

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



ĐỒ ÁN CHUYÊN NGÀNH
GIẢI PHÁP THÔNG MINH BẢO VỆ TRƯỚC MỐI ĐE DỌA
SỬ DỤNG SURICATA VÀ ELK STACK TÍCH HỢP HỌC MÁY

**INTELLIGENT THREAT PROTECTION: AI-POWERED SURICATA
AND ELK STACK**

GIẢNG VIÊN HƯỚNG DẪN:
THS. NGUYỄN CÔNG DANH

NGUYỄN TÀI HIẾU - 22520442
TRẦN HỮU HIẾU - 22520444

TP. Hồ Chí Minh - 2025

LỜI CẢM ƠN

Nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Nguyễn Công Danh, người đã tận tình hướng dẫn, đồng hành và hỗ trợ nhóm trong suốt quá trình nghiên cứu và thực hiện đồ án chuyên ngành. Những ý kiến đóng góp quý báu, sự chỉ bảo tận tâm cùng với kinh nghiệm chuyên môn sâu rộng của thầy đã giúp nhóm định hướng rõ ràng, vượt qua nhiều khó khăn và hoàn thiện dự án một cách tốt nhất.

Bên cạnh đó, chúng em cũng xin bày tỏ lòng biết ơn đến các thầy cô trong bộ môn, những người đã luôn nhiệt tình giảng dạy, chia sẻ kiến thức, định hướng phương pháp nghiên cứu và không ngừng động viên nhóm trong suốt hành trình thực hiện đồ án. Sự hỗ trợ và khích lệ từ các thầy cô đã tiếp thêm động lực để nhóm nỗ lực hoàn thành công việc với kết quả tốt nhất. Một lần nữa, nhóm xin gửi lời tri ân sâu sắc đến tất cả các thầy cô đã góp phần vào sự thành công của đồ án này.

Nguyễn Tài Hiếu

Trần Hữu Hiếu

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI.....	11
1.1. Mục tiêu nghiên cứu.....	11
1.2. Đối tượng nghiên cứu.....	11
1.3. Phạm vi nghiên cứu.....	12
1.4. Các nghiên cứu liên quan.....	12
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	14
2.1. Suricata	14
2.2. ELK.....	15
2.3. Filebeat	17
2.4. Winlogbeat.....	18
2.5. MITRE ATT&CK.....	19
2.6. AI-based IDS.....	21
CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT.....	23
3.1. Mô hình tổng quan.....	23
3.2. Mô hình ứng dụng	24
CHƯƠNG 4. TRIỂN KHAI MÔ HÌNH VÀ THỰC NGHIỆM	26
4.1. Kiến trúc hệ thống.....	26
4.2. Cài đặt và cấu hình công cụ.....	28
4.2.1. Suricata.....	28
4.2.2. ELK Stack.....	30
4.2.3. Filebeat	33
4.2.4. Winlogbeat.....	34
4.2.5. LogAlert gửi mail.....	35
4.2.6. Router	40
4.2.7. AI-based IDS.....	41
4.2.7.1. Huấn luyện và đánh giá tính khả thi của mô hình:.....	41
4.2.7.2. Đánh giá mô hình	49
4.3. Mô phỏng tấn công.....	64
4.3.1. Tấn công Cross-site scripting, SQL injection và Brute force mật khẩu.....	64
4.3.2. Tấn công malware mô phỏng WannaCry.....	69

4.3.3. Kiểm thử hệ thống AI-based IDS.....	71
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	79
1. Kết luận.....	79
2. Hướng phát triển.....	79
TÀI LIỆU THAM KHẢO.....	80

DANH MỤC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

IDS	Intrusion Detection System
IPS	Intrusion Prevention System
OISF	Open Information Security Foundation
ELK	Elasticsearch, Logstash, Kibana
SIEM	Security Information and Event Management
TLS/SSL	Transport Layer Security / Secure Sockets Layer
API	Application Programming Interface
MITRE ATT&CK	MITRE Adversarial Tactics, Techniques, and Common Knowledge
SSH	Secure Shell
DMZ	Demilitarized Zone
NAT	Network Address Translation

MỤC LỤC HÌNH ẢNH

Hình 1. Logo biểu tượng của công cụ Suricata.....	14
Hình 2. Chức năng của Suricata.....	15
Hình 3. Cấu trúc của hệ thống ELK stack.....	16
Hình 4. Luồng hoạt động của ELK stack.....	16
Hình 5. Logo biểu tượng của Filebeat.....	17
Hình 6. Sơ đồ kết hợp Filebeat vào ELK	18
Hình 7. MITRE ATT&CK.....	20
Hình 8. MITRE ATT&CK Matrix for Enterprise.....	21
Hình 9. Kiến trúc hệ thống IDS AI-BASED.....	21
Hình 10. Mô hình kiến trúc mạng doanh nghiệp được triển khai để kiểm thử... <td>23</td>	23
Hình 11. Mô hình biểu diễn luồng hoạt động của quá trình giám sát an ninh mạng.....	24
Hình 12. Câu hình Virtual Network Editor của máy.	26
Hình 13. Câu hình của máy router.	27
Hình 14. Câu hình của máy Suricata.....	27
Hình 15. Câu hình máy ELK.....	27
Hình 17. Câu hình của máy Webserver.	28
Hình 19. Thông báo sau khi đã cài đặt thành công elasticsearch.	31
Hình 20. Lệnh curl kiểm tra elasticsearch đã hoạt động và trả về phản hồi.....	32
Hình 21. Cài đặt thành công Elasticsearch và Kibana.	33
Hình 22. Trang chủ cài đặt Winlogbeat.	34
Hình 23. Cài đặt Winlogbeat.....	35
Hình 24. Chính sửa host và password của server Elastic.....	35
Hình 25. Chọn rule Logthreshold.....	36
Hình 26. Câu hình Rule cảnh báo theo threshold.	36
Hình 27. App password trên gmail của quản trị viên.....	38
Hình 28. Câu hình file config logalert.	38
Hình 29. Dịch vụ Logalert đã khởi chạy thành công.....	40
Hình 30. Ping tới google.com trên máy trong mạng nội bộ.....	41
Hình 31. Các thư viện python cần thiết.....	42
Hình 32. Gộp dữ liệu trong dataset CIC-IDS-2017.	42

Hình 33. In ra số lượng các giá trị bị thiếu.	42
Hình 34. Thêm giá trị trung bình vào chỗ bị thiếu.....	43
Hình 35. Kiểm tra lại sau khi xử lý các giá trị bị thiếu.....	43
Hình 36. In ra số lượng các loại tấn công.	43
Hình 37. Chuyển đổi các nhãn tấn công dạng chữ thành dạng số.	44
Hình 38. In ra danh sách các loại tấn công tương ứng với mỗi giá trị số	44
Hình 39. Loại bỏ các cột chỉ có một giá trị duy nhất.....	44
Hình 40. Sau khi loại bỏ các cột không cần thiết.....	45
Hình 41. Chuẩn hóa dữ liệu.	45
Hình 42. Giảm chiều dữ liệu.	46
Hình 43. Dữ liệu mới sau khi giảm chiều.	46
Hình 44. Chọn lọc dữ liệu dữ liệu.....	47
Hình 45. Cân bằng dữ liệu	47
Hình 46. Chia dữ liệu để huấn luyện.....	48
Hình 47. Khởi tạo và huấn luyện mô hình Random Forest 1.	48
Hình 48. Kết quả huấn luyện mô hình 1.	49
Hình 49. Khởi tạo và huấn luyện mô hình Random Forest 2.	49
Hình 50. Kết quả huấn luyện mô hình 2.	49
Hình 51. ROC và AUC của một mô hình hoàn hảo giả định.....	53
Hình 52. Bảng so sánh ma trận nhầm lẫn (confusion matrix) của hai mô hình..	54
Hình 53. Bảng trực quan hóa chỉ số hiệu suất quan trọng của hai mô hình . ..	54
Hình 54. Bảng so sánh độ chính xác của hai mô hình.	55
Hình 55. Bảng so sánh chỉ số Cross Validation Score của hai mô hình.....	55
Hình 56. Bảng so sánh ROC Curve của hai mô hình.....	56
Hình 57. Script python thu thập dữ liệu mạng.....	58
Hình 58. Thu thập dữ liệu mạng trong Lab.....	58
Hình 59. Dữ liệu mạng được đánh dấu và lưu vào trong file .csv.....	58
Hình 60. Class Flow - Quản lý trạng thái luồng mạng.	59
Hình 61. Class IDSDetector – Bộ não của hệ thống.....	61
Hình 62. Hàm khởi tạo.....	61
Hình 63. Hàm run().	62
Hình 64. Hàm block_worker().	62

