào danh sách
created_by	text	người thực hiện
enabled	integer	1 đang chặn, 0 tạm gỡ hoặc tắt

- Bảng blocklist: lưu trạng thái chặn theo luồng phát hiện và review, phục vụ tự động hoá bật tắt theo kết luận.

Bảng 3.9: Bảng blocklist.

Trường	Kiểu	Ý nghĩa
ip	text	khoá định danh IP
is_blocked	integer	1 đang chặn, 0 không chặn
first_seen	text	thời điểm ghi nhận lần đầu
last_seen	text	thời điểm cập nhật gần nhất
last_reason	text	lý do cập nhật trạng thái gần nhất
last_subject_id	integer	id subject gần nhất liên quan đến quyết định

- Bảng detect_runs: lưu thông tin mỗi lần chạy phát hiện tấn công theo lịch.

Bảng 3.10: Bảng detect_runs.

Trường	**Kiểu**	**Ý nghĩa**
id	integer	khoá chính
created_at	text	thời điểm chạy
server_ip	text	máy thực thi hoặc máy giám sát
bucket_key	text	khoá bucket thời gian
n_seq	integer	số chuỗi được xử lý
n_miss	integer	số chuỗi hoặc cửa sổ bị đánh dấu

- Bảng detect_subjects: lưu kết quả phát hiện tấn công theo từng subject trong một run.

Bảng 3.11: Bảng detect_subjects.

Trường	**Kiểu**	**Ý nghĩa**
id	integer	khoá chính
run_id	integer	liên kết tới detect_runs
created_at	text	thời điểm tạo bản ghi
server_ip	text	máy thực thi hoặc máy giám sát
bucket_key	text	khoá bucket thời gian
client_ip	text	IP nguồn
dst_ip	text	IP đích
client_id	text	định danh client nếu có
events	text	chuỗi sự kiện hoặc thống kê sự kiện
total_windows	integer	tổng số cửa sổ
miss_count	integer	số cửa sổ bị đánh dấu
miss_rate	real	tỷ lệ bất thường
score_min	real	điểm nhỏ nhất
score_avg	real	điểm trung bình
score_max	real	điểm lớn nhất
decision	text	nhãn quyết định
points	real	điểm ưu tiên để sắp xếp
reasons_json	text	giải thích dạng json
review_status	text	pending, true_attack, false_positive
review_note	text	ghi chú duyệt

- Bảng detect_reviews: lưu lịch sử duyệt và ghi chú cho các subject phát hiện tấn công.

Bảng 3.12: Bảng detect_reviews.

Trường	**Kiểu**	**Ý nghĩa**
created_at	text	thời điểm duyệt
subject_id	integer	liên kết tới detect_subjects
verdict	text	pending, true_attack, false_positive
note	text	ghi chú duyệt

- Bảng detect_anomalies: lưu cảnh báo bất thường theo ngày để tra cứu và duyệt.

Bảng 3.13: Bảng detect_anomalies.

Trường	**Kiểu**	**Ý nghĩa**
id	integer	khoá chính
day	text	ngày dạng YYYY-MM-DD
ts	text	thời điểm đại diện của cảnh báo
subject	text	khoá nhóm, thường gồm ip và bucket
ip	text	IP liên quan
miss	integer	số bước bị đánh dấu
total	integer	tổng số bước
miss_pct	real	tỷ lệ phần trăm
score_min	real	điểm nhỏ nhất
score_avg	real	điểm trung bình
score_max	real	điểm lớn nhất
seq_from	integer	chỉ số bắt đầu trong chuỗi
seq_to	integer	chỉ số kết thúc trong chuỗi
review_status	text	pending, true_attack, false_positive
review_by	text	người duyệt
review_at	text	thời điểm duyệt
raw_line	text	dòng gốc để hỗ trợ tìm kiếm

- Bảng flows: lưu dữ liệu flow theo ngày để tra cứu, lọc và thống kê trên giao diện.

Bảng 3.14: Bảng flows.

Trường	**Kiểu**	**Ý nghĩa**
id	integer	khoá chính
day	text	ngày dạng YYYY-MM-DD
srcip	text	IP nguồn

sport	integer	cổng nguồn
dstip	text	IP đích
dsport	integer	cổng đích
proto	text	giao thức
state	text	trạng thái kết nối
dur	real	thời lượng
sbytes	integer	byte từ nguồn
dbytes	integer	byte tới đích
sttl	integer	ttl nguồn
dttl	integer	ttl đích
sloss	integer	loss phía nguồn
dloss	integer	loss phía đích
service	text	dịch vụ
sload	real	tải phía nguồn
dload	real	tải phía đích
spkts	integer	gói phía nguồn
dpkts	integer	gói phía đích
swin	integer	tcp window nguồn
dwin	integer	tcp window đích
stcpb	integer	tcp base seq nguồn
dtcpb	integer	tcp base seq đích
smeansz	real	kích thước gói trung bình nguồn
dmeansz	real	kích thước gói trung bình đích
trans_depth	integer	độ sâu giao thức ứng dụng
res_bdy_len	integer	độ dài body phản hồi
sjit	real	jitter nguồn
djit	real	jitter đích
stime	integer	thời gian bắt đầu epoch
ltime	integer	thời gian kết thúc epoch
sintpkt	real	khoảng cách gói nguồn
dintpkt	real	khoảng cách gói đích
tcprtt	real	rtt
synack	real	synack
ackdat	real	ackdat
is_sm_ips_ports	integer	cờ đặc trưng

ct_state_ttl	integer	thống kê trạng thái ttl
ct_flw_http_mthd	integer	thống kê http method
is_ftp_login	integer	cờ ftp login
ct_ftp_cmd	integer	thống kê ftp cmd
ct_srv_src	integer	thống kê service theo src
ct_srv_dst	integer	thống kê service theo dst
ct_dst_ltm	integer	thống kê dst ltm
ct_src_ltm	integer	thống kê src ltm
ct_src_dport_ltm	integer	thống kê src dport ltm
ct_dst_sport_ltm	integer	thống kê dst sport ltm
ct_dst_src_ltm	integer	thống kê dst src ltm
attack_cat	text	nhóm tấn công nếu có
label	integer	nhãn 0 hoặc 1

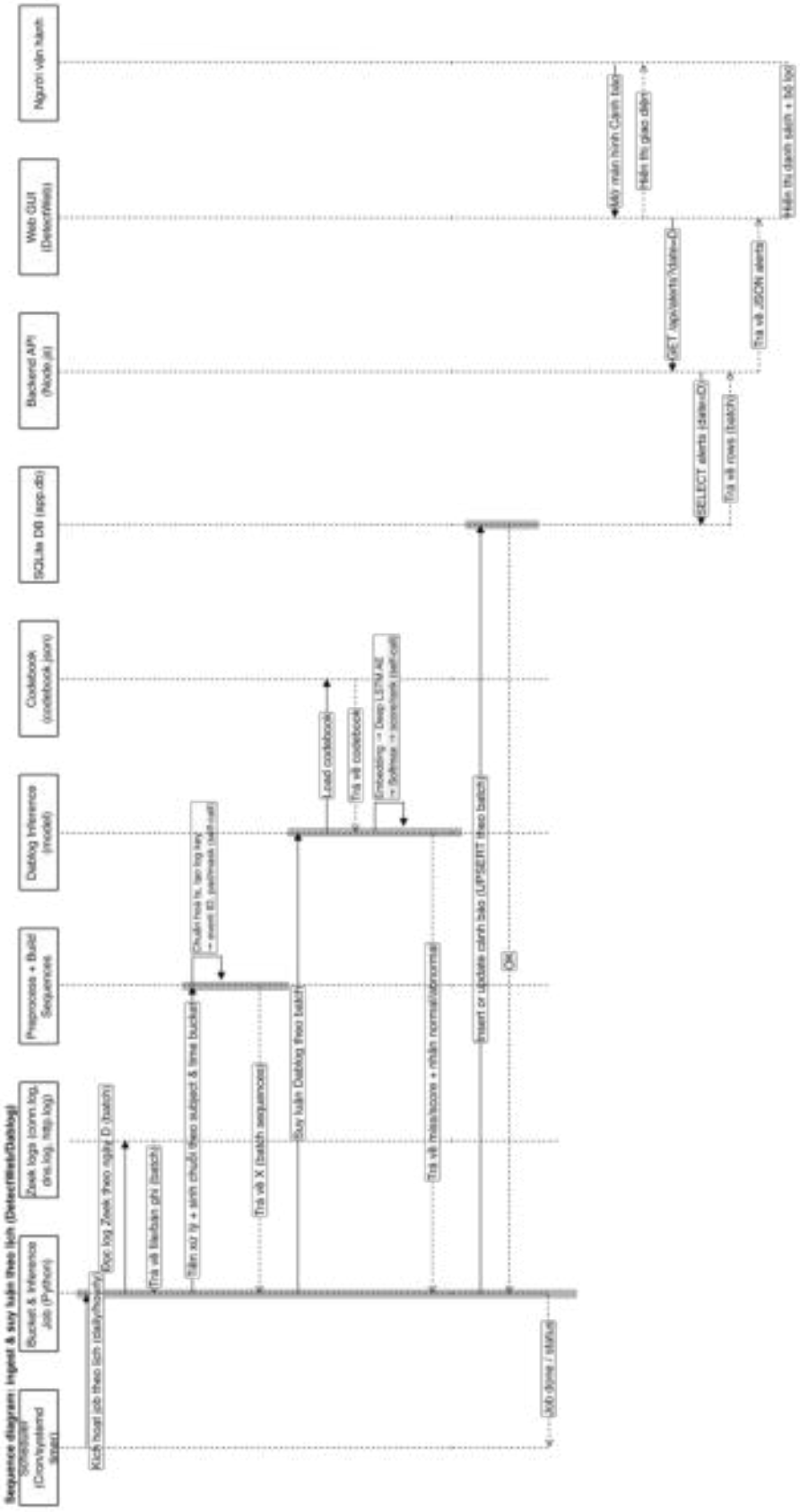
3.5 Thiết kế luồng xử lý

3.5.1 DFD mức ngữ cảnh và mức 1

Phần này mô tả luồng xử lý dữ liệu của hệ thống theo hai mức. Mức ngữ cảnh giúp nhìn tổng quan hệ thống nhận dữ liệu gì và trả ra gì. Mức 1 đi sâu các tiến trình chính theo kiến trúc đã chốt, gồm thu thập log, chuẩn hoá tạo chuỗi, suy luận, tổng hợp cảnh báo, lưu CSDL, và phục vụ truy vấn trên GUI. Luồng dữ liệu tổng quan của hệ thống được mô tả bằng DFD mức ngữ cảnh.

3.5.2 Sequence diagram cho luồng ingest và suy luận theo lịch

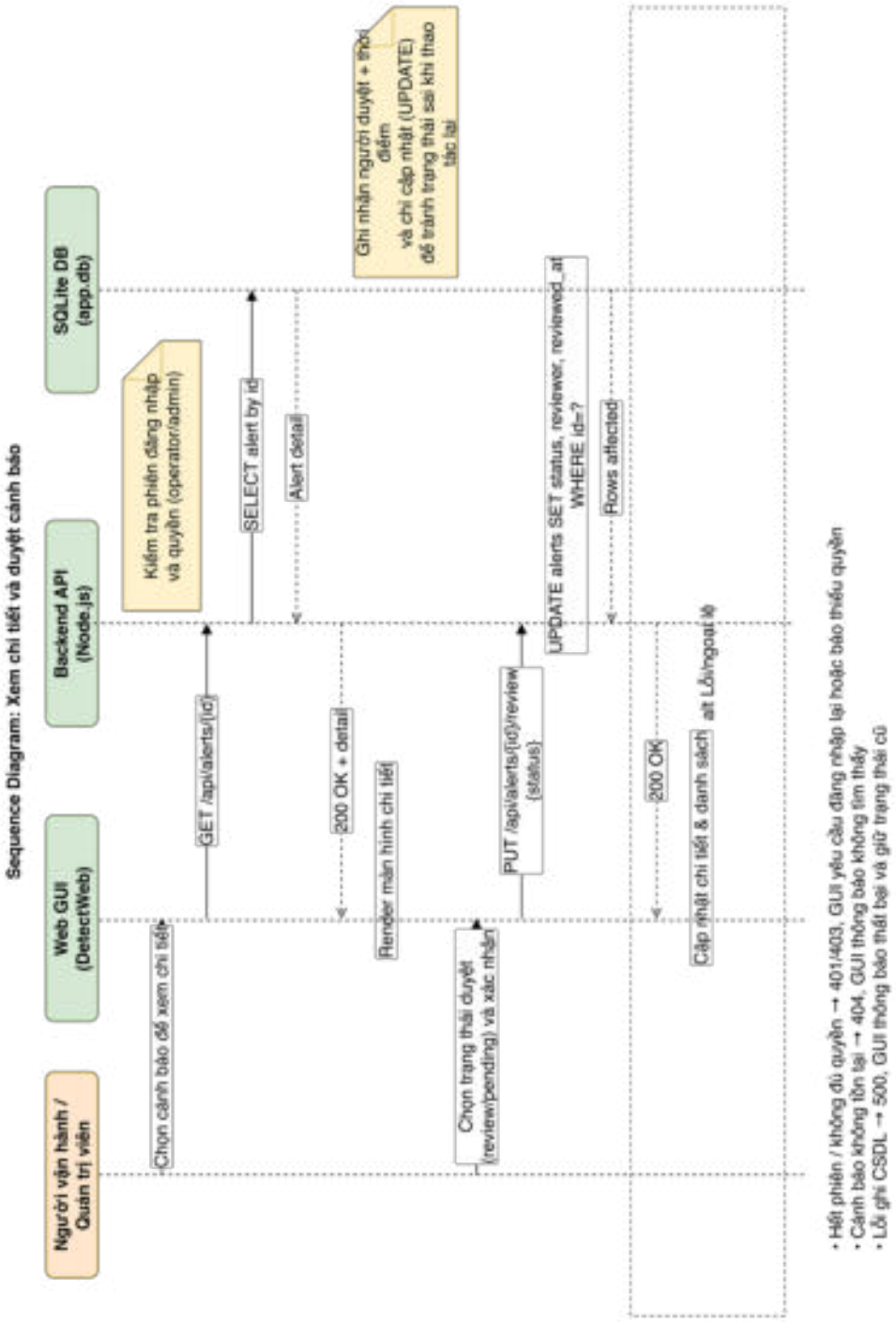
Luồng này mô tả cách hệ thống tự động xử lý dữ liệu theo ngày hoặc theo mốc định kỳ. Trọng tâm là thứ tự các bước từ đọc log, chuẩn hoá và tạo chuỗi, chạy inference, tổng hợp cảnh báo, đến ghi kết quả vào CSDL để giao diện truy vấn được.



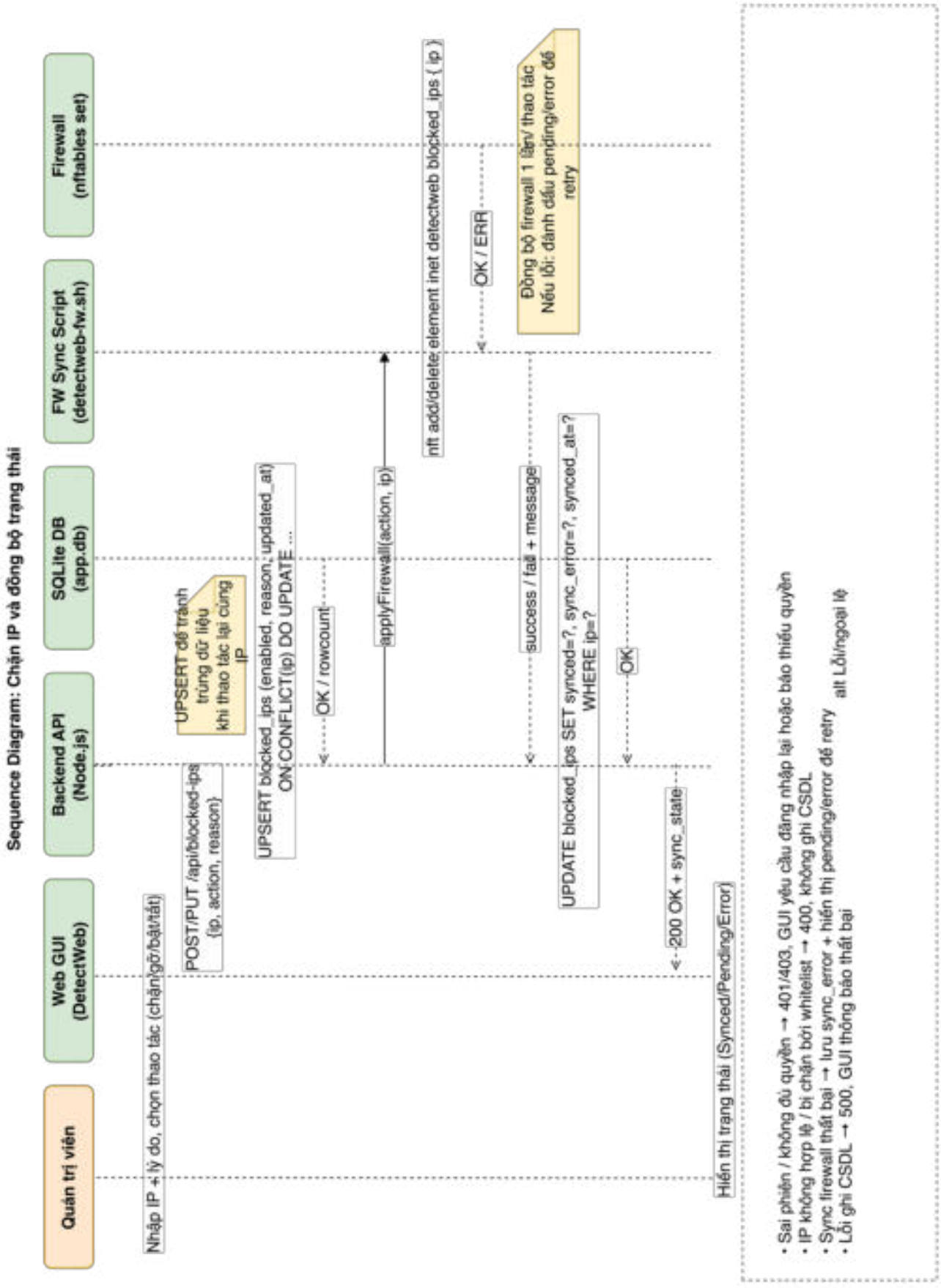
Hình 3.4: Sequence diagram ingest và suy luận theo lịch.

3.5.3 Sequence diagram cho luồng duyệt cảnh báo và chặn IP

Luồng này mô tả thao tác của người dùng trên giao diện, tập trung vào hai hành động quan trọng. Một là duyệt cảnh báo để cập nhật trạng thái xử lý. Hai là chặn hoặc gỡ chặn IP khi có kết luận phù hợp. Trình tự được mô tả theo vai trò quản trị viên vì đây là vai trò có đầy đủ quyền, với người vận hành thường chỉ thực hiện đến bước duyệt cảnh báo.



Hình 3.5: Sequence diagram duyệt cảnh báo.