Hình 65. Hàm process_alert()	63
Hình 66. Code thực hiện chặn IP.	63
Hình 67. Kết quả chạy Script để chặn IP kẻ tấn công.....	64
Hình 68. Script python tấn công XSS.	65
Hình 69. Script python tấn công SQL injection.....	65
Hình 70. Script tấn công brute force mật khẩu.	66
Hình 71. Rule cảnh báo trên Suricata.....	66
Hình 72. Suricata đã phát hiện thành công và ghi log lại.	66
Hình 73. Log của cuộc tấn công XSS được lưu trữ trên Elasticsearch.....	67
Hình 74. Log của cuộc tấn công SQL injection được lưu trữ trên Elasticsearch.	67
Hình 75. Log của cuộc tấn công Brute force được lưu trữ trên Elasticsearch....	67
Hình 76. Người quản trị đã nhận được Gmail cảnh báo sau khi có cuộc tấn công.	68
Hình 78. Chức năng hỗ trợ người quản trị đánh giá mức độ rủi ro của các cuộc tấn công.	69
Hình 79. Các kỹ thuật trong bảng MITRE ATT&CK.	69
Hình 80. Mã độc được khởi chạy và máy nạn nhân đang bị tấn công.....	70
Hình 81. Registry của máy nạn nhân sau khi chạy mã độc.	70
Hình 82. Log của công cụ giám sát sysmon.	70
Hình 83. Log của Win10 đã được Winlogbeat gửi về Elasticsearch.	71
Hình 84. Chạy Script phát hiện tấn công	72
Hình 85. Sử dụng GoldenEye để tấn công.....	72
Hình 86. Module bắt được luồng gói tin tấn công DoS.....	73
Hình 87. Đưa ra cảnh báo về tấn công.....	73
Hình 88. Cảnh báo tấn công trên Kibana.	74
Hình 89. Chạy script gửi lệnh chặn ip của kẻ tấn công đến máy switch.	74
Hình 90. Rule chặn IP của kẻ tấn công được thêm vào Iptables của máy Switch.	75
Hình 91. Dùng netcat để tạo server nhận file từ máy Victim.	75
Hình 92. Thực hiện chuyển file đến cho máy Attacker trên máy Server.	75
Hình 93. Module bắt được luồng dữ liệu của cuộc tấn công.	76
Hình 94. Mô hình phát hiện và gửi cảnh báo đến Kibana.	76

Hình 95. Cảnh báo trên Kibana.....	77
Hình 96. Chạy Script gửi lệnh chặn IP kẻ tấn công đến máy Switch.	77
Hình 97. Rule chặn IP của kẻ tấn công được thêm vào Iptables trên máy Switch.	
.....	78

TÓM TẮT ĐỀ TÀI

Trong bối cảnh hiện tại, các tổ chức trên toàn cầu đang phải đối mặt với những thách thức bảo mật trong môi trường mạng ngày càng nghiêm trọng do sự gia tăng nhanh chóng của các cuộc tấn công mạng tinh vi, bao gồm mã độc tiên tiến, tấn công APT, và các mối đe dọa chưa từng được ghi nhận trước đây. Một vấn đề cấp bách là sự thiếu hụt trong việc giám sát và phân tích dữ liệu log một cách hiệu quả, cùng với phản ứng chậm trễ trước các sự cố bảo mật do không có công cụ giám sát thời gian thực. Các hệ thống truyền thống thường gặp khó khăn trong việc xử lý khối lượng dữ liệu log khổng lồ từ nhiều nguồn khác nhau, dẫn đến việc bỏ sót các dấu hiệu tấn công quan trọng và kiến trúc mạng dễ tổn thương trước các mối đe dọa hiện đại.

Đề tài “Intelligent Threat Protection: AI-Powered Suricata and ELK Stack” nhằm giải quyết những thách thức này bằng cách tích hợp trí tuệ nhân tạo (AI) với các công cụ bảo mật như Suricata và ELK Stack (Elasticsearch, Logstash, Kibana), tạo ra một hệ thống bảo vệ thông minh và toàn diện.

Giải pháp này không chỉ tập trung vào việc quản lý log tập trung từ nhiều nguồn như Webserver, Suricata và máy Windows, mà còn nâng cao khả năng giám sát lưu lượng mạng theo thời gian thực thông qua vai trò của Suricata như một hệ thống phát hiện xâm nhập (IDS). Bên cạnh đó, việc tích hợp công cụ LogAlert cho phép các quản trị viên nhận thông báo tức thì về các hoạt động tấn công đáng ngờ, từ đó rút ngắn thời gian phản ứng. Đối với các cuộc tấn công phức tạp vượt quá khả năng phát hiện của các phương pháp truyền thống, hệ thống AI-based IDS được triển khai để phân tích sâu và nhận diện các mối đe dọa tinh vi mà Suricata có thể bỏ qua. Đồng thời, ELK Stack đóng vai trò quan trọng trong việc trực quan hóa dữ liệu log thông qua các sơ đồ, biểu đồ, và ánh xạ chung với các kỹ thuật trong khung MITRE ATT&CK, giúp tổ chức hiểu rõ hơn về bản chất và phạm vi của các cuộc tấn công.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Chương này tóm tắt mục tiêu nghiên cứu, đối tượng và phạm vi của đề tài mà nhóm đang thực hiện. Đồng thời nêu được các mục quan trọng sẽ đề cập ở các chương.

1.1. Mục tiêu nghiên cứu.

Mục tiêu nghiên cứu là phát triển một hệ thống bảo vệ mạng hiện đại và hiệu quả bằng cách tích hợp các công cụ tiên tiến như Suricata, ELK Stack và một công cụ AI-based IDS. Nghiên cứu hướng đến việc giải quyết các vấn đề về giám sát và bảo mật truyền thông, bao gồm quản lý log tập trung, giám sát lưu lượng thời gian thực, phát hiện các cuộc tấn công phức tạp, và tối ưu hóa quá trình xử lý dữ liệu log. Cụ thể, nhóm nhằm nâng cao khả năng phát hiện và phản ứng trước các mối đe dọa mạng, giảm thiểu sai sót thủ công, và cung cấp các công cụ trực quan hóa thông minh để hỗ trợ quản trị viên trong việc xây dựng chiến lược phòng thủ hiệu quả. Kết quả mong đợi là một giải pháp toàn diện, có khả năng thích ứng với các thách thức bảo mật ngày càng gia tăng trong môi trường doanh nghiệp hiện đại.

1.2. Đối tượng nghiên cứu.

Đối tượng nghiên cứu của đề tài tập trung vào các hệ thống mạng doanh nghiệp, đặc biệt là các môi trường có sử dụng Webserver, hệ điều hành Windows, và các thiết bị mạng cần giám sát lưu lượng. Nghiên cứu cũng bao gồm các nguồn log từ Suricata và các công cụ liên quan, cũng như các cuộc tấn công mạng phức tạp như DDoS, Data Exfiltration, APT, và mã độc tiên tiến. Ngoài ra, đối tượng nghiên cứu mở rộng đến các công cụ quản lý và phân tích log như ELK Stack, cùng với sự tích hợp của công cụ AI-based IDS để nâng cao hiệu quả phát hiện và giám sát. Đối tượng này được lựa chọn nhằm đáp ứng nhu cầu bảo mật thực tế của các tổ chức, nơi mà việc bảo vệ dữ liệu và hệ thống mạng là ưu tiên hàng đầu.

1.3. Phạm vi nghiên cứu.

Xây dựng mô hình hệ thống mạng mô phỏng một doanh nghiệp gồm các máy quản lý và các máy người dùng đang hoạt động. Triển khai các công cụ giám sát lưu lượng mạng của hệ thống đó bằng Suricata, ELK và AI-based IDS. Thực hiện các cuộc tấn công cơ bản đến nâng cao nhằm vào hệ thống mạng doanh nghiệp để đánh giá khả năng phát hiện của các công cụ đã triển khai. Loại tấn công bao gồm: Cross-site scripting, SQL injection, Brute force mật khẩu, Malware mô phỏng Wanna Cry và Data Exfiltration Linux.

1.4. Các nghiên cứu liên quan.

Trong bài nghiên cứu [1], nhóm tác giả đã xây dựng một giải pháp SIEM có chi phí thấp dành cho các doanh nghiệp nhỏ - những đối tượng tiêu nguồn nhân lực tài chính và kỹ thuật để triển khai các hệ thống an ninh mạng phức tạp. Dựa trên lý thuyết Cyber Kill Chain, tác giả mô phỏng toàn bộ chuỗi tấn công mạng (từ do thám đến chiếm quyền điều khiển) và đánh giá khả năng phát hiện của hệ thống qua giải pháp SIEM đã thực nghiệm. Thông qua việc thiết kế các rules tùy chỉnh, nghiên cứu kiểm chứng rằng hệ thống có thể phát hiện hiệu quả các hành vi tấn công như quét cổng, khai thác lỗ hổng, cài đặt mã độc, tạo persistence qua registry và thiết lập kết nối điều khiển từ xa (C2).

Hệ thống được xây dựng bằng các công cụ mã nguồn mở như Suricata, Filebeat, Winlogbeat, và ELK Stack, triển khai trên môi trường ảo hóa cấu hình thấp nhằm đảm bảo phù hợp với điều kiện thực tế của doanh nghiệp nhỏ. Kết quả cho thấy hệ thống có khả năng phát hiện mối đe dọa theo thời gian thực, hoạt động ổn định mà không yêu cầu phần cứng mạnh, đồng thời cung cấp trực quan hóa dữ liệu rõ ràng qua Kibana, giúp cả những người không có chuyên môn về an ninh mạng cũng có thể theo dõi và phân tích sự cố.

Ngoài ra, tác giả cũng đề cập đến vai trò hỗ trợ của AI trong việc tạo rules phát hiện, phân tích hành vi dựa trên log và gợi ý cấu hình phù hợp. Việc tích hợp SOAR được đề cập nhằm tự động hóa phản ứng (ví dụ: chặn IP, khóa tài

khoản,...), giảm thiểu phụ thuộc vào sự giám sát thủ công. Nghiên cứu cũng khuyến nghị ứng dụng mô hình học máy học các hành vi bất thường, từ đó giảm cảnh báo sai (false positives) – một vấn đề phổ biến trong hệ thống SIEM. Cuối cùng, việc sử dụng threat intelligence feeds và container hóa hệ thống bằng Docker được xem là những phát triển tiềm năng để tăng cường khả năng mở rộng và ứng phó với mối đe dọa trong tương lai.

1.5. Cấu trúc đồ án.

Nhóm xin trình bày đồ án chuyên ngành theo cấu trúc như sau:

- **Chương 1: TỔNG QUAN ĐỀ TÀI**

Trình bày cấu trúc, mục tiêu, đối tượng và phạm vi nghiên cứu mà nhóm đề xuất.

- **Chương 2: CƠ SỞ LÝ THUYẾT**

Trình bày các khái niệm, kiến thức liên quan và khái quát về các công cụ trong quá trình triển khai và thực nghiệm.

- **Chương 3: PHƯƠNG PHÁP ĐỀ XUẤT**

Chương này sẽ nói chi tiết về mô hình, luồng hoạt động mà hệ thống của đề tài hướng đến, tổng hợp các phương pháp được thực hiện trong hệ thống, phương pháp đánh giá cho mô hình.

- **Chương 4: TRIỂN KHAI MÔ HÌNH VÀ THỰC NGHIỆM**

Chương này sẽ triển khai cài đặt các công cụ và tấn công thử nghiệm nhằm đánh giá hiệu suất của mô hình mạng tích hợp.

- **Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Đưa ra kết luận về đề tài, nhận xét về ưu và nhược điểm của hệ thống, đề xuất một số hướng phát triển mở rộng cho các nghiên cứu liên quan trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này trình bày cơ sở lý thuyết của nghiên cứu. Bao gồm khái niệm, phân loại và cách hoạt động của Suricata, ELK stack, MITRE ATT&CK, Winlogbeat, Filebeat, LogAlert và AI-based IDS.

2.1. Suricata.

Suricata là một công cụ giám sát an ninh mạng, phát hiện xâm nhập (IDS) và ngăn chặn xâm nhập (IPS) nâng cao hiệu quả. Đây là mã nguồn mở được sử dụng bởi một tổ chức phi lợi nhuận do Open Information Security Foundation (OISF) phát triển [2].

Dựa trên các quy tắc và chữ ký định nghĩa sẵn, Suricata có thể nhanh chóng phát hiện các cuộc tấn công mạng. Nhờ vào khả năng phân tích sâu các gói (DPI – Deep Packet Inspection) và xử lý chúng theo thời gian thực. Không chỉ giúp phát hiện hoạt động đáng ngờ, Suricata còn tạo cảnh báo và chặn các gói tin độc hại, bảo vệ mạng trước các mối đe dọa tiềm ẩn.



Hình 1. Logo biểu tượng của công cụ Suricata¹.

Một số điểm nổi bật của Suricata:

- Tích hợp tốt các hệ thống khác: Suricata có thể làm việc cùng với các công cụ khác như ELK stack, SIEM để phân tích log, giúp cung cấp các báo cáo

¹ <https://suricata.io/branding-images/>

chi tiết hơn về tình trạng an ninh mạng.

- Chế độ vận hành đa nhiệm: Suricata có thể hoạt động ở nhiều chế độ khác nhau như IDS (Hệ thống phát hiện xâm nhập), IPS (Hệ thống ngăn chặn xâm nhập) và Network Security Monitoring (Giám sát an ninh mạng).
- Hỗ trợ cho các giao thức mạng phức tạp: Suricata có thể phân tích nhiều giao thức, từ HTTP, TLS/SSL, SMB và SSH giúp nhận diện các cuộc tấn công ẩn mình trong các giao thức [3].



Hình 2. Chức năng của Suricata².

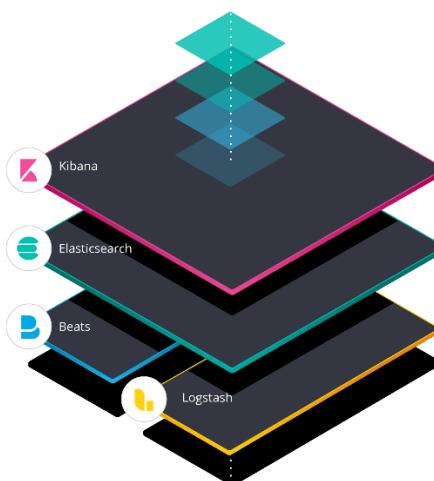
2.2. ELK.

ELK là từ viết tắt của ba dự án mã nguồn mở: Elasticsearch, Logstash và Kibana. Trong đó:

- Elasticsearch: là một search engine đóng vai trò là một kho chứa để lưu trữ logs đồng thời khả năng tìm kiếm và phân tích mạnh mẽ.
- Logstash: là một công cụ sử dụng để thu thập, xử lý log được viết bằng java. Nhiệm vụ chính của logstash là thu thập log sau đó chuyển vào Elasticsearch. Mỗi dòng log của logstash được lưu trữ dưới dạng json.
- Kibana: là công cụ cho phép trực quan hóa dữ liệu từ Elasticsearch, ở đây

² <https://suricata.io/>

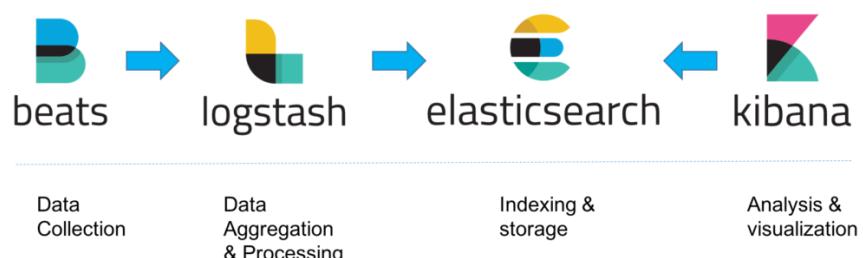
là log từ Logstash gửi về.



Hình 3. Cấu trúc của hệ thống ELK stack³.

Luồng hoạt động:

1. Đầu tiên, log sẽ được đưa đến Logstash thông qua nhiều con đường ví dụ như server gửi UDP request chứa log tới URL của Logstash, hoặc Beat đọc file log và gửi đến Logstash.
2. Logstash sẽ đọc những log này, thêm những thông tin như thời gian, IP, parse dữ liệu từ log (server nào, độ nghiêm trọng, nội dung log) ra, sau đó ghi xuống database là Elasticsearch.
3. Khi muốn xem log, người dùng vào URL của Kibana. Kibana sẽ đọc thông tin log trong Elasticsearch, hiển thị lên giao diện cho người dùng query và xử lý.



Hình 4. Luồng hoạt động của ELK stack⁴.

³ <https://www.omniwaresoft.com.tw/product-news/elastic-news/elk-what-is-elk-stack/>

⁴ <https://kb.pavietnam.vn/elk-stack-la-gi.html>

Khả năng của ELK stack:

- Quản lý log tập trung: Thay vì phải lợ mợ vào từng servers xem log thì chúng ta chỉ cần mở kibana trên trình duyệt web là có thể xem được log của tất cả các servers.
- Dễ dàng tích hợp: ELK dễ dàng tích hợp các ứng dụng như Nginx hay Apache, MSSQL, MongoDB hay Redis, Logstash đều có thể đọc hiểu và xử lý log.
- Search và filter mạnh mẽ: Elasticsearch cho phép lưu trữ thông tin kiểu NoSQL, hỗ trợ Full-Text Search nên việc query log dễ dàng và mạnh mẽ, ngay cả với dữ liệu log cực lớn.
- Scale dễ dàng: Khi muốn xử lý nhiều log hơn, chúng ta chỉ việc tăng số nodes của Elasticsearch hoặc Logstash [4].

2.3. Filebeat.

Filebeat là một công cụ thu thập log nhẹ (lightweight shipper), được thiết kế để thu thập, xử lý, và gửi dữ liệu log từ các nguồn khác nhau trong hệ thống đến các đích như Elasticsearch hoặc Logstash. Trong Elastic Stack, Filebeat đóng vai trò là "người vận chuyển" (shipper), thu thập log từ các file log, ứng dụng hoặc hệ thống container, sau đó chuyển chúng đến Elasticsearch để lưu trữ và phân tích hoặc đến Logstash để xử lý thêm. Nó đảm bảo rằng dữ liệu log luôn được cập nhật theo thời gian thực và sẵn sàng để phân tích trong Kibana [5].

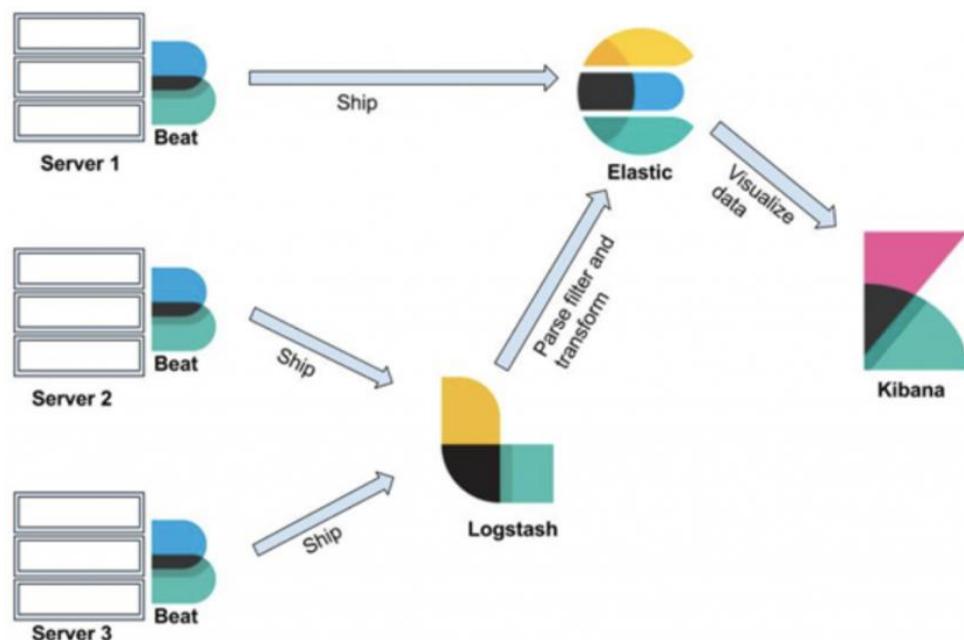


Hình 5. Logo biểu tượng của Filebeat⁵.