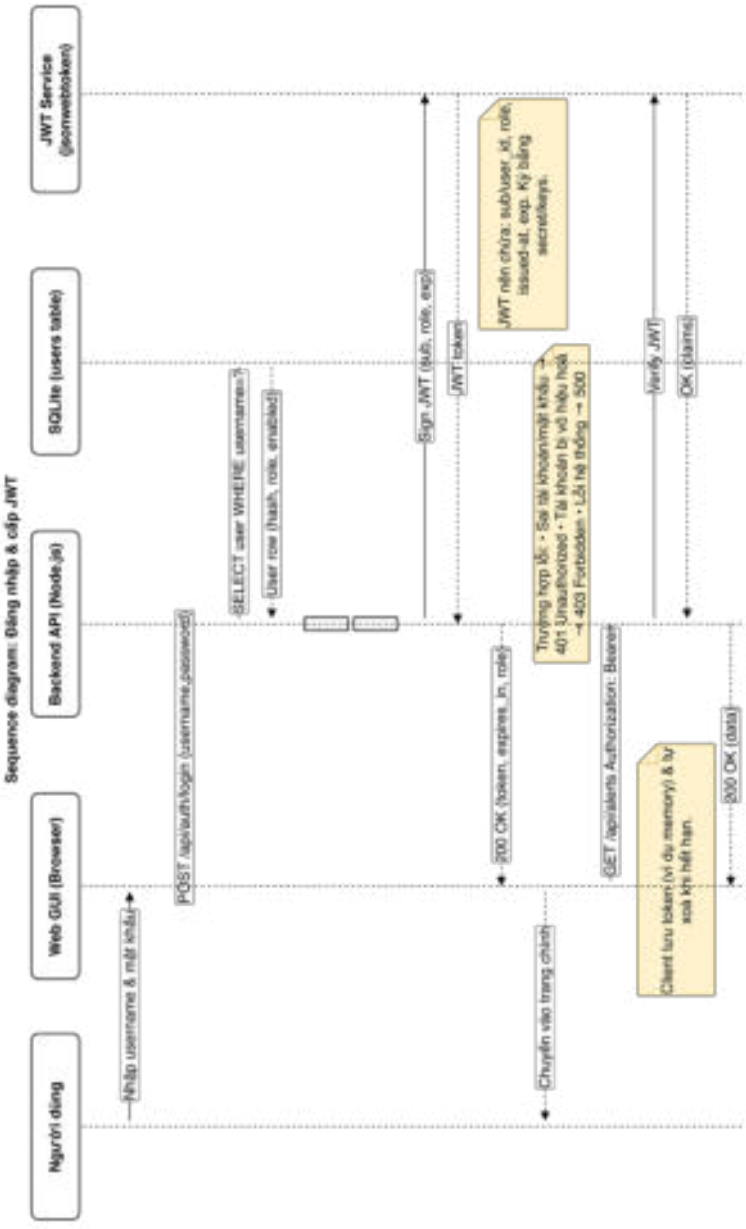


Hình 3.6: Sequence diagram chặn IP và đồng bộ trạng thái.

3.6 Thiết kế bảo mật và phân quyền

3.6.1 Xác thực người dùng

Cơ chế xác thực sử dụng mô hình đăng nhập bằng username và password, sau khi xác thực thành công sẽ cấp một JWT để dùng cho các yêu cầu tiếp theo. Backend kiểm tra thông tin đăng nhập trong bảng users, so sánh mật khẩu bằng bcrypt, sau đó tạo JWT chứa id, username, role, và thời hạn hiệu lực 12 giờ. Mọi API nghiệp vụ đều yêu cầu header Authorization theo dạng Bearer token, nếu thiếu hoặc token không hợp lệ thì trả về lỗi 401.



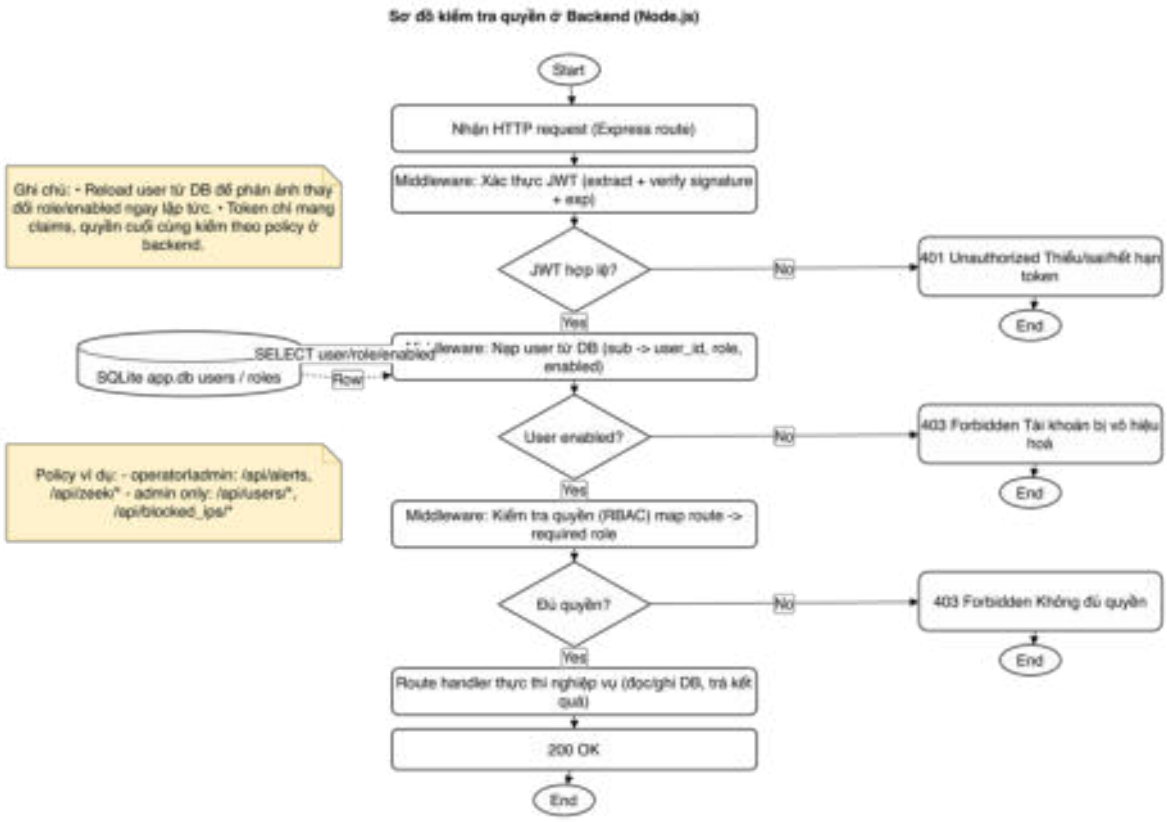
Hình 3.7: Sequence đăng nhập và cấp JWT.

3.6.2 Quản lý phiên và vòng đời token

Phiên làm việc được quản lý theo hướng stateless. Token là bằng chứng xác thực, không cần lưu trạng thái phiên phía server. Khi token hết hạn hoặc bị sai chữ ký, middleware xác thực trả về 401 và giao diện chuyển người dùng về trang đăng nhập. Cách làm này đơn giản, phù hợp hệ thống chạy một máy chủ, và giảm rủi ro phiên bị kẹt do lưu trạng thái. Trong thiết kế triển khai, khoá bí mật JWT được cấu hình bằng biến môi trường JWT_SECRET để tránh hardcode. Mật khẩu quản trị mặc định cũng được đặt qua biến môi trường để bắt buộc đổi sau lần khởi tạo đầu.

3.6.3 Phân quyền theo vai trò

Phân quyền được thiết kế theo RBAC với hai vai trò, admin và user. Middleware authRequired đảm bảo mọi API nghiệp vụ đều có token hợp lệ. Middleware adminOnly kiểm soát các thao tác nhạy cảm, gồm quản lý tài khoản, thao tác chặn IP, và các API quản trị. Phân quyền được thiết kế theo RBAC với hai vai trò, admin và user.



Hình 3.8: Sơ đồ kiểm tra quyền ở backend.

Bảng 3.15: Phân quyền theo vai trò.

Nhóm chức năng	user	admin
Xem danh sách cảnh báo tấn công	có	có
Xem danh sách cảnh báo bất thường	có	có
Duyệt cảnh báo, cập nhật review status	có	có
Xem danh sách IP đã chặn	có	có
Thêm IP chặn, bật tắt chặn	không	có
Quản lý tài khoản, tạo user, khoá user, xoá mềm user	không	có
Import dữ liệu quản trị, ví dụ import zeekcsv vào bảng flows	không	có
Xem log Zeek theo ngày, xem tail file	có	có

3.6.4 Các biện pháp bảo mật bổ sung ở mức thiết kế

Phần này tổng hợp các điểm tăng cường để hệ thống vận hành an toàn hơn khi triển khai thực tế:

- Giới hạn bề mặt tấn công trên API: cấu hình CORS theo danh sách origin tin cậy thay vì mở rộng mặc định. Giới hạn kích thước JSON request như hiện tại để giảm rủi ro gửi payload lớn.
- Giảm rủi ro dò mật khẩu: bổ sung rate limit cho API đăng nhập, và cơ chế tăng thời gian chờ khi đăng nhập sai nhiều lần. Kết hợp chính sách mật khẩu đủ dài, và bắt buộc đổi mật khẩu quản trị mặc định sau khi khởi tạo.
- An toàn truy cập file log: các API đọc file đã có kiểm tra định dạng ngày và chặn path traversal bằng cơ chế join an toàn, điều này cần được giữ như nguyên tắc bắt buộc. Chỉ cho phép đọc các file có đuôi hợp lệ và nằm trong thư mục log theo ngày.
- Bảo vệ token phía trình duyệt: ưu tiên triển khai HTTPS ở tầng reverse proxy để bảo vệ token khi truyền trên mạng. Khi hiển thị dữ liệu JSON trên giao diện, cần escape nội dung trước khi render HTML để giảm rủi ro XSS, đặc biệt khi có trường dữ liệu đến từ log.

CHƯƠNG 4: XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG

4.1 Công nghệ sử dụng và môi trường triển khai

4.1.1 Hạ tầng triển khai hệ thống

Hệ thống được triển khai trên máy chủ đám mây AWS EC2 chạy Ubuntu Server. Backend đặt trong thư mục ~/detectweb_backend, cơ sở dữ liệu SQLite lưu tại file app.db, giao diện Web dạng static đặt trong /var/www/detectweb_gui. Luồng dữ liệu đầu vào lấy từ Zeek logs theo cấu trúc thư mục năm, tháng, ngày, và từ thư mục buckets của bộ xuất UNSW theo ngày. Các đường dẫn dữ liệu được cấu hình qua biến môi trường trong backend:

- ZEEK_ROOT trỏ đến thư mục chứa log Zeek, mặc định /home/ubuntu/zeek_logs.
- UNSW_BUCKETS trỏ đến thư mục buckets dạng CSV, mặc định /home/ubuntu/unsw_exporter/buckets.
- BASE_PATH hỗ trợ triển khai dưới subpath khi đặt sau reverse proxy, ví dụ /detectweb.
- JWT_SECRET, ADMIN_USER, ADMIN_PASSWORD phục vụ bảo mật và khởi tạo tài khoản quản trị ban đầu.

4.1.2 Môi trường vận hành dịch vụ

Backend được xây dựng bằng Node.js và Express, chạy như một dịch vụ nền để đảm bảo tự khởi động lại khi máy chủ reboot. Các truy vấn dữ liệu và thao tác duyệt, chặn IP đều thực hiện qua HTTP API. Hệ thống có cơ chế cập nhật gần thời gian thực cho phần xem log bằng SSE, backend theo dõi thay đổi file bằng chokidar rồi đẩy phần nội dung mới xuống trình duyệt.

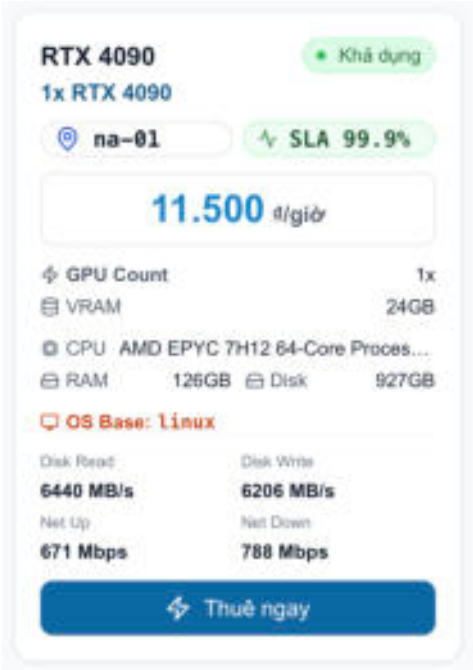
Bảng 4.1: Cấu hình vận hành chính.

Thành phần	**Môi trường chạy**	**Vai trò**
Node.js backend	EC2 Ubuntu	cung cấp API, đọc log, truy vấn DB, review, chặn IP, SSE
SQLite app.db	cùng máy EC2	lưu users, blocked_ips, flows, detect_anomalies, detect_subjects, detect_runs, detect_reviews, blocklist

Web GUI static	cùng máy EC2	hiển thị, lọc theo ngày, thao tác review, gọi API
Zeek logs	thư mục theo ngày	dữ liệu gốc phục vụ xem log và đối soát
UNSW buckets CSV	thư mục buckets	dữ liệu flow theo ngày phục vụ tra cứu và import vào DB

4.1.3 Môi trường huấn luyện mô hình học sâu

Quá trình huấn luyện mô hình được thực hiện offline trên máy chủ GPU riêng, tách khỏi máy triển khai vận hành để tránh ảnh hưởng hiệu năng giám sát. Cấu hình phần cứng dùng để train gồm 1 GPU RTX 4090 với 24GB VRAM, CPU AMD EPYC 7H12 64 core, RAM 126GB, và dung lượng đĩa 927GB. Môi trường phần mềm huấn luyện sử dụng Python, TensorFlow và Keras, kết hợp spaCy để xử lý embedding theo thiết kế mô hình, artefact mô hình được lưu dưới dạng file .keras sau khi huấn luyện.



Hình 4.1: Thông tin cấu hình máy dùng để huấn luyện thuê tại EZYCLOUDX. [10]

4.1.4 Công nghệ và thư viện chính

Các thành phần phần mềm được lựa chọn theo tiêu chí gọn nhẹ, dễ triển khai, và đủ cho vận hành trên một máy chủ. Các công nghệ và thư viện chính được sử dụng được liệt lê dưới bảng.

Bảng 4.2: Công nghệ và thư viện sử dụng.

Nhóm	Công nghệ	Mục đích
Thu thập log mạng	Zeek	giám sát lưu lượng và sinh log mạng phục vụ phân tích
Backend	Express	xây dựng HTTP API
Bảo mật HTTP	Helmet, CORS	giảm rủi ro cấu hình web, kiểm soát truy cập
Log hệ thống	Morgan	ghi log truy cập API
Xác thực	JWT, bcryptjs	đăng nhập, phân quyền theo vai trò
CSDL	better sqlite3	truy vấn nhanh, đơn giản khi triển khai một máy
Theo dõi file	chokidar, SSE	cập nhật log gần thời gian thực trên GUI
Huấn luyện mô hình	TensorFlow, Keras	xây dựng và train LSTM autoencoder, softmax theo bước
NLP embedding	spaCy	hỗ trợ embedding theo thiết kế mô hình

4.2 Xây dựng khối thu thập và truy xuất log Zeek

4.2.1 Nguồn dữ liệu đầu vào từ Zeek

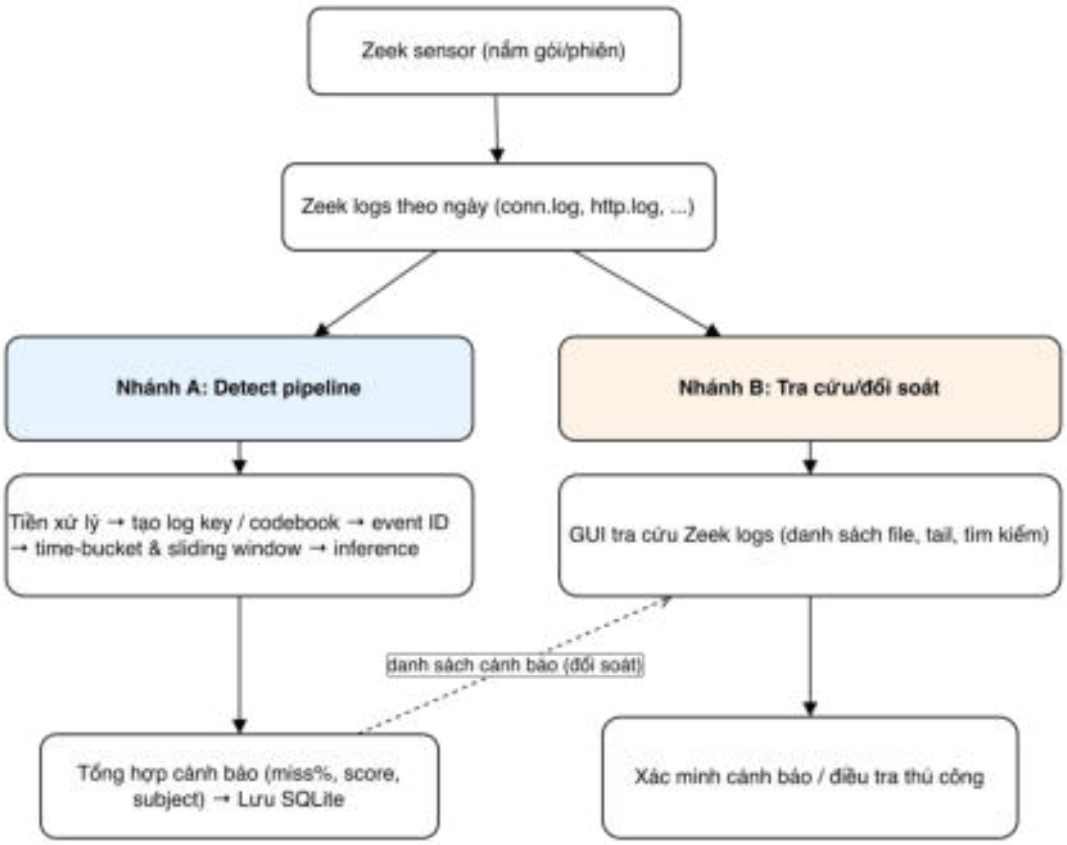
Hệ thống sử dụng Zeek như thành phần thu thập dữ liệu mạng ở lớp cảm biến. Zeek quan sát lưu lượng, trích xuất các sự kiện mạng và ghi ra log theo cấu trúc chuẩn, đảm bảo dữ liệu có thể truy vết lại theo thời gian và theo từng loại hoạt động. Trong quá trình vận hành, log Zeek đóng vai trò là dữ liệu nền. Dữ liệu này phản ánh trạng thái mạng theo từng phiên kết nối, từng truy vấn, và từng giao thức, giúp tái hiện bối cảnh khi có nghi vấn bất thường.

Về mặt tổ chức, log Zeek thường được ghi theo nhiều loại tệp, mỗi loại phản ánh một góc nhìn của mạng. Ví dụ conn log mô tả kết nối tổng quát, dns log mô tả truy vấn tên miền, http log mô tả giao dịch web. Từ góc độ thiết kế hệ thống, việc duy trì các tệp log theo loại cho phép chọn đúng nguồn khi điều tra, đồng thời cho phép hệ thống mở rộng dần phạm vi phân tích. Ban đầu có thể ưu tiên conn log để bao quát lưu lượng, sau đó bổ sung thêm log ứng dụng khi cần tăng độ giải thích cho cảnh báo.

Log Zeek trong hệ thống được dùng cho hai mục tiêu. Thứ nhất là tra cứu, đối soát khi có cảnh báo, nhằm xác minh nhanh diễn biến liên quan đến IP và thời điểm bị đánh dấu, từ đó phân loại true attack hoặc false positive. Thứ hai là dùng làm đầu vào

cho detect, tức dữ liệu Zeek sau tiền xử lý và chuẩn hoá sẽ được biến đổi thành chuỗi sự kiện để chạy suy luận và sinh cảnh báo.

Thiết kế này đảm bảo vòng lặp vận hành khép kín. Cùng một nguồn dữ liệu vừa phục vụ phát hiện tự động, vừa phục vụ xác minh thủ công. Điều này giúp tăng tính nhất quán, giảm tranh cãi về nguồn gốc cảnh báo, và hỗ trợ đánh giá chất lượng mô hình theo dữ liệu thực tế.



Hình 4.2: Minh hoạ vai trò kép của log Zeek, vừa là đầu vào detect vừa là nguồn tra cứu.

4.2.2 Quy ước tổ chức thư mục log theo ngày

Để phục vụ vận hành theo ngày và giảm rủi ro truy cập nhầm phạm vi, log Zeek được tổ chức theo cấu trúc thư mục phân cấp theo thời gian. Cách tổ chức theo năm, tháng, ngày giúp ba việc. Một là đồng bộ trực tiếp với bộ lọc ngày trên giao diện. Hai là giới hạn khối lượng dữ liệu mỗi lần truy vấn, vì người vận hành thường làm việc theo từng ngày. Ba là hỗ trợ lưu trữ lâu dài và sao lưu theo từng đơn vị thời gian, thuận lợi khi cần đối soát lại sự cố trong quá khứ. Trong triển khai, thư mục gốc chứa log được cấu hình bằng biến môi trường ZEEK_ROOT. Bên trong thư mục này, hệ thống tạo các

thư mục con theo năm, tháng, ngày. Ví dụ với ngày 2026-01-16, đường dẫn dữ liệu tương ứng có dạng. ZEEK_ROOT/2026/01/16/.

Hình 4.3: Minh hoạ về các file dữ liệu mà zeek đã trích xuất.

Bên trong thư mục ngày là các tệp log Zeek theo loại, ví dụ conn log, dns log, http log. Các file có thể ở định dạng TSV mặc định của Zeek hoặc dạng JSON nếu cấu hình như vậy, tuy nhiên phần truy xuất file của hệ thống coi log như văn bản dòng, do đó có thể đọc tail theo số dòng để hiển thị nhanh.

Việc tổ chức theo ngày được ràng buộc trực tiếp với cơ chế chọn ngày trên GUI. Khi người dùng chọn năm, tháng, ngày, backend sẽ ánh xạ lựa chọn này sang đúng đường dẫn thư mục ngày để liệt kê danh sách file. Trường hợp chọn tháng là ALL hoặc ngày là ALL, backend không đọc toàn bộ file nội dung, mà chỉ dùng để phục vụ danh sách ngày và tháng, nhằm tránh tải quá lớn và tránh thao tác vượt phạm vi.

Một yêu cầu quan trọng trong quy ước thư mục là tính ổn định. Khi hệ thống detect chạy theo lịch, nó cần biết chính xác thư mục log của ngày cần xử lý. Do đó, thư mục ngày cần được tạo đúng chuẩn, và Zeek phải ghi dữ liệu vào đúng thư mục. Khi có cơ chế xoay vòng hoặc lưu trữ, cần đảm bảo dữ liệu không bị di chuyển sang nơi khác trước khi pipeline detect hoàn thành ingest của ngày đó.

Để hỗ trợ kiểm soát truy cập file, backend chỉ cho phép truy xuất file nằm trong phạm vi ZEEK_ROOT và chỉ cho phép tên file dạng basename. Điều này giúp ngăn việc truyền tham số đường dẫn bất hợp lệ, đồng thời đảm bảo các API xem log chỉ truy cập được dữ liệu Zeek theo ngày, không thể truy cập tuỳ ý ra các thư mục khác trên máy chủ.

4.2.3 Thiết kế backend cho truy xuất danh sách và nội dung log

Khối truy xuất log được thiết kế theo nguyên tắc tách nhỏ theo nhu cầu vận hành. Người dùng thường không cần tải toàn bộ file log vì dung lượng lớn và ít phù hợp để đọc trực tiếp trên trình duyệt. Thay vào đó, hệ thống ưu tiên hai thao tác phổ biến. Một là liệt kê đúng các file log có trong ngày để người dùng chọn loại log cần xem. Hai là đọc nhanh một phần nội dung theo kiểu tail để quan sát các dòng gần thời điểm hiện tại hoặc gần thời điểm cảnh báo.

Về mặt triển khai, backend đóng vai trò cầu nối giữa cấu trúc thư mục Zeek và giao diện. Backend nhận tham số ngày từ bộ lọc, ánh xạ sang thư mục ngày trong ZEEK_ROOT, sau đó thực hiện thao tác hệ thống tệp và trả về dữ liệu dạng JSON. Cách tiếp cận này có ba lợi ích. Thứ nhất, giao diện không cần biết đường dẫn nội bộ của máy chủ. Thứ hai, backend có thể áp dụng kiểm tra hợp lệ đầu vào và giới hạn phạm vi truy cập. Thứ ba, có thể mở rộng thêm cơ chế cache và giới hạn tốc độ nếu log lớn.

Nhóm API điều hướng thời gian, để bộ lọc ngày hoạt động ổn định và không phụ thuộc dữ liệu giả lập ở phía trình duyệt, backend cung cấp các API để liệt kê năm, tháng, ngày hiện có. Khi mở giao diện, client gọi API năm để biết các năm có log. Khi chọn một năm, client gọi API tháng. Khi chọn tháng, client gọi API ngày. Luồng này giúp bộ lọc luôn đồng bộ với dữ liệu thực tế trên đĩa.

Nhóm API liệt kê danh sách file trong ngày, sau khi người dùng chọn ngày, backend trả về danh sách file log trong thư mục ngày. Danh sách này dùng để tạo một menu chọn loại log. Việc tách API liệt kê file khỏi API đọc nội dung giúp giao diện tải nhanh hơn, đồng thời tránh việc backend phải đọc file khi người dùng chỉ mới cần danh sách. Trong thiết kế, danh sách file nên trả về các thuộc tính tối thiểu. Tên file và kích thước là đủ để người dùng ước lượng mức độ lớn nhỏ. Có thể bổ sung mtime để biết file đang cập nhật. Phần này không bắt buộc nhưng hữu ích khi điều tra sự cố theo thời gian thực.

Nhóm API đọc nội dung theo kiểu tail, đọc nội dung theo tail là thao tác trọng tâm vì phù hợp với việc đọc log trên web. Backend nhận tham số date, file, và limit dòng. Backend chỉ đọc phần cuối file và trả về N dòng cuối, đồng thời có thể trả thêm tổng số dòng hoặc offset để hỗ trợ chế độ theo dõi SSE ở các phần sau. Thiết kế tail theo dòng có ưu điểm là đơn giản và phù hợp với log dạng line based. Tuy nhiên cần lưu ý trường hợp dòng rất dài, backend cần có giới hạn kích thước đọc để tránh trả về payload quá lớn. Về phía giao diện, tail giúp người vận hành quan sát nhanh các sự kiện mới nhất và đối chiếu thời gian với cảnh báo.

Định dạng dữ liệu trả về, các API trả về JSON để giao diện render linh hoạt. Với API tail, ngoài mảng dòng, nên trả thêm thông tin file và ngày để giao diện đảm bảo không bị nhầm khi người dùng chuyển ngày. Khi không có dữ liệu, backend trả danh sách rỗng, giao diện hiển thị trạng thái không có dữ liệu thay vì báo lỗi.

Bảng 4.3: Danh sách API truy xuất log Zeek.

API	Mục đích	Tham số chính

GET /api/zeek/years	liệt kê năm có log	không
GET /api/zeek/months	liệt kê tháng theo năm	year
GET /api/zeek/days	liệt kê ngày theo tháng	year, month
GET /api/zeek/files	liệt kê file log theo ngày	date
GET /api/zeek/tail	đọc N dòng cuối của file log	date, file, limit

4.2.4 Cơ chế bảo vệ truy xuất file log

Truy xuất log Zeek qua API là thao tác đọc file trên máy chủ, vì vậy cần kiểm soát chặt để tránh đọc vượt phạm vi, tránh gây quá tải, và đảm bảo chỉ người dùng hợp lệ mới truy cập được. Thiết kế bảo vệ tập trung vào ba lớp, kiểm tra đầu vào, khoá phạm vi đường dẫn, và giới hạn tài nguyên:

- Kiểm soát đầu vào Backend chỉ chấp nhận tham số date đúng định dạng YYYY-MM-DD. Tham số file chỉ chấp nhận tên file dạng basename, không chứa ký tự phân tách thư mục hoặc chuỗi điều hướng. Đồng thời chỉ cho phép phần mở rộng phù hợp như .log hoặc .json. Các yêu cầu không hợp lệ bị từ chối trước khi truy cập hệ thống tệp.
- Khoá phạm vi trong ZEEK_ROOT: backend ghép ZEEK_ROOT với đường dẫn ngày và tên file, sau đó resolve đường dẫn tuyệt đối. Chỉ khi đường dẫn resolve nằm trong ZEEK_ROOT mới cho phép đọc. Nếu resolve ra ngoài phạm vi, backend trả lỗi và không đọc file. Lớp này ngăn path traversal ngay cả khi có đầu vào bị lách.
- Giới hạn tài nguyên và kiểm soát truy cập: API đọc nội dung dùng cơ chế tail theo số dòng, kèm giới hạn tối đa cho limit và giới hạn kích thước trả về để tránh payload lớn. Các endpoint truy xuất log yêu cầu JWT, chỉ người dùng đăng nhập mới gọi được. Có thể bổ sung rate limit cho các endpoint tail và SSE để giảm rủi ro bị gọi liên tục.
- An toàn khi hiển thị trên trình duyệt: nội dung log được render dạng text, tránh render HTML trực tiếp. Khi có tìm kiếm hoặc highlight, vẫn phải đảm bảo dùng textContent hoặc escape để tránh XSS.

Bảng 4.4: Tổng hợp rủi ro và biện pháp.

Rủi ro	**Biện pháp**
đọc file ngoài phạm vi	validate date, basename file, kiểm tra resolve nằm trong ZEEK_ROOT

quá tải do đọc nhiều	tail theo dòng, giới hạn limit và kích thước trả về, rate limit
truy cập trái phép	bắt buộc JWT, phân quyền theo vai trò nếu cần
XSS khi hiển thị	render text, escape khi highlight

4.2.5 Cập nhật gần thời gian thực bằng SSE

Để hỗ trợ theo dõi log đang tăng mà không phải tải lại trang liên tục, hệ thống sử dụng SSE để đẩy dữ liệu mới từ backend về trình duyệt. SSE phù hợp cho kịch bản log viewer vì kết nối một chiều từ server tới client, đơn giản hơn so với websocket, và hoạt động tốt với HTTP thông thường.

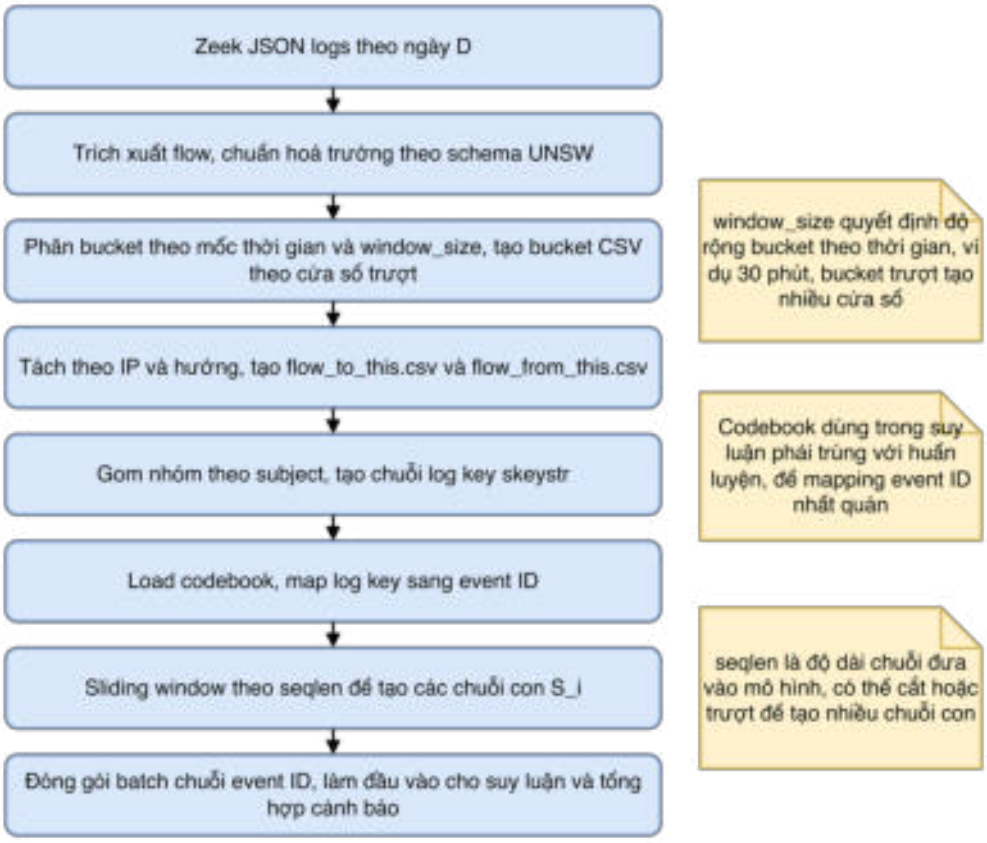
- Cách thiết lập luồng SSE: khi người dùng chọn ngày và chọn file log cần theo dõi, giao diện mở một kết nối SSE tới backend, kèm token xác thực. Backend giữ kết nối mở và trả về các event theo định dạng text event stream. Nếu token hết hạn hoặc không hợp lệ, backend đóng kết nối và giao diện quay về trạng thái yêu cầu đăng nhập lại.
- Cách phát hiện thay đổi file và lấy phần dữ liệu mới: Backend theo dõi file log bằng cơ chế watcher. Khi file thay đổi, backend không đọc lại toàn bộ file mà chỉ đọc phần mới dựa trên offset đã lưu. Dữ liệu mới được tách theo dòng, sau đó gửi về client theo từng gói nhỏ để tránh làm nghẽn trình duyệt khi log tăng nhanh. Thiết kế offset giúp hai việc. Một là giảm tải I O vì chỉ đọc phần tăng thêm. Hai là đảm bảo không gửi lặp dòng khi file thay đổi nhiều lần liên tiếp.
- Định dạng event và cách giao diện hiển thị: mỗi lần có dòng mới, backend gửi event append chứa danh sách dòng. Giao diện nhận event và nối thêm vào vùng hiển thị dạng pre. Khi người dùng đổi file hoặc đổi ngày, giao diện đóng kết nối SSE cũ và mở kết nối mới tương ứng, tránh trộn dữ liệu.

Để tránh tăng bộ nhớ khi theo dõi lâu, giao diện chỉ giữ lại một số lượng dòng gần nhất, ví dụ 2000 dòng, và tự loại bỏ các dòng cũ hơn.

4.3 Xây dựng module tiền xử lý và tạo chuỗi

4.3.1 Mục tiêu và vị trí của module trong luồng xử lý

Module tiền xử lý và tạo chuỗi có nhiệm vụ biến dữ liệu thô từ Zeek thành dữ liệu có cấu trúc tương thích với định dạng UNSW NB15, sau đó tổ chức lại thành chuỗi sự kiện theo từng đối tượng theo dõi để đưa vào bước suy luận mô hình. Dữ liệu đầu ra của module này phục vụ đồng thời hai nhu cầu, lưu lại các bản ghi dạng flow để tra cứu, và cung cấp chuỗi đầu vào ổn định cho quá trình detect.



Hình 4.4: Sơ đồ tổng quan module tiền xử lý và tạo chuỗi trong pipeline.

4.3.2 Trích xuất từ Zeek sang bản ghi flow theo định dạng UNSW

Nguồn dữ liệu gốc là các log lõi của Zeek như conn.log, http.log, ftp.log. Các log này có thể được cấu hình xuất JSON để dễ parse và xử lý dòng theo dòng. Chương trình trích xuất đọc tăng dần theo offset để tránh đọc lại toàn bộ file khi log liên tục tăng. Trạng thái offset được lưu trong một file state riêng, giúp chương trình có thể khởi động lại mà không mất tiến độ. Cách đọc tăng dần này cũng xử lý được tình huống log bị rotate hoặc bị truncate, khi kích thước file nhỏ hơn offset thì offset được reset. Ở bước lọc, chỉ giữ traffic inbound đi vào máy chủ. Địa chỉ IP máy chủ được lấy tự động từ default interface. Các bản ghi bị loại nếu là IPv6, multicast, broadcast, hoặc src là private. Ngoài ra các service nền như dhcp, arp, dns cũng được loại để giảm nhiễu và tập trung vào hành vi ứng dụng có rủi ro cao hơn. Ở bước ánh xạ trường, mỗi bản ghi conn được chuyển thành một dòng CSV theo danh sách trường của UNSW, trong đó các trường quan trọng được điền trực tiếp từ Zeek. Ví dụ srcip, sport, dstip, dsport, proto, state, service, spkts, dpkts, stime. Các trường liên quan HTTP và FTP được bổ sung bằng cách join theo uid, như trans_depth, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd. Nhãn label được đặt 0 để phản ánh dữ liệu thực tế chưa có nhãn tấn công.

Bảng 4.5: Ánh xạ trường chính từ Zeek sang UNSW.

Nguồn Zeek	Trường UNSW đích	Ý nghĩa
conn.id.orig_h	srcip	IP nguồn
conn.id.orig_p	sport	cổng nguồn
conn.id.resp_h	dstip	IP đích
conn.id.resp_p	dsport	cổng đích
conn.proto	proto	giao thức
conn.conn_state	state	trạng thái kết nối
conn.service	service	dịch vụ, đặt mặc định "-" nếu rỗng
conn.orig_pkts, conn.resp_pkts	spkts, dpkts	số gói hai chiều
conn.ts	stime	thời gian epoch giây
http.trans_depth	trans_depth	độ sâu giao dịch HTTP
http.method	ct_flw_http_mthd	có phương thức HTTP thì gán 1
ftp.user, ftp.password, ftp.reply_code	is_ftp_login	suy ra trạng thái login
ftp.command	ct_ftp_cmd	đếm số lệnh FTP theo uid

4.3.3 Cơ chế bucket hoá theo cửa sổ trượt

Sau khi tạo dòng CSV, dữ liệu được ghi vào thư mục buckets dưới dạng các file nhỏ theo mốc thời gian. Tên file mang định danh ngày giờ phút theo múi giờ Asia Ho Chi Minh, ví dụ unsw_YYYYMMDD_HHMM.csv. Cơ chế này giúp hai việc, giảm kích thước mỗi file để xử lý nhanh, và cho phép detect hoạt động theo cửa sổ trượt gần thời gian thực. Hệ thống chỉ giữ lại các bucket nằm trong cửa sổ gần nhất WINDOW_MINUTES phút. Các bucket cũ hơn sẽ bị xoá định kỳ bởi hàm prune. Nhờ đó dung lượng đĩa không tăng vô hạn và detect luôn tập trung vào dữ liệu mới.

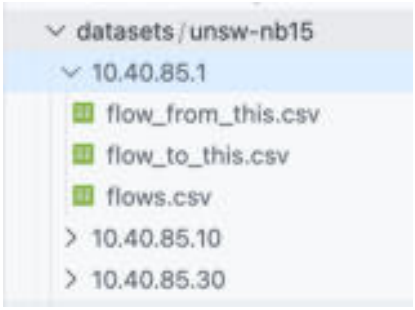
Hình 4.5: Minh hoạ bucket file theo dòng thời gian.

4.3.4 Tổ chức lại dữ liệu theo IP để tạo đầu vào detect

Dữ liệu bucket CSV sau đó được gom lại theo IP để tạo cấu trúc thư mục tương tự dữ liệu UNSW, tức mỗi IP có một thư mục riêng, bên trong có các file flow_from_this.csv và flow_to_this.csv. Cách tổ chức này phù hợp với cách mô hình

đọc dữ liệu, vì mỗi lần detect có thể tập trung vào các chuỗi liên quan đến một IP mục tiêu. Trong vận hành, chương trình auto runner chạy theo chu kỳ và gọi bước split trước khi gọi detect. Bước split tạo ra các file theo IP, bước detect đọc từ thư mục của IP máy chủ và file flow_to_this.csv, nghĩa là tập trung vào các flow inbound đi vào máy chủ.

Bảng 4.6: Cấu trúc thư mục sau khi split theo IP.