⁵ <https://medium.com/@luisalbertotaveras9/how-to-effortlessly-install-filebeat-and-enable-their-modules-09d6f81b7ec9>

Một số tính năng nổi bật của Filebeat:

- Module có sẵn của Filebeat như Nginx, Apache, MySQL,...) được thiết kế để thu thập và xử lý log từ các dịch vụ phổ biến, dễ dàng tích hợp và giảm thời gian cấu hình.
- Filebeat có khả năng chịu tải cao và hiệu năng tối ưu. Filebeat được tối ưu hóa để xử lý các luồng log lớn mà không làm giảm hiệu năng của hệ thống. Dù bạn đang quản lý một ứng dụng nhỏ hay một hệ thống lớn với hàng triệu sự kiện mỗi giây, Filebeat vẫn hoạt động ổn định.
- Khả năng bảo mật và xử lý lỗi của Filebeat. Filebeat sử dụng cơ chế đánh dấu vị trí log đã đọc (Registry), giúp đảm bảo rằng không có log nào bị bỏ sót, ngay cả khi hệ thống khởi động lại hoặc gặp sự cố. Filebeat hỗ trợ các giao thức mã hóa như TLS/SSL, đảm bảo rằng dữ liệu log được bảo vệ trong suốt quá trình truyền tải [6].



Hình 6. Sơ đồ kết hợp Filebeat vào ELK⁶.

2.4. Winlogbeat.

Winlogbeat là một công cụ mã nguồn mở được thiết kế để thu thập và

⁶ <https://logstash.com/blog/what-is-filebeat-and-why-is-it-important/>

chuyển tiếp nhật ký sự kiện của Windows (Windows Event Logs) để gửi sang Elasticsearch hoặc Logstash để phân tích và giám sát. Nó hoạt động như một “log shipper” nhẹ, giúp thu thập dữ liệu từ các nhật ký sự kiện của Windows, như sự kiện ứng dụng, bảo mật, hệ thống hoặc phần cứng.

Các đặc điểm chính của Winlogbeat:

- Winlogbeat sử dụng các API của Windows để đọc từ một hoặc nhiều nhật ký sự kiện (event logs) như Application, Security, System hoặc các kênh tùy chỉnh như Microsoft-Windows-Sysmon/Operational.
- Winlogbeat theo dõi vị trí đọc trong nhật ký sự kiện và lưu trữ thông tin này trên đĩa, cho phép tiếp tục thu thập dữ liệu sau khi khởi động lại mà không bỏ sót sự kiện.
- Winlogbeat theo dõi vị trí đọc trong nhật ký sự kiện và lưu trữ thông tin này trên đĩa, cho phép tiếp tục thu thập dữ liệu sau khi khởi động lại mà không bỏ sót sự kiện.

2.5. MITRE ATT&CK.

MITRE ATT&CK Framework ra đời vào năm 2013 bởi MITRE (một công ty có trụ sở tại Mỹ) với mục đích tạo ra một tài liệu toàn diện về các chiến thuật, kỹ thuật cũng như quy trình mà những kẻ tấn công mạng thường sử dụng. Từ đó, nó đã trở thành một cơ sở tri thức giúp tiêu chuẩn hóa an ninh phòng thủ và tất cả các chuyên gia an ninh mạng đều có thể truy cập được [7].

MITRE ATT&CK là viết tắt của: Adversarial Tactics Techniques & Common Knowledge.



Hình 7. MITRE ATT&CK⁷.

Ma trận ATT&CK bao gồm hai phần chính: tactics (các chiến thuật) và techniques (các kỹ thuật).

Chiến thuật là các loại tấn công tập trung vào mục tiêu chính để đạt được mục đích tấn công. Hiện tại ATT&CK có 14 loại chiến thuật:

- **Reconnaissance** - Thu thập thông tin về mục tiêu
- **Resource Development** - Thu thập các tài nguyên có thể được sử dụng để tấn công, chẳng hạn như cơ sở hạ tầng, tài khoản
- **Initial Access** - Truy cập ban đầu
- **Execution** - Thực thi mã độc
- **Persistence** - Giữ quyền truy cập vào hệ thống mục tiêu
- **Privilege Escalation** - Leo thang đặc quyền
- **Defense Evasion** - Tránh phát hiện
- **Credential Access** - Đánh cắp tên người dùng và mật khẩu
- **Discovery** - Khám phá hệ thống và mạng nội bộ mục tiêu
- **Lateral Movement** - Mở rộng phạm vi khai thác
- **Collection** - Thu thập dữ liệu mục tiêu để đánh cắp hoặc thao túng

⁷ <https://blog.kaymera.com/industry-news-and-articles/what-is-the-mitre-attck-framework>

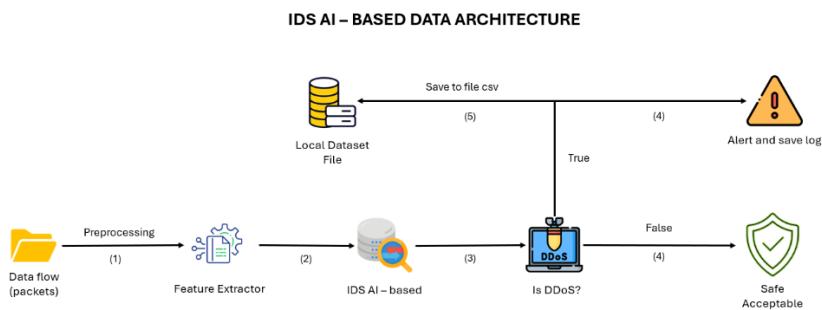
- **Command and Control** - Kết nối với các hệ thống bị xâm nhập để kiểm soát chúng
- **Exfiltration** - Đánh cắp dữ liệu thu thập được
- **Impact** - Thay đổi hoặc phá hủy dữ liệu của mục tiêu



Hình 8. MITRE ATT&CK Matrix for Enterprise⁸.

2.6. AI-based IDS.

AI-based IDS là một module được phát triển riêng bằng Python, sử dụng mô hình học máy Random Forest đã được huấn luyện để phân tích và phân loại các luồng traffic mạng trong thời gian thực [8].



Hình 9. Kiến trúc hệ thống IDS AI-BASED.

Giám sát và phản ứng:

- Traffic mạng từ các máy chủ và máy khách được sao chép (mirror) thông

⁸ <https://www.trellix.com/security-awareness/threat-intelligence/what-is-mitre-attack-framework/>

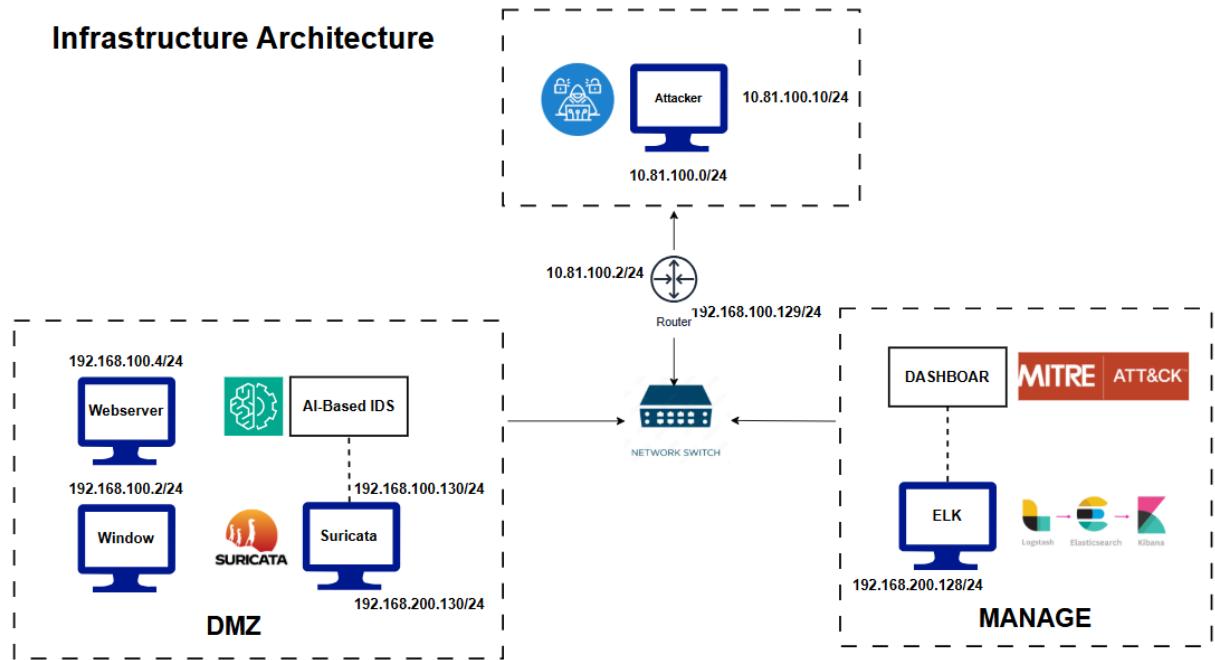
qua một máy ảo Switch trung tâm và gửi đến máy chủ giám sát.