4.3.5 Tạo chuỗi sự kiện và định nghĩa subject

Từ các dòng flow, chuỗi sự kiện được xây dựng bằng cách chuyển mỗi dòng thành một khoá sự kiện skeystr. Khoá này được sinh từ hàm trích đặc trưng theo kịch bản logkeys và hệ số key_divisor, nhằm rời rạc hoá các giá trị và đưa về không gian biểu diễn phù hợp với codebook. Các sự kiện được gom nhóm theo subject để phản ánh ngữ cảnh theo thời gian và cặp giao tiếp. Subject được tạo từ srcip, dstip, và mốc thời gian, cụ thể theo dạng from src to dst on day hour half. Trong đó half được tính bằng minute chia 30, nhờ vậy mỗi subject tương ứng với một cửa sổ nửa giờ. Cách gom nhóm này làm cho chuỗi có ý nghĩa vận hành rõ ràng, dễ đối soát với thời gian cảnh báo và dễ quy về client_ip. Để tránh nhiễu do chuỗi quá ngắn, các subject có số event dưới ngưỡng tối thiểu sẽ bị bỏ qua. Quy tắc này giúp giảm false alarm do mẫu quá ít thông tin.

Bảng 4.7: Ví dụ subject và ý nghĩa.

Subject mẫu	Diễn giải
from A to B on 16 14 0	luồng từ A vào B trong ngày 16, giờ 14, nửa giờ đầu
from A to B on 16 14 1	luồng từ A vào B trong ngày 16, giờ 14, nửa giờ sau

4.3.6 Điều phối chạy định kỳ và đảm bảo không chồng lấn

Quá trình tạo dữ liệu và detect được điều phối bằng systemd timer theo chu kỳ. Chương trình auto runner khởi tạo model và codebook một lần, sau đó lặp vô hạn theo chu kỳ 60 giây, mỗi vòng thực hiện split và chạy detect một lần. Cách làm này giảm thời gian khởi tạo lại model và giữ nhịp phát hiện ổn định. Trong detect có cơ chế khoá đơn instance bằng file lock để tránh hai lượt chạy chồng lên nhau khi hệ thống bị trễ hoặc

timer kích hoạt lại sớm. Cơ chế này giúp kết quả ghi ra file predict và ghi xuống SQLite nhất quán, tránh tình trạng ghi đè hoặc corrupt file trung gian.

4.4 Huấn luyện mô hình

4.4.1 Tổ chức dữ liệu, xây dựng tập huấn luyện và tập kiểm thử

Dữ liệu huấn luyện xuất phát từ tập UNSW NB15, đây là bộ dữ liệu chuẩn trong nghiên cứu phát hiện xâm nhập, được xây dựng bởi Australian Centre for Cyber Security và công bố rộng rãi cho mục đích học thuật. Trong pipeline huấn luyện, dữ liệu được tổ chức theo thư mục IP, bên trong mỗi IP có các file CSV theo hướng luồng, điển hình là flow_from_this.csv và flow_to_this.csv. Quy ước này phù hợp với cách biểu diễn theo "thực thể giao tiếp", giúp gom các sự kiện theo từng IP và theo cửa sổ thời gian.

Tập huấn luyện được lấy từ nhóm IP normal theo vai trò "senders" trong UNSW NB15, các chuỗi được đọc từ flow_from_this.csv. Mỗi dòng CSV được ánh xạ thành một khoá sự kiện dạng chuỗi, khoá này được sinh bởi hàm trích đặc trưng theo kịch bản logkeys và phép rời rạc hoá theo tham số key_divisor. Các khoá sự kiện sau đó được gom lại thành chuỗi theo một định danh subject có cấu trúc "from srcip to dstip on day hour window", trong đó window là chỉ số cửa sổ thời gian tính theo phút chia cho window_size. Cách gom nhóm này đảm bảo chuỗi phản ánh ngữ cảnh liên lạc và biến động theo thời gian, đồng thời tạo điều kiện đối soát về sau theo ngày và theo khung giờ.

Tập kiểm thử được xây dựng từ flow_to_this.csv của một phần IP normal và một phần IP victims. Một subject được xem là bất thường khi chuỗi nhãn chứa nhãn tấn công và số lượng event có nhãn tấn công lớn hơn 0, các subject còn lại được xem là bình thường. Cơ chế này tạo ra tập kiểm thử hỗn hợp, cho phép đánh giá khả năng tách lớp giữa normal và abnormal trên cùng một quy trình suy luận.

4.4.2 Lệnh chạy huấn luyện và cấu hình tham số

Quá trình huấn luyện được thực hiện thông qua script thực nghiệm exp-unswnb15.py với chế độ ad, lệnh chạy như sau:

```
python3   exp-unswnb15.py ad \
          -i datasets/unsw-nb15/ \
          -o out/unsw-nb15/train_30epoch_1034 \
          -m dablog \
          --epochs 30 \
          --window-size 30 \
```

```
    --key-divisor 100 \
    --use-gpu 0
```

Trong đó, các tham số chính có ý nghĩa như sau:

- Tham số -i chỉ thư mục dữ liệu UNSW NB15 theo cấu trúc mỗi IP là một thư mục con, bên trong chứa các file luồng theo hướng. Tập huấn luyện mặc định đọc từ flow_from_this.csv, tập kiểm thử mặc định đọc từ flow_to_this.csv.
- Tham số -o chỉ thư mục đầu ra, thư mục này lưu model đã huấn luyện, codebook và các file phục vụ đánh giá như predict, subjectranks, auroc, auprc.
- Tham số -m dablog chọn mô hình Dablog, tương ứng kiến trúc LSTM autoencoder sâu và lớp softmax theo từng bước thời gian.
- Tham số --epochs 30 đặt số vòng lặp huấn luyện. Việc dừng sớm vẫn có thể xảy ra do callback early stopping theo loss, tuy nhiên trong lần chạy minh hoạ mô hình huấn luyện đủ số epoch và được lưu lại ở cuối quá trình.
- Tham số --window-size 30 quy định cách gom các sự kiện vào cùng một subject theo cửa sổ thời gian 30 phút. Trong code, subject được tạo theo dạng from src to dst on day hour minute_div_window, trong đó minute_div_window là datetime.minute // 30. Tham số này ảnh hưởng trực tiếp tới độ dài chuỗi, mật độ sự kiện trong một subject và tính nhất quán theo ngữ cảnh thời gian.
- Tham số --key-divisor 100 quy định mức rời rạc hoá khi tạo logkey. Trong hàm tạo khoá, tổng số gói spkts + dpkts được chia nguyên cho 100 để tạo thành một đặc trưng dạng bậc thang, mục tiêu là giảm nhiễu và làm khoá ổn định hơn giữa các phiên lưu lượng.
- Tham số --use-gpu 0 giới hạn TensorFlow sử dụng GPU có chỉ số 0, phù hợp với môi trường chỉ có một GPU.

4.4.3 Tham số mặc định quan trọng và tác động tới mô hình

Ngoài các tham số truyền từ dòng lệnh, một số tham số mặc định trong script có ảnh hưởng trực tiếp tới đầu vào và đầu ra của mô hình:

- Tham số seqlen mặc định bằng 10, đây là độ dài cửa sổ chuỗi khi đóng gói batch. Kết quả là đầu ra softmax có dạng None, 10, vocab_size, phù hợp với bảng tóm tắt kiến trúc đã trình bày trước đó.
- Tham số logkeys mặc định bằng 0, tương ứng kịch bản tạo khoá số 0 trong module key. Kịch bản này sử dụng tập thuộc tính gồm proto, state, service, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, trans_depth, is_sm_ips_ports và thêm đặc trưng tổng gói đã rời rạc hoá. Cách chọn khoá này giúp kết hợp tín hiệu

giao thức và trạng thái, đồng thời đưa thêm một thành phần phản ánh cường độ lưu lượng ở mức thô.

- Tham số stats_step mặc định bằng 0.01 được dùng ở giai đoạn kiểm thử để tạo danh sách misses đủ dày cho phân tích rank score. Trong cấu hình model, giá trị này được gán cho rank_threshold khi tạo file predict. Điều này có nghĩa trong bước sinh misses, chỉ cần nhãn đúng không nằm trong nhóm rất nhỏ các dự đoán đầu là cửa sổ đã bị đánh dấu. Thiết kế này phục vụ mục tiêu đánh giá và dựng đường ROC, PR theo rank, không phải là ngưỡng triển khai cuối cùng.

4.4.4 Mô tả bảng tóm tắt kiến trúc mô hình

A. Bảng mô hình tổng thể Model functional 1

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
Label_Input (InputLayer)	(None, None)	0	-
Label_Embed (Embedding)	(None, None, 64)	18,624	Label_Input[0][0]
not_equal (NotEqual)	(None, None)	0	Label_Input[0][0]
RNN (Sequential)	(None, 10, 291)	97,507	Label_Embed[0][0], not_equal[0][0]

Total params: 116,131 (453.64 KB)
Trainable params: 116,131 (453.64 KB)
Non-trainable params: 0 (0.00 B)

Hình 4.6: Bảng mô hình tổng thể Model functional 1.

Bảng Model functional 1 mô tả luồng dữ liệu từ đầu vào đến đầu ra của hệ thống suy luận:

- Layer đầu vào là Label Input, có dạng tensor hai chiều với kích thước (None, None). Giá trị None thứ nhất biểu thị batch size thay đổi tuỳ theo cấu hình huấn luyện. Giá trị None thứ hai biểu thị độ dài chuỗi theo thời gian có thể thay đổi, tuy nhiên trong thí nghiệm chuỗi được đóng gói theo độ dài cố định seqlen. Từ bảng mô hình con ở phần sau có thể suy ra seqlen bằng 10.
- Lớp Label Embed là Embedding, có đầu ra (None, None, 64). Điều này cho thấy mỗi chỉ số sự kiện trong codebook được ánh xạ sang vector liên tục 64 chiều. Số tham số của Embedding là 18,624. Giá trị này bằng 291 nhân 64, do đó có thể xác nhận kích thước codebook trong lần huấn luyện là 291 token, mỗi token có embedding 64 chiều.
- Lớp not equal là NotEqual, tạo ra mask theo thời gian nhằm đánh dấu vị trí đệm. Đầu ra (None, None) thể hiện mask có cùng chiều thời gian với chuỗi đầu vào. Mask này được nối vào khối RNN để hỗ trợ bỏ qua các phần padding khi tính toán.

- Khối RNN là một mô hình con dạng Sequential. Đầu ra (None, 10, 291) có ý nghĩa. Mỗi mẫu đầu vào được ánh xạ thành một chuỗi 10 bước thời gian. Ở mỗi bước, mô hình trả về phân phối xác suất trên 291 token của codebook. Tổng tham số của mô hình tổng thể là 116,131, trong đó toàn bộ tham số huấn luyện đều nằm trong Embedding và khối RNN.

B. Bảng mô hình con Model RNN

Model: "RNN"

Layer (type)	Output Shape	Param #
LSTM_1 (LSTM)	(None, None, 64)	33,024
LSTM_2 (LSTM)	(None, 32)	12,416
repeat_vector (RepeatVector)	(None, 10, 32)	0
LSTM_3 (LSTM)	(None, 10, 32)	8,320
LSTM_4 (LSTM)	(None, 10, 64)	24,832
Softmax (TimeDistributed)	(None, 10, 291)	18,915

Total params: 97,507 (380.89 KB)
Trainable params: 97,507 (380.89 KB)
Non-trainable params: 0 (0.00 B)

Hình 4.7: Bảng mô hình con Model RNN.

Bảng Model RNN mô tả chi tiết kiến trúc Deep LSTM Autoencoder và lớp phân loại theo thời gian:

- Tầng LSTM 1 có đầu ra (None, None, 64). Đây là tầng encoder thứ nhất, giữ chuỗi hidden state theo thời gian với số chiều 64. Việc giữ chuỗi theo thời gian cho phép tầng encoder kế tiếp học quan hệ dài hạn giữa các bước sự kiện.
- Tầng LSTM 2 có đầu ra (None, 32). Đây là tầng nén, lấy trạng thái cuối để tạo vector biểu diễn ngữ cảnh 32 chiều. Vector này có thể xem như latent representation của cửa sổ chuỗi đầu vào. Lớp RepeatVector có đầu ra (None, 10, 32).
- Lớp này nhân bản vector latent 32 chiều thành chuỗi 10 bước, tương ứng seqlen bằng 10. Đây là bước chuyển từ encoder sang decoder, tạo điều kiện cho decoder tái tạo lại chuỗi theo chiều thời gian.
- Tầng LSTM 3 có đầu ra (None, 10, 32). Đây là tầng decoder thứ nhất, duy trì chiều biểu diễn 32 trong quá trình tái tạo chuỗi.
- Tầng LSTM 4 có đầu ra (None, 10, 64). Đây là tầng decoder thứ hai, mở rộng chiều biểu diễn lên 64 để tăng năng lực biểu diễn trước khi đi vào lớp phân loại.
- Lớp Softmax được triển khai dưới dạng TimeDistributed, có đầu ra (None, 10, 291). TimeDistributed cho phép áp dụng cùng một lớp Dense lên từng bước thời gian. Softmax biến vector 64 chiều tại mỗi bước thành phân phối xác suất trên 291 token. Nhờ đó mô hình có thể tối ưu hoá bằng categorical cross entropy theo

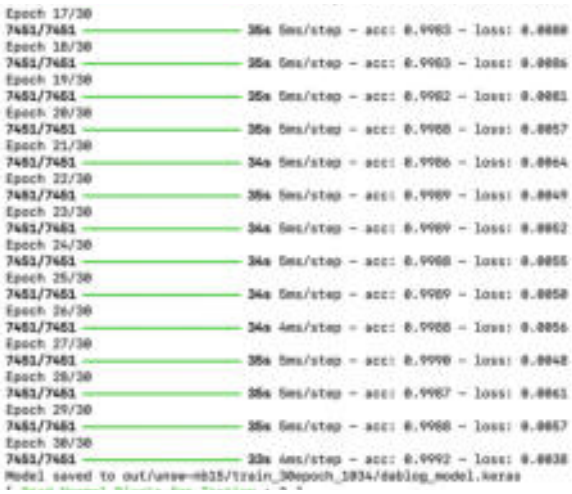
từng bước và suy ra độ bất thường bằng rank score trong giai đoạn kiểm thử và triển khai.

Tổng tham số của khối RNN là 97,507. Giá trị này khớp với tham số của hàng RNN trong bảng mô hình tổng thể, chứng tỏ toàn bộ phần học sâu nằm trong mô hình con này.

4.4.5 Kết quả huấn luyện

Quá trình huấn luyện được thực hiện với 30 epoch. Trong các epoch cuối, mô hình hội tụ ổn định, độ chính xác trên tập huấn luyện duy trì xấp xỉ 0.998 đến 0.999, đồng thời hàm mất mát giảm xuống mức rất thấp.

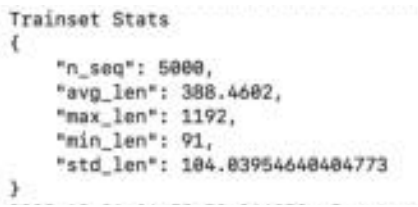
Ở epoch 30, log cho thấy accuracy đạt 0.9992 và loss đạt 0.0038, mô hình được lưu ra file out/unsw-nb15/train_30epoch_1034/dablog_model.keras.



Hình 4.8: Tiến trình huấn luyện theo epoch, thể hiện loss và accuracy, cùng dòng lưu model.

Thời gian huấn luyện mỗi epoch quan sát được khoảng 33 đến 35 giây, với 7451 step mỗi epoch. Chỉ báo này phản ánh tốc độ xử lý tốt và đủ phù hợp để lặp lại nhiều lần khi cần tinh chỉnh tham số.

Ngoài ra, thống kê chuỗi của tập huấn luyện cho thấy số lượng chuỗi huấn luyện là 5000. Độ dài chuỗi trung bình khoảng 388.46 event, độ dài nhỏ nhất 91, lớn nhất 1192, độ lệch chuẩn khoảng 104.04. Thống kê này cho thấy dữ liệu huấn luyện tập trung vào các chuỗi có độ dài vừa phải, phù hợp với cơ chế cắt cửa sổ seqlen trong huấn luyện.



Hình 4.9: Thống kê tập huấn luyện, gồm n seq, avg len, min len, max len.

4.4.6 Kết quả kiểm thử và đánh giá

Sau khi huấn luyện, mô hình được chạy kiểm thử trên tập test. Thống kê tập test cho thấy n seq là 1981. Độ dài chuỗi trung bình khoảng 352.65 event. Độ dài nhỏ nhất 1, lớn nhất 18453, độ lệch chuẩn khoảng 562.89. Việc max len rất lớn cho thấy tập test chứa một số subject có mật độ sự kiện cao, đây là điều kiện khó và phù hợp để kiểm tra độ bền vững của cơ chế phát hiện.

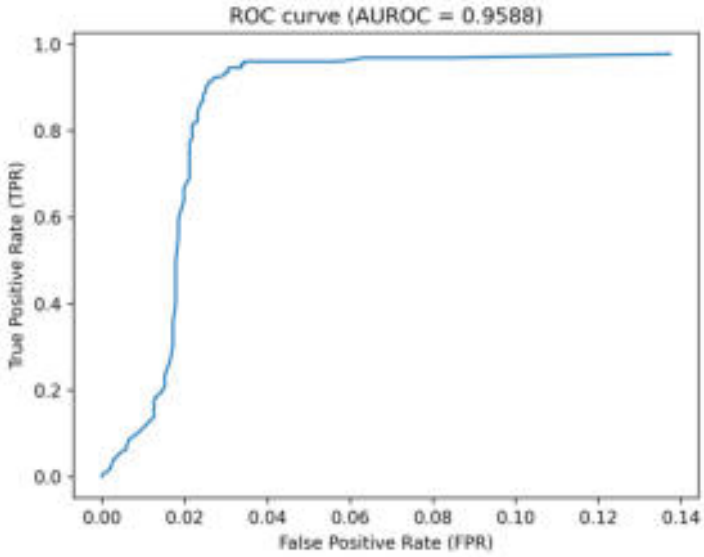
Hình 4.10: Log kiểm thử, gồm thống kê testset, batch shape, và prototype misses.

Trong pha suy luận kiểm thử, batch được đóng gói theo seqlen bằng 10, log hiển thị batch shape theo dạng, đầu vào có kích thước 684790 nhân 10, và đầu ra one hot tương ứng 684790 nhân 10 nhân 291. Quá trình derive anomalies ghi nhận prototype misses là 29886, đây là số lượng vị trí bị đánh dấu miss theo tiêu chí rank trong toàn bộ các cửa sổ kiểm thử, dùng làm cơ sở tổng hợp điểm bất thường theo subject.

Sau khi suy ra các điểm bất thường theo subject, chất lượng mô hình được đánh giá thông qua các đồ thị ROC, Precision Recall, đường biến thiên Precision Recall F1 theo ngưỡng, cùng với phân bố rank score theo lớp. Các đồ thị này giúp kiểm tra đồng thời hai khía cạnh, khả năng tách lớp tổng quát của rank score, và khả năng chọn ngưỡng triển khai phù hợp với yêu cầu vận hành.

A. Đường ROC và ý nghĩa AUROC

Hình 4.11 là đường ROC, trục hoành là FPR, tức tỉ lệ cảnh báo nhầm trên lớp bình thường, trục tung là TPR, tức tỉ lệ phát hiện đúng trên lớp bất thường. Mỗi điểm trên đường cong tương ứng với một giá trị ngưỡng rank score khác nhau.

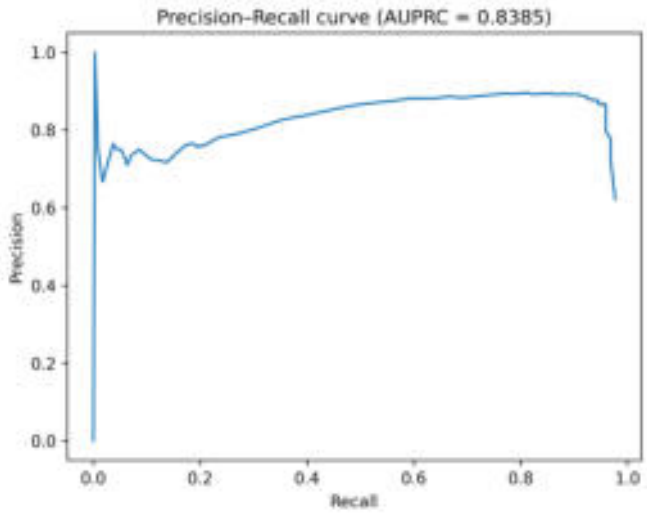


Hình 4.11: Đường ROC và AUROC.