- Module AI phân tích bản sao của traffic, đảm bảo không ảnh hưởng đến hiệu năng mạng.
- Dữ liệu mạng được trích xuất các đặc trưng để đưa vào model AI, từ đó dự đoán đó là traffic lành tính (BENIGN) hay là tấn công (ATTACK).
- Các cảnh báo từ hệ thống được ghi lại dưới dạng log có cấu trúc (JSON) và được Filebeat thu thập, gửi đến Elasticsearch để lưu trữ và đánh chỉ mục.
- Kibana đóng vai trò là giao diện trung tâm, giúp trực quan hóa, truy vấn và phân tích các cảnh báo từ cả hai nguồn, tạo thành một hệ thống giám sát an ninh tập trung (SIEM).
- Một Agent Phản ứng Chủ động (Active Response Agent) độc lập truy vấn các cảnh báo từ Elasticsearch và tự động thực thi lệnh chặn IP của kẻ tấn công trên máy Switch thông qua kết nối SSH.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

Chương này sẽ trực quan hóa mô hình mạng doanh nghiệp qua 2 sơ đồ chính giúp mô phỏng một mô hình mạng doanh nghiệp giống như trong thực tế.

3.1. Mô hình tổng quan



Hình 10. Mô hình kiến trúc mạng doanh nghiệp được triển khai để kiểm thử.

Mô hình mạng doanh nghiệp được triển khai với các thành phần chính được phân chia thành các vùng mạng như DMZ (Demilitarized Zone) và khu vực quản lý (Manage). Dưới đây là mô tả chi tiết về từng máy và tính năng được thể hiện:

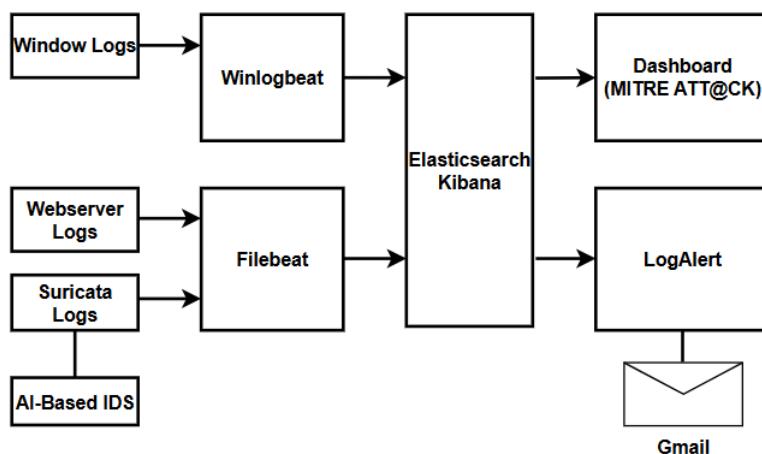
- **Vùng mạng DMZ (192.168.100.0/24):**

- Webserver (192.168.100.4) là máy chủ cung cấp dịch vụ web cho người dùng bên ngoài truy cập. Trên máy chủ này có tạo một trang web DVWA - trang web hỗ trợ trong việc kiểm thử các lỗ hổng bảo mật liên quan đến Web Application. Đây là mục tiêu tấn công của kẻ tấn công nhằm khai thác dữ liệu.
- Máy Windows (192.168.100.2) là máy tính chạy hệ điều hành Windows 10. Đây là mục tiêu nhằm kiểm thử khả năng phát hiện hoạt động đáng ngờ của mã độc trên hệ điều hành Windows.

- Máy Suricata (192.168.200.130) là công cụ hoạt động chế độ IDS (Instruction Detection) nhằm giám sát lưu lượng vào vùng mạng DMZ. Trên máy Suricata cũng cài đặt công cụ AI-based IDS do nhóm nghiên cứu sử dụng thuật toán AI để hỗ trợ Suricata trong quá trình phát hiện hành vi bất thường hoặc mối đe dọa.
- **Vùng mạng Manage (192.168.200.0/24):**
 - Máy server ELK (192.168.200.128) nhận log từ winlogbeat và filebeat để phân tích và hiển thị dashboard. Hỗ trợ ánh xạ đến MITRE ATT&CK.
- **Vùng mạng trung tâm:**
 - Router (192.168.100.128) đóng vai trò định tuyến kết nối 2 vùng mạng nội bộ với Internet và điều hướng lưu lượng mạng. Áp dụng chính sách bảo mật (như NAT, firewall rules).
- **Vùng mạng Internet (10.81.100.0/24):**
 - Máy kẻ tấn công (10.81.100.10) nơi thực hiện các cuộc tấn công vào webserver và máy windows.

3.2. Mô hình ứng dụng

Application Architecture



Hình 11. Mô hình biểu diễn luồng hoạt động của quá trình giám sát an ninh mạng.

Mô hình trên mô tả luồng hoạt động của hệ thống để xử lý log và trực quan hóa lượng log đã bắt được từ các nguồn.

1. Nguồn dữ liệu gồm Window Logs từ công cụ quản lý sysmon được cài đặt trên máy Windows 10 thu thập bằng Winlogbeat để đưa về Elasticsearch. Log từ lưu lượng mạng sẽ được Suricata ghi lại nhờ chức năng giám sát vùng mạng, log được đưa về Elasticsearch bằng công cụ Filebeat.
2. Toàn bộ log từ các nguồn sẽ được đưa vào Elasticsearch để lưu trữ và phân tích. Đồng thời sử dụng công cụ Kibana để hiển thị các log một cách trực quan và ánh xạ với bảng MITRE ATT&CK.
3. Nhờ công cụ LogAlert sẽ hỗ trợ phát hiện các log có hành vi bất thường và gửi về Email của quản trị viên.

CHƯƠNG 4. TRIỂN KHAI MÔ HÌNH VÀ THỰC NGHIỆM

Ở chương này nhóm sẽ tiến hành cài đặt môi trường và các công cụ. Đồng thời thử nghiệm các cuộc tấn công vào mô hình mạng để đánh giá khả năng của các công cụ.

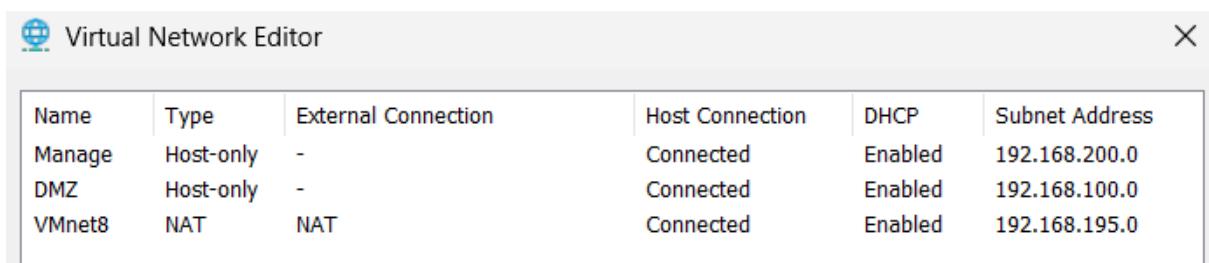
4.1. Kiến trúc hệ thống.

Để triển khai mô hình mạng doanh nghiệp cho việc kiểm thử, nhóm đã tạo 6 máy ảo với các hệ điều hành ubuntu, linux và windows trên phần mềm VMware Workstation.

Tên máy	Card mạng	Phiên bản
ELK	192.168.200.128	ubuntu-24.04.2
Suricata	192.168.100.130/192.168.200.130	ubuntu-24.04.2
Router	10.81.100.2/192.168.100.129	ubuntu-24.04.2
Windows 10	192.168.100.2	Win10-22H2-x64
Attacker	10.81.100.10	Kali-linux-2024.3
Webserver	192.168.100.4	Kali-linux-2024.3

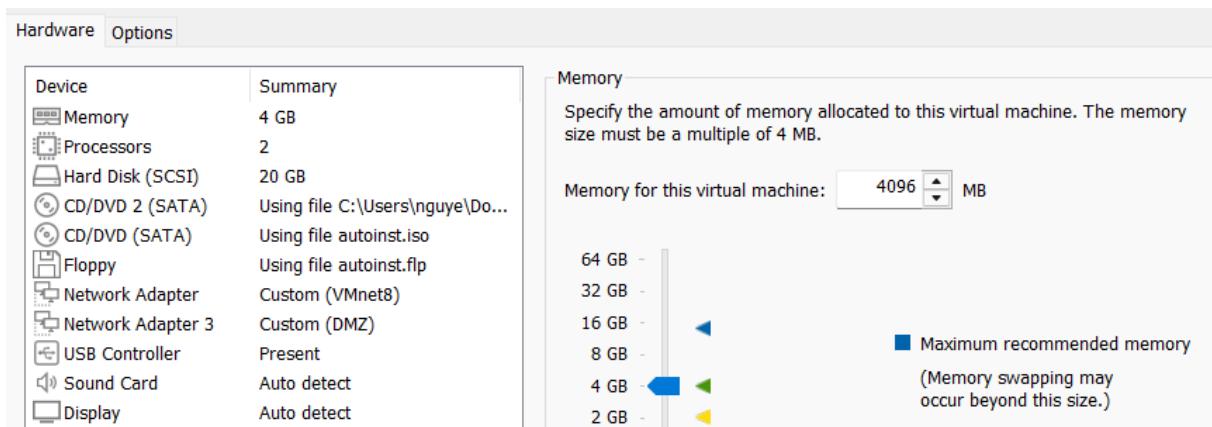
Thông tin cấu hình các máy trên VMware:

- **Virtual Network Editor:**



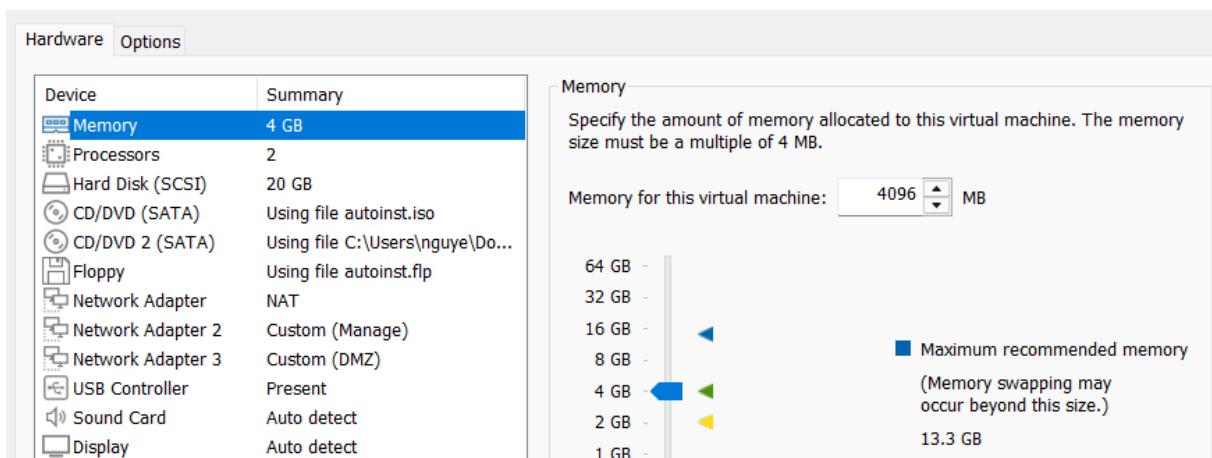
Hình 12. Cấu hình Virtual Network Editor của máy.

- **Thông tin máy router:**



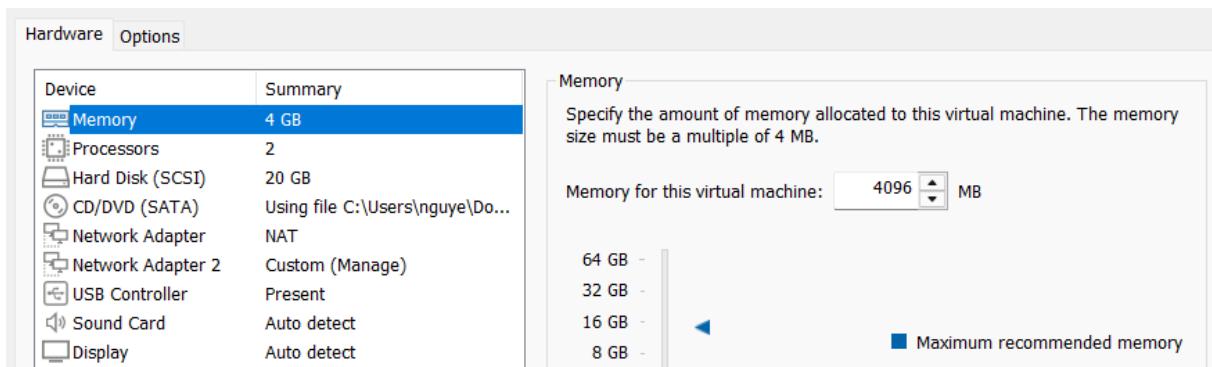
Hình 13. Cấu hình của máy router.

- **Thông tin máy Suricata:**



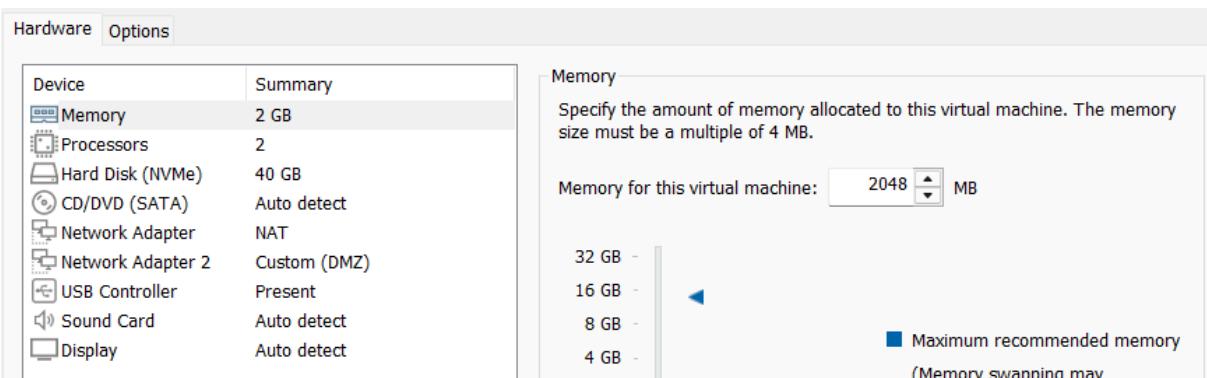
Hình 14. Cấu hình của máy Suricata.

- **Thông tin máy ELK:**



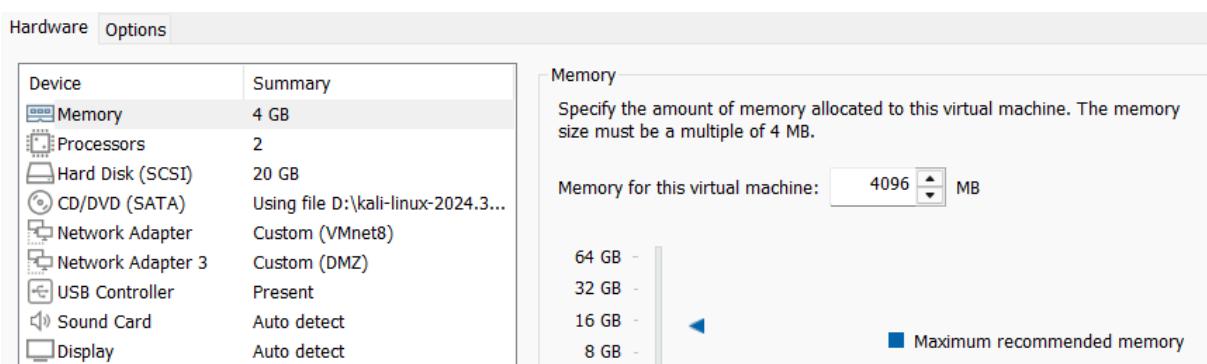
Hình 15. Cấu hình máy ELK.

- **Thông tin máy Windows 10:**



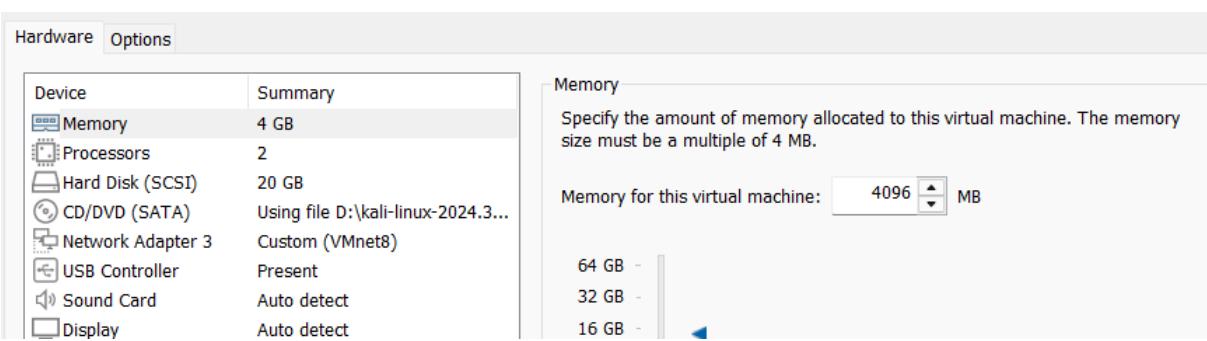
Hình 16. Cấu hình máy Windows 10.

- **Thông tin máy Webserver:**



Hình 17. Cấu hình của máy Webserver.

- **Thông tin máy Attacker:**



Hình 18. Cấu hình của máy Attacker.