Hình vẽ được xem là tốt khi đường cong tiến sát góc trên bên trái, nghĩa là đạt TPR cao trong khi FPR vẫn thấp. Trong kết quả này, đường ROC tăng nhanh lên vùng TPR rất cao ngay ở mức FPR nhỏ, cho thấy chỉ cần chấp nhận một lượng nhỏ cảnh báo giả thì đã thu được phần lớn bất thường. AUROC bằng 0.9588 phản ánh khả năng phân biệt tổng quát tốt. Nếu mô hình hoạt động chưa tốt, đường ROC sẽ gần đường chéo, AUROC xấp xỉ 0.5, nghĩa là gần như không hơn đoán ngẫu nhiên.

B. Đường Precision Recall và ý nghĩa AUPRC

Hình 4.12 là đường Precision Recall, trục hoành là Recall, trục tung là Precision. Đường cong này đặc biệt quan trọng khi dữ liệu bị mất cân bằng giữa lớp bình thường và bất thường, vì ROC đôi khi vẫn cao ngay cả khi precision không tốt.

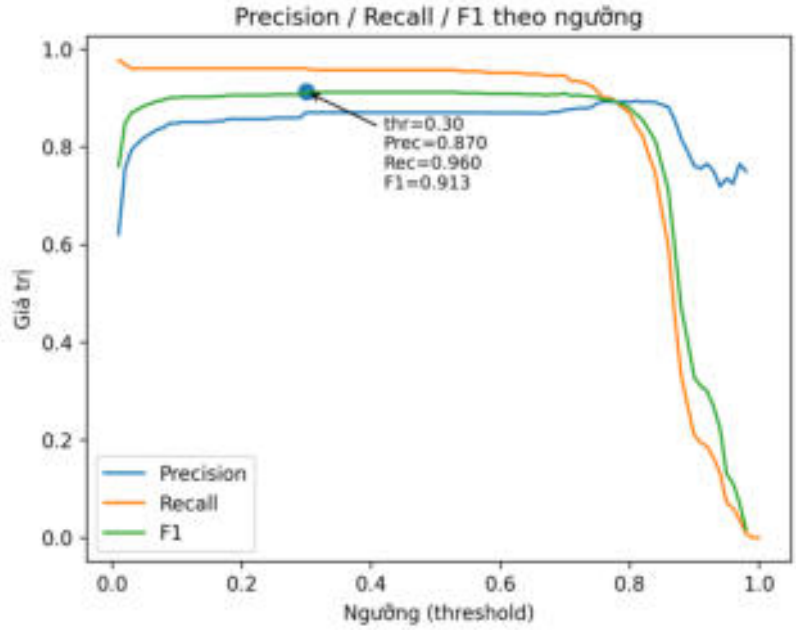


Hình 4.12: Đường Precision Recall và AUPRC.

Một mô hình tốt sẽ duy trì precision cao khi recall tăng, nghĩa là bắt được nhiều bất thường nhưng vẫn giữ được tỉ lệ đúng cao. Trong kết quả này, AUPRC bằng 0.8385 cho thấy độ chính xác tổng hợp theo miền recall là tốt, và đường cong duy trì vùng precision khoảng 0.8 đến 0.9 trong một dải recall rộng. Nếu mô hình chưa tốt, đường precision sẽ tụt nhanh khi recall tăng, và AUPRC sẽ thấp, thường tiến gần về tỉ lệ dương tính của tập dữ liệu.

C. Đường Precision Recall F1 theo ngưỡng và cách đọc điểm thr 0.45

Hình 4.13 biểu diễn Precision, Recall và F1 khi quét ngưỡng rank score từ 0 đến 1. Trục hoành là ngưỡng, trục tung là giá trị của các chỉ số.

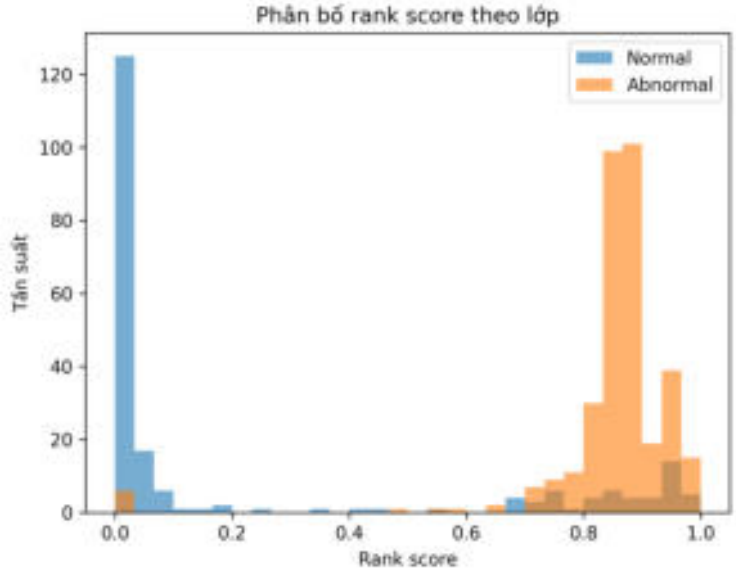


Hình 4.13: Đường Precision Recall và AUPRC.

Trong hình, đường recall thường giảm khi ngưỡng tăng vì ngưỡng cao làm hệ thống "khó tính" hơn, dẫn đến bỏ sót nhiều hơn. Ngược lại, precision thường tăng khi ngưỡng tăng vì chỉ giữ lại các mẫu rất chắc chắn, làm giảm cảnh báo giả. F1 là trung bình điều hoà của precision và recall nên thường có một vùng đạt cực đại, đây là điểm cân bằng. Điểm đánh dấu thr 0.3 cho thấy precision bằng 0.870, recall bằng 0.960, F1 bằng 0.913. Việc recall vẫn rất cao tại ngưỡng này chứng tỏ mô hình vẫn bắt được đa số bất thường, đồng thời precision đủ tốt để hạn chế cảnh báo giả, phù hợp mục tiêu triển khai ban đầu. Nếu mô hình chưa tốt, thường sẽ thấy F1 không tạo được đỉnh rõ ràng, hoặc đỉnh nằm ở vùng ngưỡng rất thấp kèm precision thấp, điều này hàm ý hệ thống phải đánh đổi bằng rất nhiều cảnh báo giả mới đạt được recall.

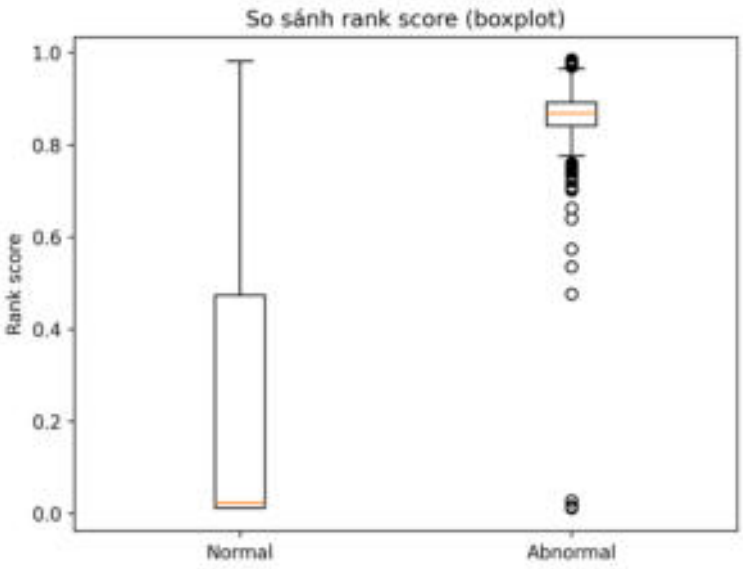
D. Phân bố rank score theo lớp và boxplot so sánh

Hình 4.14 là histogram phân bố rank score theo lớp normal và abnormal. Trục hoành là rank score, trục tung là tần suất. Ý nghĩa của đồ thị này là kiểm tra mức độ tách biệt của hai phân bố.



Hình 4.14: Histogram phân bố rank score theo lớp.

Mô hình hoạt động tốt khi phân bố abnormal dịch rõ sang phía phải, tập trung ở vùng rank cao, trong khi phân bố normal tập trung ở vùng rank thấp. Trong hình, normal tập trung mạnh ở gần 0, còn abnormal tập trung ở khoảng 0.8 đến 1.0. Vùng chồng lấn nhỏ phản ánh khả năng phân biệt tốt. Nếu mô hình chưa tốt, hai phân bố sẽ chồng lấn lớn, hoặc abnormal không dịch sang vùng cao, khi đó bất kỳ ngưỡng nào cũng sẽ tạo ra nhiều FP hoặc nhiều FN.

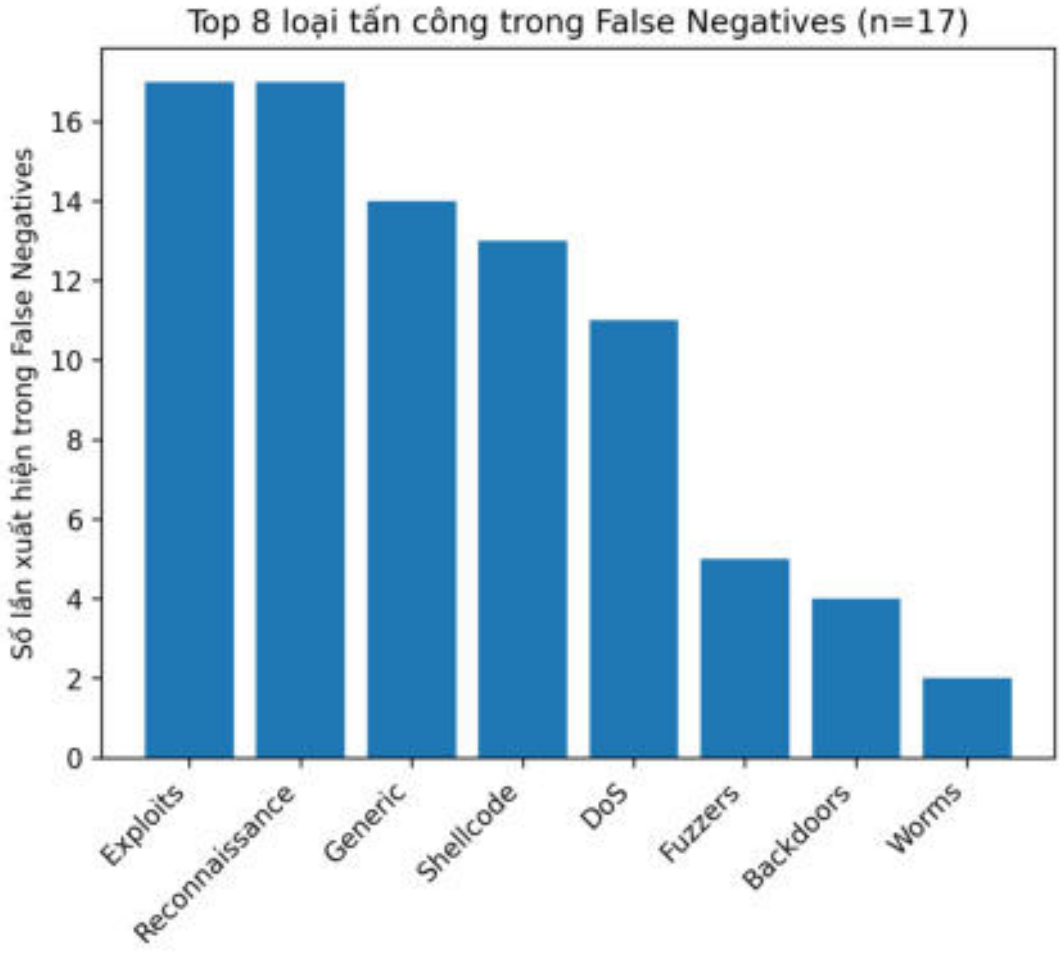


Hình 4.15: Boxplot so sánh rank score giữa normal và abnormal.

Hình 4.15 là boxplot so sánh rank score, mỗi hộp thể hiện trung vị, tứ phân vị, và các điểm ngoại lai. Boxplot tốt khi trung vị của abnormal cao hơn rõ rệt, hộp ít chồng lấn, và khoảng biến thiên của abnormal hẹp hơn normal. Trong kết quả này, abnormal có trung vị cao và tương đối tập trung, trong khi normal có trung vị thấp và có một số ngoại lai kéo dài lên vùng cao, phù hợp với hiện tượng vẫn còn một số subject bình thường nhưng có hành vi hiếm.

E. Phân tích false negatives theo loại tấn công

Hình 4.16 liệt kê các loại tấn công xuất hiện trong nhóm false negatives. Trục hoành là loại tấn công, trục tung là số lần xuất hiện trong FN. Mục tiêu của hình này là chỉ ra các nhóm khó phát hiện, từ đó định hướng cải tiến trong tương lai, ví dụ bổ sung logkey, tăng seqlen, hoặc thay đổi chiến lược chọn ngưỡng.



Hình 4.16: Top loại tấn công trong false negatives.

Đồ thị này không dùng để kết luận mô hình tốt hay xấu một cách trực tiếp, mà dùng để giải thích giới hạn. Tuy nhiên, nếu tổng số FN nhỏ và chỉ tập trung vào một vài

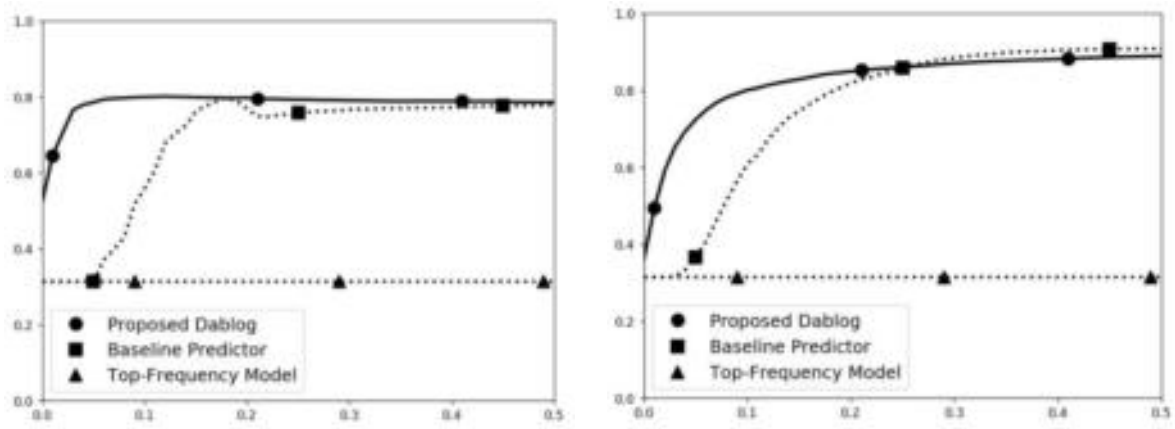
nhóm, điều đó cho thấy mô hình đã bao phủ tốt phần lớn hành vi bất thường, và phần còn lại có thể cải thiện theo hướng đặc thù.

4.4.7 So sánh giữa DeepLog và DabLog huấn luyện trên UNSW-NB15

Để so sánh công bằng giữa Baseline (DeepLog-style) và DabLog 1 chiều, đồ án cố định các điều kiện sau:

- Cùng pipeline tạo chuỗi theo subject và cùng window_size.
- Cùng cấu hình rời rạc hoá log key theo logkeys và key_divisor.
- Cùng seqlen khi đóng gói cửa sổ.
- Cùng cách đánh giá theo rank score/miss và tổng hợp theo subject.

Trên mỗi cấu hình số lượng log key, hệ thống quét ngưỡng τ trên rank score để tìm điểm làm việc tối ưu theo F1-score, phản ánh cân bằng giữa giảm báo động nhầm và giảm bỏ sót.



(a) 366 khóa. (b) 706 khoá.

Hình 4.17: Hai đồ thị F1 theo ρ_N trên UNSW-NB15.

Hình 4.17 mô tả so sánh giữa DabLog và Baseline (DeepLog-style) khi huấn luyện trên UNSW-NB15 với hai cấu hình số lượng log keys khác nhau: (a) 366 keys và (b) 706 keys. Trên đồ thị, trục x là ρ_N và trục y là $\mathrm{F1}-\mathrm{score}$. Ngoài hai mô hình chính, tài liệu còn đưa thêm một đối chứng đơn giản là frequency model, đánh dấu bất thường nếu chuỗi chứa sự kiện không thuộc Top-N keys xuất hiện thường xuyên nhất. Quan sát cho thấy cả hai mô hình đều có đường cong F1 mượt và vùng plateau rộng, hàm ý dữ liệu UNSW-NB15 có ít điểm dữ liệu quá hiếm (infrequent) trong bối cảnh biểu diễn theo keys. Đồng thời, F1 của Baseline và DabLog khá tương đồng, tuy nhiên DabLog vẫn nhỉnh hơn nhẹ trên cả hai cấu hình 366 keys và 706 keys. Vì vậy, trong thực nghiệm trên UNSW-NB15, DabLog được kết luận là có lợi thế hơn baseline.

4.5 Tích hợp mô hình vào hệ thống

4.5.1 Nạp mô hình và codebook vào hệ thống

Trong triển khai, mô hình và codebook được nạp một lần khi tiến trình suy luận khởi động, sau đó tái sử dụng cho các vòng chạy tiếp theo để giảm thời gian khởi tạo. Quy trình nạp gồm hai thành phần chính. Thứ nhất là codebook, codebook ánh xạ chuỗi khoá sự kiện sang chỉ số nguyên, bảo đảm các token khi suy luận được mã hoá giống huấn luyện. Codebook được đọc từ file json đã lưu sau huấn luyện. Thứ hai là mô hình Keras, kiến trúc gồm embedding và deep LSTM autoencoder kèm tầng softmax theo từng bước thời gian. Khi nạp, mô hình được dựng đúng cấu trúc, sau đó tải trọng số đã huấn luyện từ file weights. Khi nạp xong, hệ thống giữ model và codebook trong bộ nhớ, nhờ đó mỗi vòng chạy chỉ thực hiện bước tạo batch và dự đoán.

4.5.2 Tổ chức đầu vào suy luận theo subject và cửa sổ thời gian

Đầu vào trực tiếp của suy luận là các dòng dữ liệu flow đã chuẩn hoá, mỗi dòng tương ứng một phiên kết nối. Từ mỗi dòng, hệ thống tạo một mô tả subject có ý nghĩa vận hành, gồm client ip và mốc thời gian theo dạng ngày, giờ, nửa giờ. Các dòng được gom nhóm theo subject, sau đó mỗi nhóm được chuyển thành chuỗi sự kiện, mỗi sự kiện là một token rời rạc được tạo từ các thuộc tính tiêu biểu của flow như giao thức, trạng thái, dịch vụ, cổng đích, và các thống kê lưu lượng đã được rời rạc hoá theo key divisor. Việc rời rạc hoá giúp giảm nhiễu và giảm số lượng giá trị liên tục, đồng thời làm cho không gian token ổn định giữa dữ liệu huấn luyện và dữ liệu thực tế. Để tránh cảnh báo kém tin cậy khi dữ liệu quá ít, hệ thống loại các subject có số lượng sự kiện thấp hơn ngưỡng tối thiểu, ví dụ 20 sự kiện, chỉ giữ các subject đủ dài để mô hình quan sát được ngữ cảnh.