4.2. Cài đặt và cấu hình công cụ.

4.2.1. Suricata.

Bước 1: Cài đặt các thư viện hỗ trợ của Suricata

Cài đặt bằng lệnh sau:

```
Sudo apt get update
```

```
Apt-get install rustc cargo make libpcre3 libpcre3-dbg libpcre3-dev build-essential autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-config -y
```

Bước 2: Cài đặt Suricata

Đầu tiên cần tải phiên bản mới nhất của Suricata trên website chính thức của nền tảng phát triển.

```
Wget https://www.openinfosecfoundation.org/download/suricata-7.0.10.tar.gz
```

Sau khi cài đặt hoàn tất, giải nén tệp đã tải xuống bằng lệnh:

```
Tar -xvzf suricata-7.0.10.tar.gz
```

Tiếp theo, cài đặt Suricata bằng lệnh:

```
Cd suricata-7.0.10
```

```
Make install-full
```

Bước 3: Cấu hình Suricata.

Tệp cấu hình Suricata mặc định được đặt tại /etc/suricata/suricata.yaml. Cần cấu hình file này để Suricata biết cần giám sát vùng mạng nào.

```
Nano /etc/suricata/suricata.yaml
```

Thay đổi các dòng sau:

```
HOME_NET: “[192.168.100.0/24]” //vùng mạng cần giám sát
```

```
EXTERNAL_NET: “!$HOME_NET”
```

Lưu và đóng tệp khi hoàn thành.

Bước 4: Kiểm tra Suricata đã cài đặt thành công hay chưa.

```
Suricata -V
```

```
This is Suricata version 7.0.10 RELEASE
```

4.2.2. ELK Stack.

Cài đặt và cấu hình Elasticsearch

Bước 1: Cài đặt khóa GPG là công cụ mã hóa và xác thực dữ liệu sử dụng để đảm bảo gói phần mềm Elasticsearch đến từ nguồn đáng tin cậy.

```
Wget -qO- https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
```

Cài đặt gói apt-transport-https có chức năng hỗ trợ tải các gói phần mềm qua giao thức HTTPS.

```
Sudo apt install apt-transport-https
```

Bước 2: Thêm repo vào source list.

```
Echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
```

Bước 3: Cài đặt elasticsearch phiên bản 8.x từ gói package đã cài.

```
Sudo apt install elasticsearch
```

Sau khi cài đặt xong thì sẽ có hiển thị một vài thông tin

- Mật khẩu elasticsearch để sử dụng
- Các lệnh thực thi để thay đổi
- Chức năng TLS đã bật

Tài khoản mặc định:

- **User:** elastic
- **Password:** tự tạo và sẽ hiển thị

```
Selecting previously unselected package elasticsearch.
(Reading database ... 74552 files and directories currently installed.)
Preparing to unpack .../elasticsearch_8.15.3_amd64.deb ...
Creating elasticsearch group... OK
Creating elasticsearch user... OK
Unpacking elasticsearch (8.15.3) ...
Setting up elasticsearch (8.15.3) ...
----- Security autoconfiguration information -----
Authentication and authorization are enabled.
TLS for the transport and HTTP layers is enabled and configured.

The generated password for the elastic built-in superuser is : a-8I9VI3IsFm827rHE8

If this node should join an existing cluster, you can reconfigure this with
'/usr/share/elasticsearch/bin/elasticsearch-reconfigure-node --enrollment-token <token-here>'
after creating an enrollment token on your existing cluster.

You can complete the following actions at any time:

Reset the password of the elastic built-in superuser with
'/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic'.

Generate an enrollment token for Kibana instances with
'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana'.

Generate an enrollment token for Elasticsearch nodes with
'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s node'.
```

Hình 19. Thông báo sau khi đã cài đặt thành công elasticsearch.

Cấu hình xác thực elasticsearch.

File cấu hình sẽ được lưu ở /etc/elasticsearch/elasticsearch.yml

Bước 1: Cấu hình network.host.

Network.host: 0.0.0.0

Bước 2: Khởi động lại dịch vụ.

Sudo systemctl restart elasticsearch

Tự động bật dịch vụ lúc khởi động.

Sudo systemctl daemon-reload

Sudo systemctl enable elasticsearch

Bước 3: Kiểm tra bằng URL.

```
Curl -cacert /etc/elasticsearch/certs/http_ca.crt -u elastic:pass
https://localhost:9200
```

```
VIIs3IsFm827rHE8 https://localhost:9200
{
  "name" : "ELASTIC",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "r82m7qI-T3qcdY4Jv5u80Q",
  "version" : {
    "number" : "8.15.3",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "f97532e680b555c3a05e73a74c28afb666923018",
    "build_date" : "2024-10-09T22:08:00.328917561Z",
    "build_snapshot" : false,
    "lucene_version" : "9.11.1",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Hình 20. Lệnh curl kiểm tra elasticsearch đã hoạt động và trả về phản hồi.

Cấu hình và cài đặt Kibana

Bước 1: Cài đặt Kibana.

```
Sudo apt install Kibana
```

Bước 2: Cấu hình Kibana.

File cấu hình Kibana được lưu ở /etc/kibana/kibana.yml.

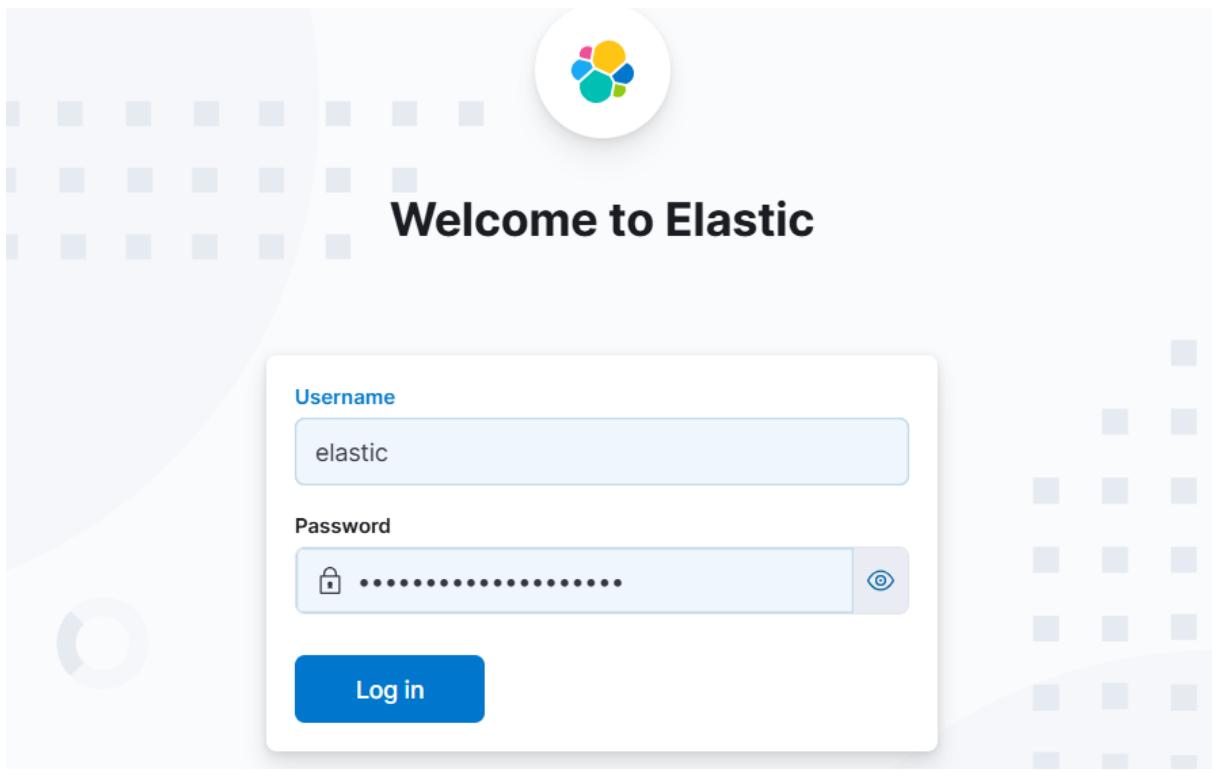
```
Server.host: 0.0.0.0
```

Bước 3: Khởi động lại dịch vụ và cấu hình tự bật lúc khởi động.

```
Sudo systemctl restart kibana
```

```
Sudo systemctl enable kibana
```

Sau khi đã cài đặt Elasticsearch và Kibana xong truy cập vào trang <https://localhost:5601> để theo dõi trang hoạt động của Elastic [9].



Hình 21. Cài đặt thành công Elasticsearch và Kibana.

4.2.3. Filebeat.

Bước 1: Cài đặt filebeat.

```
sudo apt install filebeat
```

Bước 2: Cấu hình cho filebeat tại /etc/filebeat/filebeat.yml

Comment các dòng sau:

```
#output.elasticsearch:  
# Array of hosts to connect to.  
#hosts: ["localhost:9200"]
```

Bước 3: Kích hoạt Logstash output.

```
output.logstash:  
# The Logstash hosts  
hosts: ["localhost:5044"]
```

Bước 4: Kích hoạt module Suricata trên Filebeat.

```
sudo filebeat modules enable suricata  
sudo filebeat modules list
```

Bước 5: Tạo Index Template và Kibana Dashboard.