4.5.3 Đóng gói batch, chạy dự đoán, và suy ra bất thường

Sau khi tạo các chuỗi theo subject, hệ thống đóng gói batch theo seqlen, tức số bước thời gian mỗi cửa sổ. Mỗi cửa sổ được mô hình dự đoán phân phối xác suất trên toàn bộ không gian nhãn sự kiện tại từng bước thời gian. Tiêu chí bất thường ở mức cửa sổ dựa trên rank, tức mức độ bất ngờ của sự kiện thật so với các dự đoán của mô hình. Khi sự kiện thật bị đẩy xuống nhóm xác suất thấp, rank tăng, cửa sổ đó được xem là miss. Tham số rank threshold trong triển khai quyết định mức nhạy, rank threshold càng nhỏ thì càng nhạy, tức càng dễ đánh dấu miss. Từ danh sách các cửa sổ bị miss, hệ thống tổng hợp về mức subject để tạo chỉ số vận hành. Số cửa sổ bị miss, tổng số cửa sổ của subject, tỷ lệ miss rate. Thống kê điểm score theo các cửa sổ bị đánh dấu, gồm giá trị

nhỏ nhất, trung bình, lớn nhất, giúp GUI có thêm tín hiệu để sắp xếp và lọc. Chỉ số diễn giải, như số sự kiện quan sát, và các lý do theo luật để hỗ trợ quyết định cảnh báo.

4.5.4 Ghi kết quả vào SQLite và liên kết với Backend

Kết quả suy luận được ghi vào SQLite theo 2 mức. Mức lần chạy, lưu vào bảng detect runs, gồm thời điểm chạy, server ip, bucket key, số chuỗi, số lượng miss, phục vụ theo dõi lịch sử vận hành. Mức subject, lưu vào bảng detect subjects, gồm client ip, dst ip, bucket key, số sự kiện, tổng cửa sổ, miss count, miss rate, score min, score avg, score max, quyết định và điểm, kèm lý do dưới dạng json để GUI hiển thị chi tiết. Ngoài ra, khi subject đạt mức nguy cơ cao theo luật, hệ thống cập nhật bảng blocklist để ghi nhận ip cần chặn hoặc trạng thái đã chặn, Backend và GUI dùng bảng này để hiển thị danh sách IP đã chặn và hỗ trợ thao tác quản trị.

4.5.5 Tối ưu vận hành và an toàn khi chạy lặp

Để bảo đảm ổn định khi chạy theo lịch, khối suy luận có các cơ chế vận hành sau. Khoá một tiến trình tại một thời điểm, tránh hai vòng chạy chồng lên nhau gây ghi đè file và ghi trùng CSDL. Xoay vòng file kết quả dự đoán, kiểm tra tính toàn vẹn file nén trước khi dùng lại, giảm rủi ro hỏng file khi bị ngắt giữa chừng. Bật tắt ghi CSDL bằng biến môi trường, giúp chuyển sang chế độ chỉ log khi cần kiểm tra nhanh. Dọn rác bộ nhớ theo chu kỳ, hạn chế tăng bộ nhớ khi chạy lâu.

4.6 Xây dựng Backend API

4.6.1 Quy ước thiết kế và cơ chế phân quyền

Backend cung cấp các HTTP API theo phong cách REST, dữ liệu trao đổi sử dụng JSON, các thao tác xem log dùng thêm cơ chế streaming SSE để cập nhật gần thời gian thực. Mọi API nghiệp vụ đều yêu cầu xác thực, ngoại trừ endpoint kiểm tra sống và endpoint đăng nhập. Cơ chế xác thực sử dụng JWT theo mô hình Bearer Token. Sau khi đăng nhập thành công, hệ thống trả về token, phía GUI gắn token vào header Authorization: Bearer <token> cho các request tiếp theo. Thời hạn token được cấu hình 12 giờ, payload chứa id, username, role để phục vụ phân quyền. Phân quyền được tổ chức theo 2 vai trò chính, admin và user. Vai trò admin được phép quản trị tài khoản, thêm hoặc bật tắt danh sách IP chặn. Vai trò user được phép xem dữ liệu, xem cảnh báo, thực hiện review cảnh báo. Hệ thống áp dụng các nguyên tắc kiểm soát đầu vào ở mức API, gồm ràng buộc định dạng ngày YYYY-MM-DD, giới hạn limit để tránh truy vấn quá lớn, lọc tên file bằng basename để hạn chế path traversal.

4.6.2 Nhóm API xác thực và quản trị người dùng

Nhóm API này phục vụ đăng nhập và quản trị tài khoản để vận hành hệ thống. Đăng nhập trả về JWT, quản trị người dùng chỉ cho phép vai trò admin.

Bảng 4.8: Các API xác thực và quản trị người dùng.

Endpoint	Method	Mục đích	Phân quyền	Đầu vào chính	Đầu ra chính
/api/auth/login	POST	Đăng nhập, cấp JWT	Public	username, password	token, role, username
/api/users	GET	Danh sách người dùng	Admin	query rỗng	users
/api/users	POST	Tạo người dùng	Admin	username, password, role	ok
/api/users/:id	PATCH	Bật tắt tài khoản	Admin	enabled	ok
/api/users/:id	DELETE	Xoá mềm tài khoản	Admin	path id	ok

Một số ràng buộc quan trọng, tên đăng nhập bị giới hạn theo regex, mật khẩu tối thiểu 6 ký tự. Hệ thống không cho phép tự khoá hoặc tự xoá chính tài khoản đang đăng nhập để tránh khoá nhầm trong vận hành.

4.6.3 Nhóm API xem log Zeek, bucket UNSW, và Zeek CSV

Nhóm API này phục vụ điều tra, đối soát theo ngày, đồng thời hỗ trợ xem nhanh dữ liệu đầu vào liên quan đến các cảnh báo. Thiết kế phân tách theo 3 nguồn, log Zeek theo cây thư mục ngày, bucket UNSW đã chuẩn hoá, và Zeek CSV theo ngày để GUI tải đúng file.

Bảng 4.9: Các API xem log Zeek, bucket UNSW, và Zeek CSV.

Endpoint	Method	Mục đích	Phân quyền	Đầu vào chính	Đầu ra chính
/api/zeek/years	GET	Liệt kê năm	User	rỗng	years
/api/zeek/months	GET	Liệt kê tháng theo năm	User	year	months
/api/zeek/days	GET	Liệt kê ngày theo năm tháng	User	year, month	days

/api/zeek/files	GET	Danh sách file log theo ngày	User	date	files
/api/zeek/tail	GET	Lấy n dòng cuối của log	User	date, file, limit	lines, size
/api/buckets/files	GET	Danh sách file bucket	User	rỗng	files
/api/buckets/tail	GET	Lấy n dòng cuối bucket	User	file, limit	lines, size
/api/zeekcsv/manifest	GET	Trả manifest file theo ngày	User	date	files[{name,path}]
/api/zeekcsv/tail	GET	Tail Zeek CSV theo ngày	User	date, limit	lines, file
/api/admin/import/zeekcsv	POST	Import Zeek CSV vào DB flows	Admin	query date	imported, day

Dạng trả về của các API tail thường gồm lines là mảng chuỗi, kèm size hoặc file để phía GUI hiển thị trạng thái. Các API liệt kê thư mục trả về mảng tên thư mục hoặc tên file, giúp GUI tạo bộ lọc năm tháng ngày.

4.6.4 Nhóm API phục vụ tra cứu flows và thống kê

Nhóm API này đọc dữ liệu flows trong SQLite, phục vụ điều tra theo IP, hướng lưu lượng, giao thức, dịch vụ, trạng thái. Kết quả trả về có phân trang để GUI tải dần khi dữ liệu lớn.

Bảng 4.10: Các API phục vụ tra cứu flows và thống kê.

Endpoint	**Method**	**Mục đích**	**Phân quyền**	**Đầu vào chính**	**Đầu ra chính**
/api/flows/ips	GET	Liệt kê IP xuất hiện trong ngày	User	date	ips[]
/api/flows/search	GET	Tìm kiếm flows có bộ lọc	User	date, ip, direction, proto, service, state, limit, offset	total, rows[]

/api/stats/by_hour	GET	Thống kê số flows theo giờ	User	date	series[{hour,n}]
/api/stats/top_ips	GET	Top IP nguồn và đích	User	date, limit	top_src, top_dst

Trong đó, /api/flows/search hỗ trợ direction gồm src, dst, any để lọc theo IP nguồn, IP đích, hoặc cả hai. Thống kê theo giờ dùng epoch stime, quy đổi múi giờ UTC+7 để biểu diễn đúng theo vận hành.

4.6.5 Nhóm API cảnh báo bất thường, review, và danh sách IP chặn

Nhóm API này là lõi vận hành. Cảnh báo bất thường được truy vấn theo day, hỗ trợ tìm kiếm nhanh theo IP hoặc subject. Tác vụ review cập nhật review_status, review_by, review_at để tạo vòng phản hồi giữa mô hình và vận hành. Danh sách IP chặn tách riêng, cho phép admin thêm mới và bật tắt.

Bảng 4.11: Các API cảnh báo bất thường, review, và danh sách IP chặn.

Endpoint	**Method**	**Mục đích**	**Phân quyền**	**Đầu vào chính**	**Đầu ra chính**
/api/detect/anomalies	GET	Danh sách cảnh báo theo ngày	User	date, limit, offset, q	total, items[]
/api/detect/anomalies/review	POST	Review cảnh báo	User	id, status	ok
/api/blockedips	GET	Xem danh sách IP chặn	User	rỗng	items[]
/api/blockedips/add	POST	Thêm IP chặn hoặc cập nhật lý do	Admin	ip, reason	ok
/api/blockedips/toggle	POST	Bật tắt trạng thái chặn	Admin	ip, enabled	ok

Ở phía dữ liệu cảnh báo, items[] trả về các thống kê như miss, miss_pct, score_min, score_avg, score_max, và đoạn chỉ số chuỗi seq_from, seq_to. Các trường này giúp giải thích mức độ bất thường, đồng thời liên kết về dữ liệu gốc để đối soát khi cần.

4.6.6 Nhóm API streaming SSE và endpoint giám sát

Để hỗ trợ xem log gần thời gian thực trên GUI, hệ thống cung cấp endpoint SSE. Khi client đăng ký stream theo type, date, file, server ghi nhận offset hiện tại và chỉ gửi phần log mới phát sinh. Cơ chế này giảm tải so với polling liên tục, đồng thời phù hợp với nhu cầu quan sát thay đổi trên log.

Bảng 4.12: API streaming SSE và endpoint giám sát.

Endpoint	Method	Mục đích	Phân quyền	Đầu vào chính	Đầu ra chính
/api/stream	GET	SSE stream log Zeek hoặc bucket	User	type, date, file	event ready, append, error
/api/health	GET	Kiểm tra sống	Public	rỗng	{ok:true}

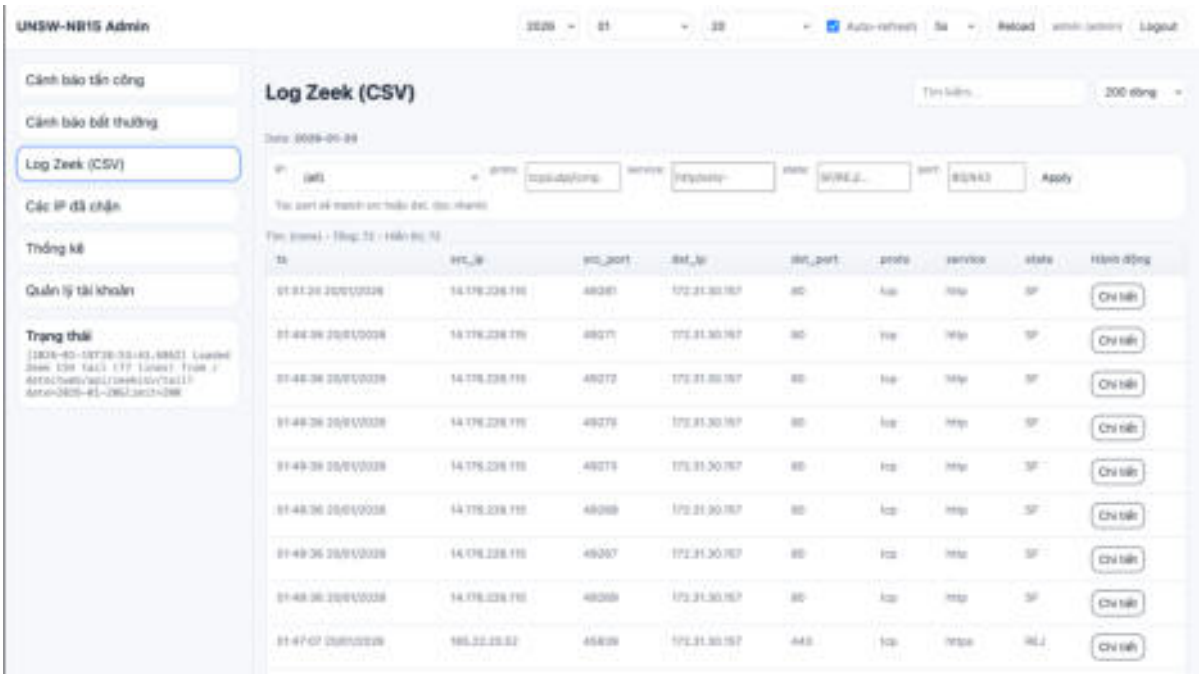
SSE trả dữ liệu theo event. Event ready báo stream đã sẵn sàng và offset ban đầu. Event append mang theo các dòng mới lines[], giúp GUI nối tiếp vào vùng hiển thị.

4.7 Xây dựng giao diện Web/GUI

4.7.1 Mục tiêu và cách tổ chức giao diện

Giao diện Web được xây dựng nhằm phục vụ giám sát và điều tra theo ngày, tập trung vào ba nhóm thao tác chính, xem danh sách cảnh báo, xem chi tiết từng bản ghi, và thực hiện xác minh thủ công để giảm cảnh báo sai.

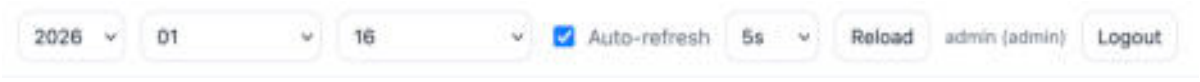
Giao diện được triển khai theo kiểu trang đơn, điều hướng bằng hash route, giúp chuyển đổi nhanh giữa các màn hình mà không cần tải lại toàn bộ trang. Các màn hình được tổ chức theo nhóm chức năng rõ ràng, cảnh báo, log Zeek, danh sách IP đã chặn, thống kê, và quản trị tài khoản. Mỗi màn hình đều sử dụng chung các điều khiển lọc theo ngày, tìm kiếm, giới hạn số dòng hiển thị, và cơ chế làm mới dữ liệu.



Hình 4.18: Màn hình tổng quan giao diện chính.

4.7.2 Thanh điều hướng và các điều khiển dùng chung

Giao diện cung cấp thanh điều hướng cố định để truy cập nhanh các màn hình chính, gồm Cảnh báo tấn công, Cảnh báo bất thường, Log Zeek (CSV), Các IP đã chặn, Thống kê, Quản lý tài khoản. Trạng thái hệ thống được hiển thị ở cuối thanh điều hướng nhằm phản hồi nhanh các sự kiện như tải dữ liệu thành công, lỗi gọi API, hoặc cảnh báo quyền truy cập. Bộ điều khiển chung ở phần đầu trang gồm bộ lọc ngày, ô tìm kiếm, giới hạn số bản ghi, nút tải lại, và tuỳ chọn tự làm mới theo chu kỳ. Trong đó, lọc ngày được thiết kế theo năm, tháng, ngày, hỗ trợ chế độ xem theo năm bằng cách chọn tháng là ALL, khi đó bộ chọn ngày sẽ bị khoá để tránh sai lệch ngữ nghĩa.

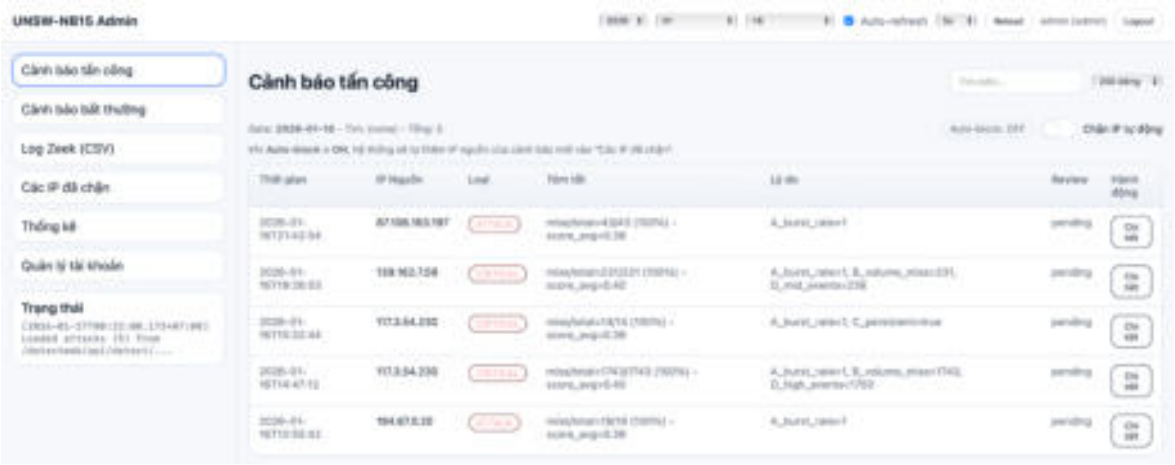


Hình 4.19: Bộ lọc ngày và thanh công cụ chung.

4.7.3 Màn hình Cảnh báo tấn công

Màn hình cảnh báo tấn công hiển thị danh sách các đối tượng bị nghi ngờ theo kết quả tổng hợp từ các lần chạy suy luận theo ngày. Dữ liệu được trình bày dạng bảng, mỗi dòng tương ứng một subject, giúp người vận hành ưu tiên xử lý nhanh theo mức độ bất thường. Các cột chính trong bảng gồm, thời gian, server ip, ip nguồn, loại quyết định, điểm score, trạng thái review, và nút chi tiết. Trạng thái review dùng để đánh dấu

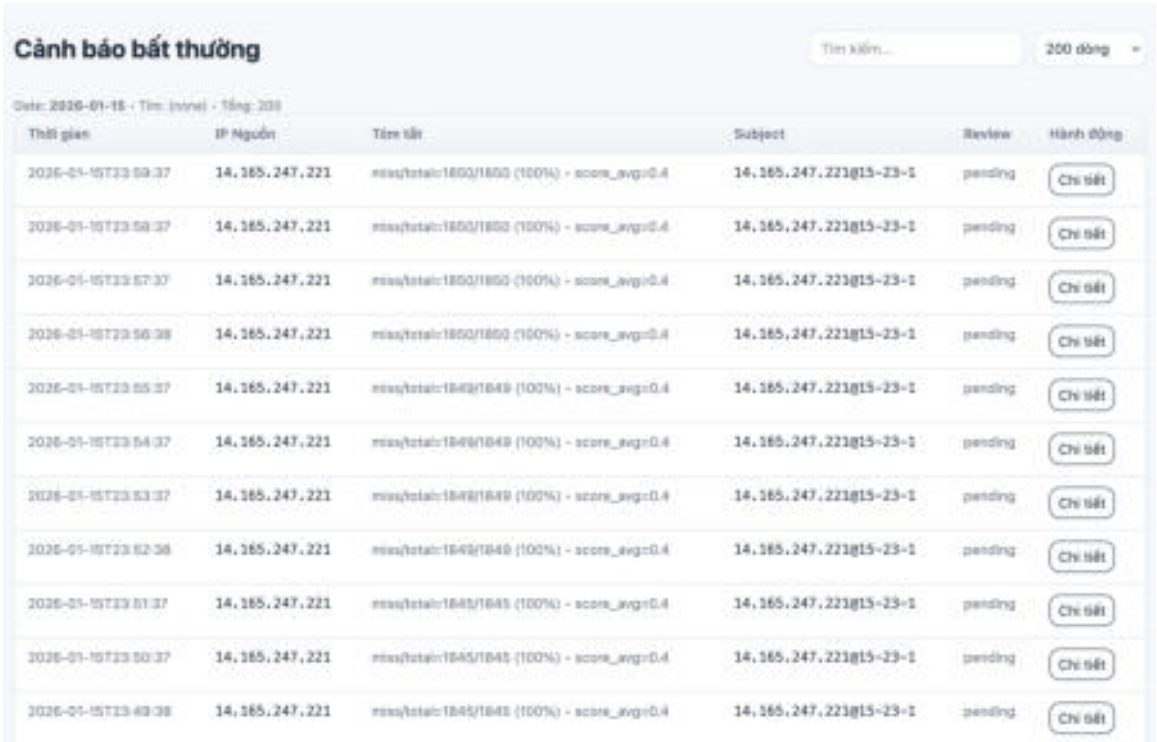
tiến độ xử lý thủ công, giúp thống kê lại tỉ lệ false positive trên các cảnh báo đã được xác minh.



Hình 4.20: Bảng danh sách cảnh báo tấn công.

4.7.4 Màn hình Cảnh báo bất thường

Màn hình Cảnh báo bất thường có cấu trúc tương tự, nhưng tập trung vào bất thường theo chuỗi sự kiện. Bảng hiển thị các trường quan trọng phục vụ điều tra nhanh, gồm thời gian phát hiện, ip, tỉ lệ miss trên tổng cửa sổ, score trung bình, subject, và review status. Với các bản ghi dạng chuỗi, thông tin miss, total, seq_from, seq_to là cơ sở để truy vết đoạn chuỗi gây bất thường và đối soát lại với log Zeek theo cùng ngày.



Hình 4.21: Bảng danh sách cảnh báo bất thường.

4.7.5 Hộp thoại chi tiết và cơ chế review

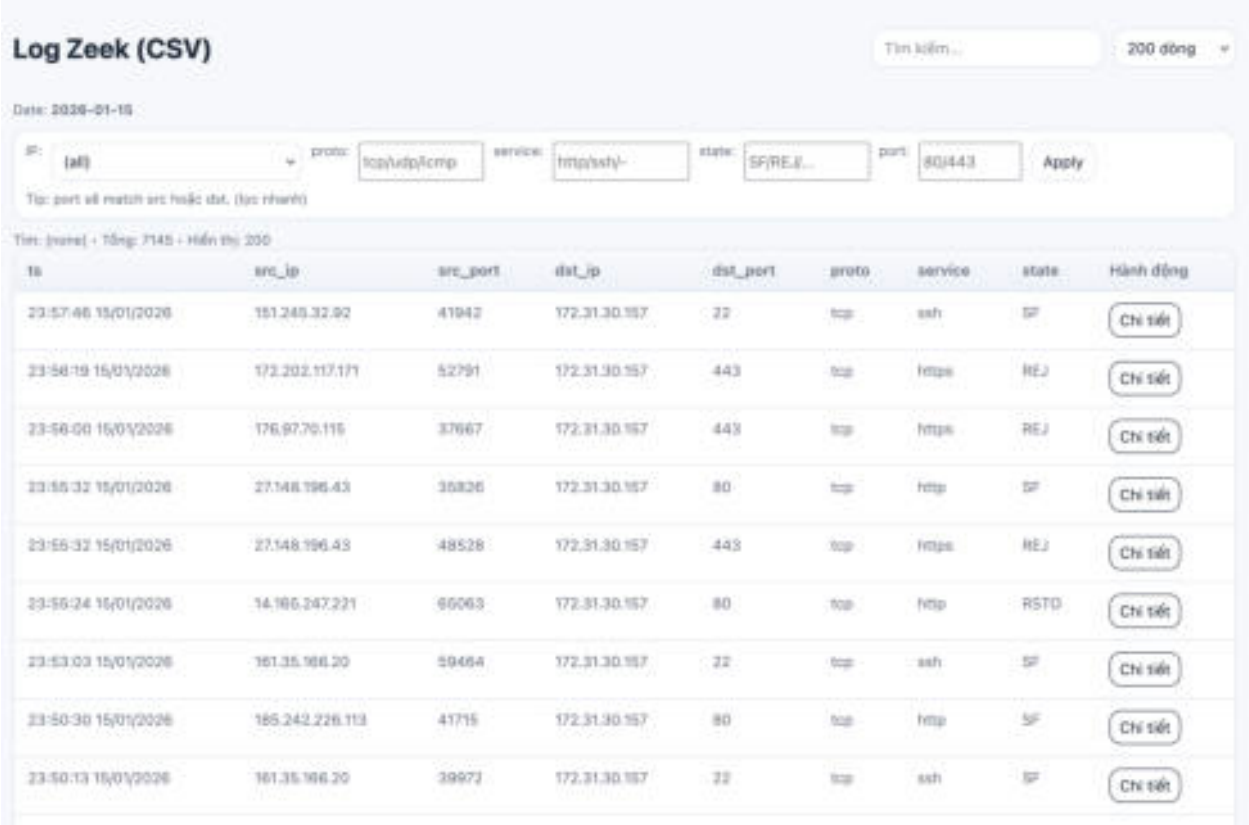
Khi nhấn "Chi tiết" ở bất kỳ màn hình nào, hệ thống mở hộp thoại chi tiết và hiển thị toàn bộ nội dung bản ghi ở dạng JSON có định dạng, kèm phần header gồm date filter, review status, và id được băm để tạo định danh ổn định theo nội dung cảnh báo. Cơ chế này giúp thống nhất thao tác review mà không phụ thuộc trực tiếp vào khoá chính của cơ sở dữ liệu.

Hình 4.22: Hộp thoại chi tiết cảnh báo và thanh review.

Thanh review bar trong hộp thoại cung cấp ba lựa chọn, Pending, True attack, False positive. Khi người vận hành chọn trạng thái mới, giao diện cập nhật ngay và phản ánh lại ở bảng danh sách và trang thống kê. Trường hợp đánh dấu False positive, nếu tài khoản đang đăng nhập là admin và bản ghi có ip, hệ thống có thể gọi API để gỡ chặn ip tương ứng nhằm tránh duy trì chặn sai.

4.7.6 Màn hình Log Zeek

Màn hình Log Zeek (CSV) phục vụ điều tra theo flow, hỗ trợ lọc nhanh theo ip, proto, service, state, và port. Thiết kế bộ lọc theo dạng panel giúp người vận hành thu hẹp phạm vi quan sát, sau đó tải danh sách bản ghi phù hợp và hiển thị theo bảng. Các cột chính gồm ts, src ip, src port, dst ip, dst port, proto, service, state, và nút chi tiết. Khi bấm chi tiết, giao diện mở hộp thoại và hiển thị đầy đủ trường của flow, phục vụ đối soát trực tiếp với các cảnh báo bất thường theo cùng ip và khung thời gian.



Hình 4.23: Màn hình Log Zeek với bộ lọc và bảng flow.

4.7.7 Màn hình Các IP đã chặn

Màn hình Các IP đã chặn hỗ trợ quản lý danh sách địa chỉ IP đã được đưa vào trạng thái chặn trên hệ thống. Mục tiêu của màn hình này là giúp người vận hành theo dõi các quyết định chặn, tra cứu nhanh theo ngày, đồng thời cung cấp thao tác gỡ chặn khi xác minh lại cho thấy đó là cảnh báo sai hoặc không cần chặn nữa. Ở phần đầu trang, hệ thống hiển thị bộ lọc theo Date và trạng thái Trạng thái. Trạng thái mặc định có thể đặt là Đang chặn, nhằm tập trung vào các IP hiện còn hiệu lực. Bên cạnh đó là ô Tìm kiếm để lọc nhanh theo IP hoặc nội dung liên quan, và bộ chọn số dòng hiển thị, ví dụ 200 dòng, phục vụ quan sát nhiều bản ghi trong một lần tải.

Khối thao tác quản trị nằm ngay dưới bộ lọc, gồm hai trường nhập liệu:

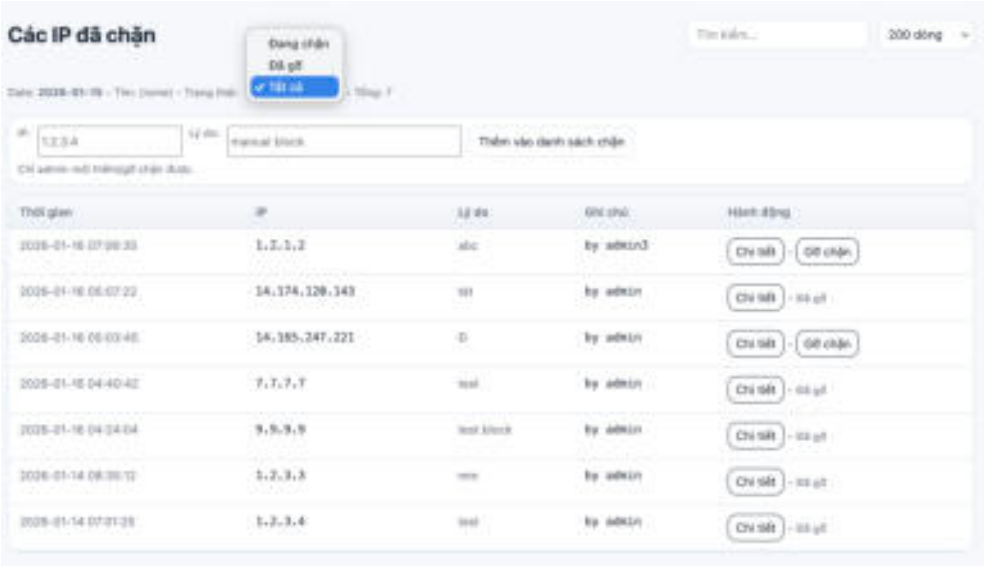
- Trường IP, cho phép nhập địa chỉ IP cần chặn thủ công.
- Trường Lý do, ghi nhận nguyên nhân chặn, ví dụ manual block, hoặc mô tả ngắn phục vụ truy vết sau này.

Sau khi nhập, nút Thêm vào danh sách chặn tạo bản ghi mới hoặc cập nhật bản ghi tương ứng. Giao diện có thông báo “Chỉ admin mới thêm gỡ chặn được”, nhằm nhấn mạnh cơ

chế phân quyền, người dùng thường chỉ có quyền xem, tài khoản admin mới có quyền thay đổi trạng thái chặn. Danh sách IP được trình bày dạng bảng, gồm các cột chính:

- Thời gian, thời điểm bản ghi chặn được tạo hoặc cập nhật
- IP, địa chỉ IP bị chặn
- Lý do, lý do chặn do hệ thống hoặc người vận hành cung cấp
- Ghi chú, thông tin bổ sung, thường dùng để lưu người thực hiện như by admin
- Hành động, cung cấp hai nút Chi tiết và Gỡ chặn

Nút Chi tiết dùng để mở hộp thoại xem đầy đủ thông tin bản ghi, phục vụ đối soát trong quá trình xử lý sự kiện. Nút Gỡ chặn chuyển IP khỏi trạng thái đang chặn, phục vụ tình huống xác minh lại và giảm tác động không cần thiết đến lưu lượng hợp lệ.



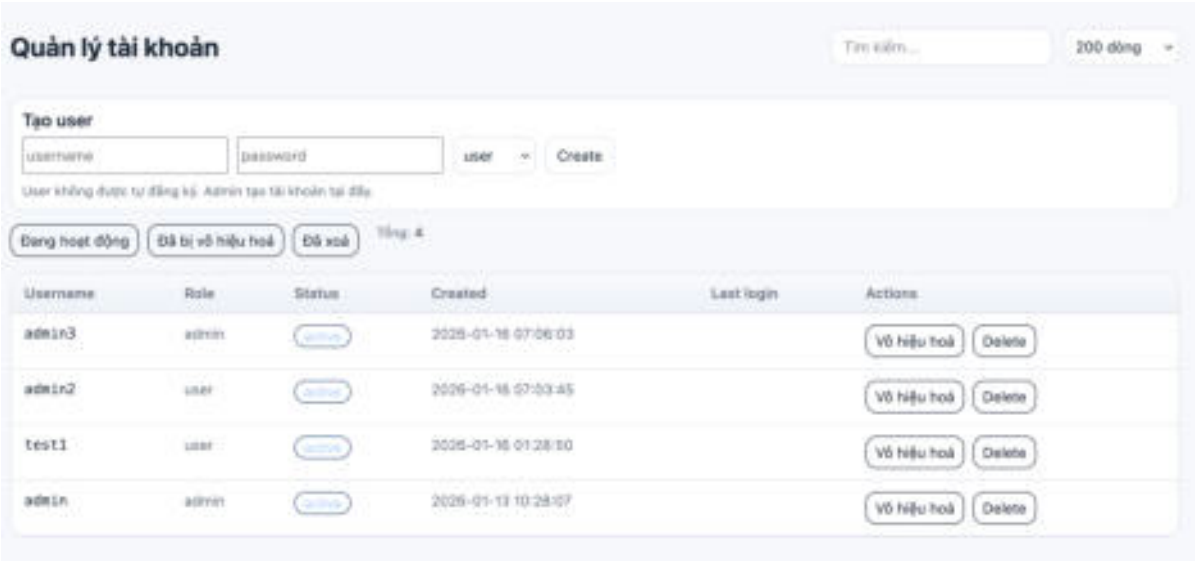
Hình 4.24: Màn hình Các IP đã chặn.

4.7.8 Màn hình Thống kê

Màn hình Thống kê tổng hợp nhanh hiệu quả vận hành theo ngày, trong đó có phần đếm số cảnh báo theo loại và phần FP rate trên các cảnh báo đã được xác minh. Cách tính FP rate dựa trên các cảnh báo đã được gắn nhãn True attack hoặc False positive, giúp phản ánh chất lượng cảnh báo sau khi đã có sự can thiệp của người vận hành. Ngoài ra, hệ thống thống kê theo period dạng day cho ba nhóm, cảnh báo tấn công, cảnh báo bất thường, và số IP bị chặn, hỗ trợ quan sát xu hướng.

Hình 4.25: Màn hình thống kê FP rate và số lượng theo ngày.

4.7.9 Màn hình Quản lý tài khoản



Hình 4.26: Màn hình quản lý tài khoản và các tab trạng thái.

Màn hình Quản lý tài khoản dành cho admin, hỗ trợ tạo user mới, phân role user hoặc admin, và quản lý vòng đời tài khoản theo ba trạng thái, đang hoạt động, đã bị vô hiệu hoá, và đã xoá. Mỗi dòng hiển thị username, role, trạng thái, thời điểm tạo, lần đăng nhập gần nhất, và các nút thao tác tương ứng như vô hiệu hoá, kích hoạt, hoặc xoá. Thiết kế tách ba tab trạng thái giúp giảm nhầm lẫn khi vận hành, đồng thời hạn chế rủi ro thao tác sai trên tài khoản đang hoạt động.

4.8 Triển khai, vận hành và giám sát

4.8.1 Mô hình triển khai trên máy chủ

Hệ thống được triển khai theo mô hình một máy chủ, chạy trên EC2 Ubuntu. Các thành phần chính gồm:

- Backend Nodejs cung cấp API
- Giao diện Web GUI phục vụ vận hành
- Các tiến trình Python để tạo bucket từ Zeek và chạy suy luận
- Cơ sở dữ liệu SQLite lưu flows, cảnh báo, review, và danh sách IP chặn.

Cấu trúc triển khai tách theo thư mục, Backend đặt trong ~/detectweb_backend, GUI đặt trong /var/www/detectweb_gui, dữ liệu log và bucket đặt theo cây thư mục ngày để dễ đối soát và tối ưu truy vấn theo thời gian.

4.8.2 Cơ chế dịch vụ và lịch chạy tự động

Để bảo đảm hệ thống chạy ổn định và tự động theo thời gian, các thành phần được cấu hình chạy dưới dạng systemd service và systemd timer:

- Nhóm thu thập và chuẩn hoá dữ liệu: Zeek ghi log theo thời gian thực, chương trình exporter đọc log và tổng hợp thành bucket theo chu kỳ, ví dụ mỗi 30 phút. Exporter chạy như một service luôn hoạt động, hoặc được kích hoạt theo lịch nếu cần giảm tài nguyên.
- Nhóm suy luận và ghi cảnh báo: Tiến trình auto detect runner chạy theo lịch, đọc bucket mới nhất, gọi module detect để tính điểm bất thường, sau đó ghi kết quả vào SQLite. Cách tổ chức này giúp tách biệt suy luận khỏi Backend, tránh làm chậm API và giúp dễ thay thế mô hình.
- Nhóm ingest vào CSDL: Các script ingest flows và ingest detect anomalies chạy theo lịch hằng ngày, đưa dữ liệu đầu vào và kết quả suy luận vào bảng tương ứng, tạo dữ liệu ổn định cho GUI truy vấn theo ngày.

Hình 4.27: Các dịch vụ chạy tự động theo lịch.

4.8.3 Vận hành log, theo dõi lỗi, và truy vết sự cố

Hệ thống sử dụng journald để lưu log của các service. Log Backend giúp theo dõi request, lỗi xác thực, lỗi truy vấn SQLite. Log exporter giúp phát hiện tình huống thiếu file Zeek, lỗi parse, hoặc thiếu quyền truy cập thư mục. Log detect runner giúp quan sát tiến trình suy luận, số lượng subject, số lượng miss, và các cảnh báo bất thường theo ngày. Khi xảy ra sự cố, quy trình truy vết ưu tiên theo thứ tự:

- Một là kiểm tra trạng thái service, để xác định thành phần nào dừng bất thường.
- Hai là kiểm tra log 200 dòng gần nhất của service đó, để xác định lỗi cấu hình, lỗi quyền, lỗi file, hoặc lỗi dữ liệu.
- Ba là đối soát dữ liệu đầu vào, bằng cách xem bucket theo cùng thời điểm và xem log Zeek tương ứng.

4.8.4 Sao lưu, phục hồi, và quản lý phiên bản dữ liệu

Do sử dụng SQLite, dữ liệu vận hành tập trung trong file app.db. Hệ thống áp dụng sao lưu định kỳ theo nguyên tắc, sao lưu trước khi migrate, sao lưu theo ngày đối với dữ liệu đang tăng, và lưu tối thiểu một số bản gần nhất để có thể quay lui khi phát sinh lỗi.

Sao lưu được thực hiện bằng cách tạo bản copy theo timestamp, ví dụ app.db.bak.YYYYMMDD_HHMMSS. Quy trình phục hồi gồm dừng Backend để tránh ghi đồng thời, thay file app.db bằng bản sao lưu, sau đó chạy kiểm tra nhanh một số truy vấn thống kê, và khởi động lại Backend. Ngoài CSDL, thư mục log Zeek và bucket cũng cần chính sách lưu giữ. Log Zeek thường tăng nhanh, do đó cần giữ theo vòng đời, ví dụ 7 ngày hoặc 14 ngày tuỳ dung lượng đĩa. Bucket là dữ liệu trung gian phục vụ detect, có thể giữ lâu hơn một chút để phục vụ điều tra, nhưng vẫn cần dọn dẹp theo lịch.

4.8.5 Quy trình cập nhật mô hình và ngưỡng phát hiện

Cập nhật mô hình được thực hiện theo quy trình có kiểm soát để tránh làm gián đoạn vận hành và tránh thay đổi ngưỡng gây tăng cảnh báo giả. Gồm có 4 bước chính:

- Bước chuẩn bị: mô hình mới được huấn luyện và lưu cùng codebook. Cần bảo đảm codebook tương thích với dữ liệu tạo chuỗi trong hệ thống, nếu thay đổi cách tạo token thì phải cập nhật đồng thời module tạo chuỗi.
- Bước kiểm tra trước triển khai: chạy suy luận thử trên một khoảng dữ liệu Zeek đã lưu, so sánh số lượng cảnh báo và phân bố score với mô hình đang chạy. Nếu số cảnh báo tăng đột biến, cần xem lại ngưỡng và cơ chế lọc subject quá ngắn.
- Bước triển khai: đặt mô hình mới vào thư mục quy ước, cập nhật đường dẫn model và codebook trong cấu hình detect. Khởi động lại tiến trình suy luận, theo dõi log ít nhất một vài chu kỳ chạy để bảo đảm không lỗi load model và không lỗi ghi SQLite.
- Bước hiệu chỉnh ngưỡng: ngưỡng rank score có thể chọn theo đỉnh F1 trên tập kiểm thử, đồng thời cần kiểm chứng lại trên dữ liệu thực tế để cân bằng giữa FP và FN. Trong vận hành, có thể áp dụng chiến lược hai ngưỡng, ngưỡng thấp để đưa vào danh sách theo dõi, ngưỡng cao để đẩy lên cảnh báo ưu tiên, cách này giúp giảm áp lực xử lý cảnh báo nhưng vẫn giữ khả năng phát hiện sớm.

4.9 Kiểm thử

4.9.1 Mục tiêu và phạm vi kiểm thử

Kiểm thử được thực hiện nhằm bảo đảm hệ thống vận hành ổn định theo chu kỳ, dữ liệu đầu vào từ Zeek được xử lý đúng thành bucket, module suy luận ghi kết quả đúng về SQLite, Backend API trả về đúng định dạng cho GUI, và các thao tác review, chặn IP hoạt động đúng theo phân quyền. Bên cạnh phần đánh giá mô hình ở mục 4.4.7, mục này tập trung vào kiểm thử mức hệ thống và phần mềm, gồm kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hiệu năng, và kiểm thử an ninh cơ bản.

4.9.2 Kiểm thử đơn vị

Kiểm thử đơn vị tập trung vào các hàm và module có tính quyết định đến tính đúng đắn của dữ liệu và tính an toàn của API. Các nhóm nội dung được kiểm thử gồm:

- Thứ nhất, kiểm thử chuẩn hoá dữ liệu và tạo khoá sự kiện. Dữ liệu flow sau khi chuẩn hoá phải tạo ra token ổn định, đúng định dạng, và tương thích với codebook. Trường hợp thiếu trường, giá trị rỗng, hoặc giá trị vượt miền hợp lệ phải được xử lý theo quy ước, không gây lỗi dây chuyền trong quá trình suy luận.

- Thứ hai, kiểm thử các hàm mã hoá chuỗi và đóng gói cửa sổ. Với cùng một chuỗi đầu vào, bước tạo cửa sổ theo seqlen phải cho kết quả nhất quán, số cửa sổ khớp với kỳ vọng, padding đúng khi chuỗi ngắn hơn seqlen, và mask không làm sai lệch vị trí dự đoán.
- Thứ ba, kiểm thử lớp bảo vệ API và kiểm tra dữ liệu đầu vào. Các hàm kiểm tra định dạng ngày, giới hạn limit, lọc tên file, và kiểm tra quyền theo role phải chặn được các request sai định dạng và các request vượt quyền. Các lỗi phải trả về đúng mã trạng thái và thông điệp gọn, để GUI xử lý được.
- Thứ tư, kiểm thử thao tác ghi đọc SQLite ở mức hàm. Các truy vấn insert, update, select quan trọng như lưu cảnh báo, cập nhật review, thêm IP chặn, gỡ chặn, phải bảo đảm không tạo bản ghi trùng, và không ghi sai trạng thái enabled.

Bảng 4.13: Danh sách kiểm thử đơn vị.

Nhóm hàm	Dữ liệu vào	Kỳ vọng	Tiêu chí đạt
Chuẩn hoá flow	flow thiếu trường, flow giá trị lạ	vẫn tạo token hợp lệ	không lỗi, token đúng định dạng
Tạo cửa sổ theo seqlen	chuỗi ngắn, chuỗi dài	số cửa sổ đúng	số cửa sổ khớp công thức, padding đúng
Kiểm tra ngày và limit	ngày sai định dạng, limit lớn	bị từ chối	trả lỗi, không truy vấn DB
Phân quyền	user gọi API admin	bị từ chối	trả lỗi, không thay đổi DB
Ghi review	id hợp lệ, status hợp lệ	cập nhật đúng	status đổi, review by và review at có giá trị

4.9.3 Kiểm thử tích hợp

Kiểm thử tích hợp kiểm tra luồng đi xuyên qua nhiều thành phần, mục tiêu là xác nhận các module ghép nối đúng và dữ liệu không bị biến dạng khi đi qua các lớp:

Luồng tích hợp chính được kiểm thử theo thứ tự.

- Luồng dữ liệu đầu vào. Zeek sinh log, exporter tạo bucket, bucket được phát hiện và đưa vào module detect, kết quả cảnh báo được ghi vào SQLite. Tiêu chí đạt là sau một chu kỳ chạy, bảng cảnh báo có bản ghi mới theo ngày, và các trường thống kê như miss, total, score trung bình có giá trị hợp lệ.

- Luồng API và GUI. GUI gọi API đăng nhập, nhận token, truy vấn danh sách cảnh báo theo ngày, mở chi tiết, thực hiện review, và thấy trạng thái cập nhật ngay trong danh sách. Tiêu chí đạt là thao tác review không làm tải lại trang, dữ liệu sau refresh vẫn giữ đúng trạng thái.
- Luồng chặn IP. Từ màn hình cảnh báo hoặc màn hình IP đã chặn, admin thêm một IP vào danh sách chặn, kiểm tra IP xuất hiện trên bảng, và kiểm tra trạng thái enabled đổi đúng khi gỡ chặn. Tiêu chí đạt là DB cập nhật đúng và giao diện phản ánh đúng ngay sau thao tác.
- Luồng xem log. GUI chọn ngày, xem danh sách file, tail log, và quan sát cập nhật gần thời gian thực nếu bật stream. Tiêu chí đạt là không bị lỗi quyền truy cập file, không bị đọc nhầm thư mục, và stream chỉ đẩy phần mới.

4.9.4 Kiểm thử hiệu năng

Kiểm thử hiệu năng tập trung vào ba điểm nghẽn, tạo dữ liệu và ghi DB, truy vấn API phục vụ GUI, và thời gian suy luận theo bucket:

- Với pipeline dữ liệu, tiêu chí chính là một chu kỳ xử lý không bị trễ so với lịch, không tăng bộ nhớ theo thời gian, và không tạo backlog file bucket chưa xử lý. Các chỉ số theo dõi gồm thời gian chạy exporter mỗi chu kỳ, thời gian suy luận mỗi chu kỳ, và số subject được xử lý.
- Với Backend API, tiêu chí chính là thời gian phản hồi ổn định khi truy vấn danh sách cảnh báo và danh sách flows theo ngày, đặc biệt khi limit đặt lớn. Các phép đo tập trung vào p50 và p95 thời gian đáp ứng, và độ ổn định khi refresh liên tục trên GUI.
- Với stream log, tiêu chí chính là không làm tăng tải CPU bất thường, không rò rỉ kết nối, và dữ liệu append không bị trùng dòng hoặc mất dòng khi file log tăng nhanh.

Bảng 4.14: Tiêu chí kiểm thử hiệu năng.

Hạng mục	Chỉ số theo dõi	Tiêu chí đạt
Exporter tạo bucket	thời gian mỗi chu kỳ, số file tạo	ổn định theo chu kỳ, không backlog
Suy luận detect	thời gian mỗi bucket, số subject	hoàn thành trong khoảng thời gian cho phép
API danh sách cảnh báo	thời gian phản hồi p95	ổn định khi refresh, không timeout

API tìm flows	thời gian phản hồi p95	ổn định theo limit hợp lý
Stream log	CPU và số kết nối mở	không tăng dần, không rò rỉ

4.9.5 Kiểm thử an ninh cơ bản

Kiểm thử an ninh cơ bản tập trung vào xác thực, phân quyền, và giảm rủi ro từ dữ liệu đầu vào:

- Về xác thực, hệ thống được kiểm tra theo các tình huống, đăng nhập sai mật khẩu, token hết hạn, token bị sửa, và gọi API không kèm token. Tiêu chí đạt là hệ thống từ chối đúng, không trả dữ liệu nhạy cảm, và không tạo tác dụng phụ lên DB.
- Về phân quyền, các API quản trị như tạo user, bật tắt user, và thêm gỡ chặn IP chỉ được phép với role admin. Tiêu chí đạt là user thường không thể gọi các API này, kể cả khi cố tình thay đổi route trên trình duyệt.
- Về đầu vào, các tham số như date, limit, file name, ip được kiểm tra ràng buộc và được xử lý an toàn. Tiêu chí đạt là không truy cập được file ngoài thư mục cho phép, và không gây lỗi truy vấn DB khi nhập chuỗi bất thường.
- Về cấu hình HTTP, kiểm tra các header bảo vệ cơ bản, và cấu hình CORS đúng nguồn truy cập dự kiến. Tiêu chí đạt là trình duyệt không chấp nhận truy cập trái phép từ nguồn không được phép.

4.9.6 Tổng hợp kết quả và nhận xét

Kết quả kiểm thử cho thấy các luồng tích hợp chính hoạt động đúng, dữ liệu được ghi nhận theo ngày, giao diện truy vấn ổn định, và vòng thao tác review cập nhật trạng thái nhất quán. Các kiểm thử an ninh cơ bản bảo đảm API không bị truy cập khi thiếu token hoặc sai role. Về hiệu năng, hệ thống đáp ứng tốt trong kịch bản vận hành một máy chủ, tuy nhiên khi tăng mạnh số lượng flows theo ngày, cần ưu tiên tối ưu truy vấn theo chỉ mục và giới hạn phân trang hợp lý để giữ thời gian phản hồi ổn định.

Bảng 4.15: Tổng hợp kết quả kiểm thử.

Nhóm kiểm thử	**Kết quả**	**Ghi chú**
Đơn vị	đạt	cần bổ sung thêm ca biên cho dữ liệu rỗng
Tích hợp	đạt	kiểm tra theo nhiều ngày khác nhau
Hiệu năng	đạt	theo dõi thêm khi dữ liệu tăng mạnh
An ninh cơ bản	đạt	có thể bổ sung rate limit khi mở rộng

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Đồ án đã xây dựng được một hệ thống hoàn chỉnh phục vụ phát hiện và phân tích các mối đe doạ an ninh mạng dựa trên học sâu, vận hành theo quy trình end to end từ thu thập dữ liệu mạng, tiền xử lý, suy luận mô hình, lưu trữ kết quả, đến hiển thị và hỗ trợ thao tác xác minh trên giao diện Web. Hệ thống tận dụng Zeek để ghi nhận lưu lượng mạng, chuyển đổi thành dữ liệu dạng bucket, sau đó áp dụng mô hình Deep LSTM Autoencoder kết hợp phân loại theo từng bước thời gian để phát hiện bất thường theo chuỗi sự kiện. Kết quả cảnh báo được quản lý theo ngày trong SQLite, hỗ trợ quá trình tra cứu, đối soát và phản ứng thông qua cơ chế review và danh sách IP chặn.

Về mặt hiệu năng mô hình, kết quả kiểm thử trên dữ liệu UNSW NB15 cho thấy mô hình có khả năng phân tách lớp tốt thông qua AUROC cao và AUPRC phù hợp với đặc thù mất cân bằng lớp. Việc lựa chọn ngưỡng rank score theo tiêu chí cực đại F1 giúp cân bằng giữa độ bao phủ phát hiện và tỉ lệ cảnh báo sai, phù hợp cho triển khai thực tế.

1. Kết quả đạt được

Sau quá trình thiết kế, huấn luyện mô hình và triển khai trên hệ thống thực tế, đề tài đã xây dựng được một pipeline hoàn chỉnh từ thu thập log đến suy luận và hiển thị cảnh báo phục vụ vận hành. Dưới đây là những kết quả chính mà em đạt được:

- Thiết kế và hiện thực hoá kiến trúc hệ thống end-to-end với các khối rõ ràng: thu thập Zeek, sinh bucket theo thời gian, suy luận bằng Python, lưu trữ bằng SQLite, Backend Node.js cung cấp API và Web GUI phục vụ vận hành. Kiến trúc này giúp tách lớp chức năng, thuận lợi cho triển khai và bảo trì.
- Xây dựng pipeline dữ liệu theo thời gian có khả năng chạy tự động, trong đó bucket được sinh theo chu kỳ và suy luận chạy theo lịch, giảm phụ thuộc thao tác thủ công. Cách tổ chức này phù hợp bối cảnh vận hành liên tục và hỗ trợ kiểm soát luồng dữ liệu theo ngày/giờ.
- Hiện thực hoá cơ chế biểu diễn chuỗi sự kiện phục vụ mô hình học sâu, gồm trừu tượng hoá bản ghi thành event ID thông qua codebook, chuẩn hoá độ dài chuỗi bằng các token đặc biệt và áp dụng masking để đảm bảo tính nhất quán giữa huấn luyện và suy luận.
- Triển khai mô hình học sâu theo kiến trúc chuỗi, bao gồm embedding, Deep LSTM Autoencoder và lớp softmax theo bước thời gian, từ đó xây dựng chỉ số

bất thường dựa trên rank/score, hỗ trợ phát hiện hành vi lệch chuẩn theo ngữ cảnh thời gian.

- Thực nghiệm và đánh giá có cơ sở định lượng, sử dụng ROC, PR và đường Precision–Recall–F1 theo ngưỡng để lựa chọn điểm làm việc. Kết quả đánh giá giúp việc chọn ngưỡng triển khai không mang tính cảm tính mà dựa trên chỉ số đo được.
- Xây dựng Web GUI phục vụ vận hành thực tế, gồm các màn hình cảnh báo, xem log Zeek, quản lý IP chặn, thống kê và quản lý tài khoản. Cơ chế review giúp đóng vòng phản hồi giữa cảnh báo tự động và xác minh thủ công, góp phần giảm false positive trong vận hành.
- Triển khai trên máy chủ thực tế và vận hành thử thành công, chứng minh khả năng tích hợp giữa mô hình học sâu và hệ thống giám sát mạng ở mức ứng dụng triển khai, không chỉ dừng ở thí nghiệm offline.

2. Những hạn chế

Do thời gian thực hiện đồ án có hạn, môi trường phần cứng triển khai còn giới hạn, và đặc thù dữ liệu mạng thực tế biến động mạnh theo bối cảnh, hệ thống vẫn còn một số hạn chế sau:

- Chưa giải quyết triệt để lệch miền dữ liệu (domain shift) giữa UNSW-NB15 và log Zeek thực tế, nên độ ổn định của ngưỡng có thể thay đổi theo giai đoạn vận hành.
- Chưa xây dựng cơ chế ngưỡng thích nghi theo nền theo thời gian, hiện vẫn thiên về ngưỡng cố định nên có thể kém phù hợp ở các thời điểm có biến động tải hoặc thay đổi hành vi người dùng.
- Khả năng giải thích cảnh báo còn hạn chế, chủ yếu dừng ở thống kê tổng hợp; việc chỉ ra nguyên nhân gốc vẫn cần điều tra thủ công do phạm vi đồ án chưa cho phép làm sâu phần explainability.
- Benchmark và đánh giá dài hạn chưa đầy đủ, chưa chạy nhiều kịch bản/dataset và chưa theo dõi nhiều tuần/tháng vì giới hạn thời gian và tài nguyên vận hành.
- Cơ chế phản ứng (chặn IP) chưa có lớp an toàn nâng cao, như whitelist/TTL/phê duyệt nhiều bước, do yêu cầu triển khai thực tế cần thêm quy trình vận hành và kiểm thử rủi ro.

3. Hướng phát triển

Trọng tâm không chỉ là cải thiện chất lượng phát hiện, mà còn tối ưu trải nghiệm vận hành (giảm nhiễu cảnh báo, tăng khả năng truy vết) và chuẩn hoá quy trình triển

khai để hệ thống có thể chạy bền vững trong thời gian dài. Đồng thời, các cải tiến cũng hướng tới việc giảm phụ thuộc vào điều tra thủ công bằng cách làm rõ nguyên nhân cảnh báo và tăng mức độ tự động hoá phản ứng. Trong thời gian tới, để nâng cao độ chính xác, tính ổn định khi vận hành thực tế và khả năng mở rộng của hệ thống, đề tài định hướng phát triển theo các nhóm hạng mục sau:

- Thứ nhất, mở rộng dữ liệu huấn luyện và fine-tune theo dữ liệu thực tế. Có thể thu thập log Zeek trực tiếp trong môi trường triển khai, xây dựng tập "normal" theo từng giai đoạn (theo tuần/tháng hoặc theo ca làm việc), sau đó tinh chỉnh mô hình và/hoặc cập nhật codebook nhằm giảm lệch miền dữ liệu và tăng tính thích nghi.
- Thứ hai, nghiên cứu cơ chế ngưỡng thích nghi theo nền và theo ngữ cảnh. Thay vì một ngưỡng cố định, có thể dùng ngưỡng theo percentile theo ngày/giờ, hoặc chiến lược hai ngưỡng (cảnh báo ưu tiên và theo dõi) để phân tầng mức độ rủi ro, giảm nhiễu trong giờ cao điểm nhưng vẫn giữ khả năng bắt bất thường.
- Thứ ba, tăng khả năng giải thích cảnh báo. Bổ sung thống kê "top token bị miss" theo cửa sổ, hoặc ánh xạ ngược token về nhóm hành vi/giao thức/nhóm trường tạo log key, từ đó giúp người vận hành hiểu nhanh yếu tố gây bất thường và rút ngắn thời gian điều tra trên log Zeek.
- Thứ tư, nâng cấp giám sát và quan sát hệ thống. Xây dựng dashboard theo thời gian thực cho số lượng bucket, số lượng subject, phân bố score, tỉ lệ cảnh báo nhầm theo ngày; đồng thời bổ sung cảnh báo vận hành khi pipeline bị trễ, khi exporter không sinh bucket đúng lịch, hoặc khi chất lượng dữ liệu đầu vào suy giảm.
- Thứ năm, tăng độ tin cậy và khả năng mở rộng. Khi dữ liệu lớn, có thể chuyển từ SQLite sang hệ quản trị phù hợp hơn; tách riêng dịch vụ suy luận theo mô hình hàng đợi để tránh nghẽn; và chuẩn hoá các quy trình backup, migrate, rollback để cập nhật mô hình an toàn.
- Thứ sáu, hoàn thiện cơ chế phản ứng. Triển khai chặn tạm thời theo TTL, danh sách an toàn (whitelist), cơ chế phê duyệt trước khi chặn trong môi trường nhạy cảm; đồng thời mở rộng phản ứng sang tạo ticket, gửi thông báo, hoặc tích hợp với SIEM để đồng bộ quy trình xử lý sự cố.

Kết lại, các hướng phát triển trên tập trung vào ba mục tiêu chính: giảm lệch miền dữ liệu và tăng thích nghi, tăng khả năng giải thích để hỗ trợ vận hành, và nâng độ tin cậy–mở rộng để triển khai dài hạn trong môi trường thực tế.

TÀI LIỆU THAM KHẢO

[1] Scarfone, K., & Mell, P. (2007). NIST SP 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards and Technology.

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf

[2] Scarfone, K., & Mell, P. (2012). NIST Special Publication 800-94 Revision 1 (Draft): Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards and Technology (NIST).

https://csrc.nist.gov/files/pubs/sp/800/94/r1/ipd/docs/draft_sp800-94-rev1.pdf

[3] Nour Moustafa and Jill Slay. "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network DataSet)". In: 2015 Military Communicationsand Information Systems Conference (MilCIS). IEEE. 2015, pp. 1–6.

https://research.unsw.edu.au/projects/unsw-nb15-dataset

[4] Cisco Systems, Inc. (2021). Overview of NetFlow (Flexible NetFlow configuration content for Cisco ASR 920 Series). Cisco Systems, Inc.

https://www.cisco.com/c/dam/en/us/td/docs/routers/asr920/configuration/guide/netmgmt/fnf-xe-3e-asr920-book.html

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

https://www.deeplearningbook.org

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. MIT Press.

https://www.bioinf.jku.at/publications/older/2604.pdf

[7] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17) (Dallas, TX, USA, Oct 30–Nov 3, 2017). ACM.

https://users.cs.utah.edu/~lifeifei/papers/deeplog.pdf

[8] Lun-Pin Yuan, Peng Liu, and Sencun Zhu. “Recomposition vs. Prediction: A Novel Anomaly Detection for Discrete Events Based On Autoencoder”. In: arXiv preprint arXiv:2012.13972 (2020).

Đường dẫn:

[9] https://docs.zeek.org/en/master/logs/index.html

[10] https://www.ezycloudx.com