```
sudo filebeat setup --index-management -E output.logstash.enabled=false -E  
'output.elasticsearch.hosts=["localhost:9200"]'  
sudo filebeat setup -E output.logstash.enabled=false -E  
output.elasticsearch.hosts=['localhost:9200'] -E  
setup.kibana.host=localhost:5601
```

Bước 6: Khởi chạy Filebeat.

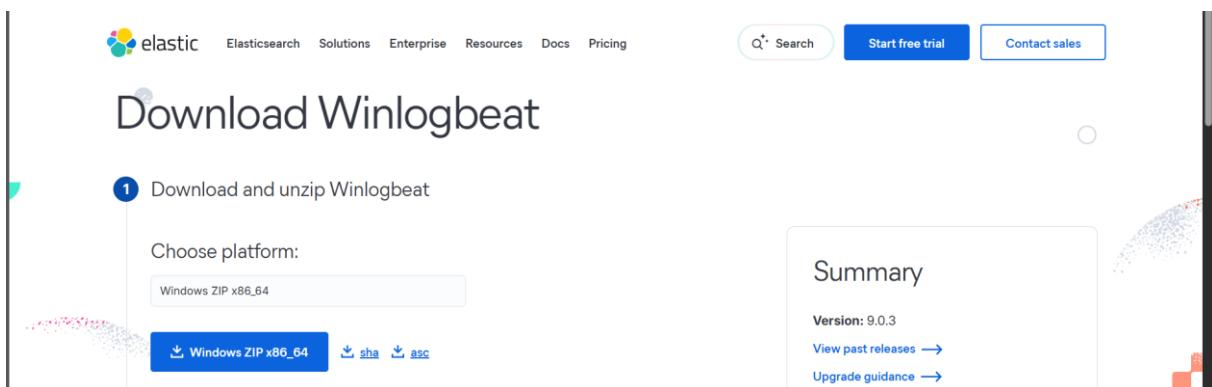
```
sudo systemctl start filebeat  
sudo systemctl enable filebeat
```

Sau khi cài đặt thành công Filebeat sẽ giúp gửi log từ Suricata về Elasticsearch [10].

4.2.4. Winlogbeat.

Để đưa được log từ dịch vụ Sysmon về Elastic cần một công cụ là Winlogbeat [11].

Bước 1: Cài đặt Winlogbeat từ trang chủ của Elastic về



Hình 22. Trang chủ cài đặt Winlogbeat.

Bước 2: Giải nén tệp vừa tải về, mở PowerShell với quyền Administrator và chạy lệnh `.\install-service-winlogbeat.ps1` để cài đặt.

```

PS C:\Users\Administrator> cd 'C:\Program Files\Winlogbeat'
PS C:\Program Files\Winlogbeat> .\install-service-winlogbeat.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful,
this script can potentially harm your computer. If you trust this script, use
the Unblock-File cmdlet to allow the script to run without this warning message.
Do you want to run C:\Program Files\Winlogbeat\install-service-winlogbeat.ps1?
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): R

Status     Name          DisplayName
-----      --           -----
Stopped    winlogbeat   winlogbeat

```

Hình 23. Cài đặt Winlogbeat.

Bước 3: Kết nối với server Elasticsearch.

Mở file winlogbeat.yml trong file đã giải nén bằng notepad hoặc trình chỉnh sửa bất kì.

```

output.elasticsearch:
  hosts: ["https://myEHost:9200"]
  username: "winlogbeat_internal"
  password: "YOUR_PASSWORD" ①
  ssl:
    enabled: true
    ca_trusted_fingerprint: "b9a10bbe64ee9826abeda6546fc988c8bf798b41957c33d05db736716513dc9c"

```

Hình 24. Chính sửa host và password của server Elastic.

Bước 4: Khởi chạy Winlogbeat bằng lệnh

```

PS > .\winlogbeat.exe setup -e
PS C:\Program Files\Winlogbeat> Start-Service winlogbeat

```

Mặc định dịch vụ Winlogbeat sẽ chạy như tài khoản Local System.

4.2.5. LogAlert gửi mail.

Để sử dụng được chức năng gửi mail cần bật chức năng sử dụng khóa mã hóa – Encryption Keys [12].

Cấu hình Encryption Keys:

Bước 1: dùng file binary trong đường dẫn /usr/share/kibana/bin

```
./kibana-encryption-keys generate
```

Copy các thiết lập đó vào file cấu hình của kibana tại /etc/kibana/kibana.yml

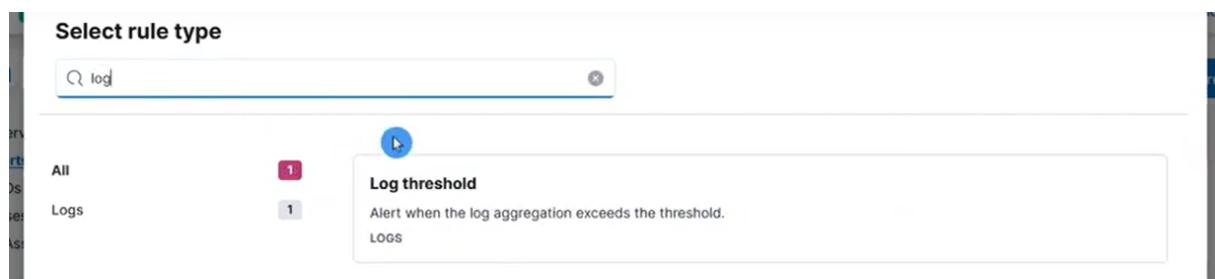
Bước 2: Khởi động lại dịch vụ Kibana

```
Systemctl restart kibana
```

Cấu hình cảnh báo LogAlert:

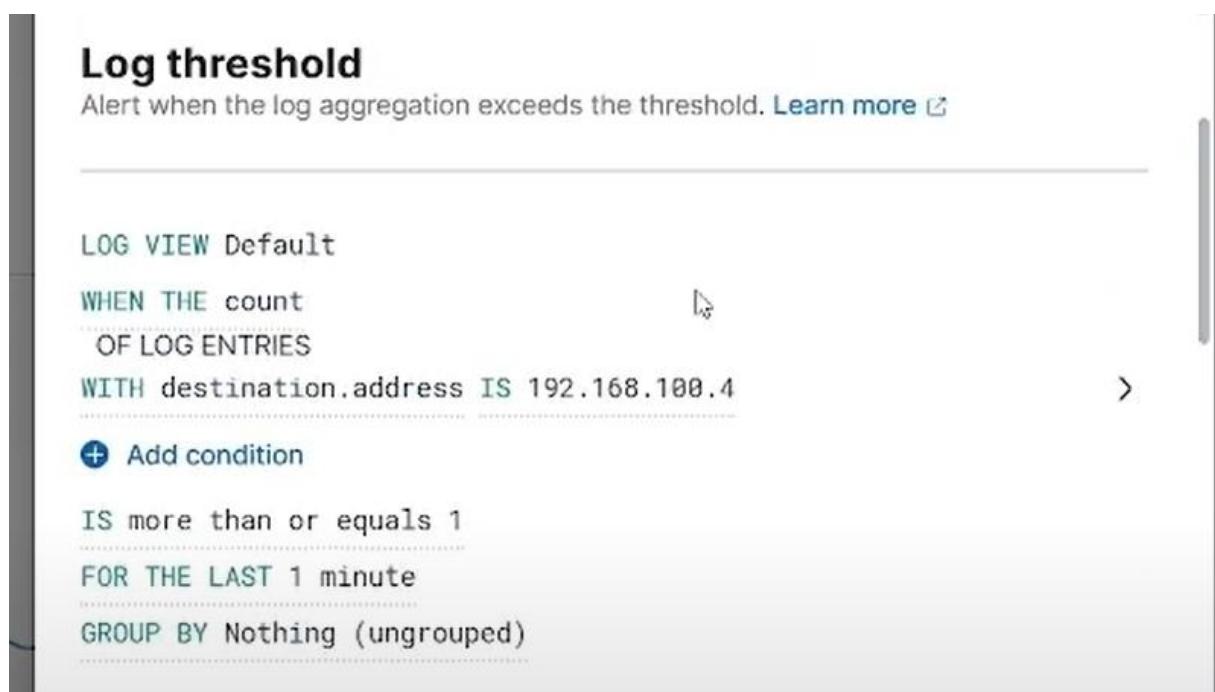
Bước 1: Tạo rule cảnh báo.

Chọn Alerts trong OBSERABILITY -> Tạo một RULE: manage rules -> Create Rule -> Chọn LOG THRESHOLD.



Hình 25. Chọn rule Logthreshold.

Bước 2: Cấu hình Rule cảnh báo:



Log threshold
Alert when the log aggregation exceeds the threshold. [Learn more](#)

LOG VIEW Default

WHEN THE count
OF LOG ENTRIES
WITH destination.address IS 192.168.100.4

+ Add condition

IS more than or equals 1
FOR THE LAST 1 minute
GROUP BY Nothing (ungrouped)

Hình 26. Cấu hình Rule cảnh báo theo threshold.

Rule này sẽ gửi cảnh báo nếu có ít nhất 1 log entry tới địa chỉ 192.168.100.4 đây là địa chỉ của nạn nhân trong 1 phút.

Cấu hình gửi qua Email

Do bản Basic sẽ không có chức năng cảnh báo qua Email hay các nền tảng khác. Biện pháp là gửi cảnh báo vào kibana.log và sẽ dùng LogAlert để giám sát từ khóa “TOP TEN OWASP ALERT” đây là tên của rule đã cài đặt ở trên.

Bước 1: Tải và cài đặt Logalert

Tạo thư mục cho logalert

```
Mkdir /opt/logalert
```

Dùng curl để tải binary và chuyển vào

```
Curl -L
```

```
https://github.com/jhuckaby/logalert/releases/latest/download/logalert-linux-x64 > /opt/logalert/logalert.bin
```

Gán quyền cho tập tin logalert và khởi chạy chương trình logalert

```
Chmod 755 /opt/logalert/logalert.bin
```

```
/opt/logalert/logalert.bin
```

Bước 2: Tạo mật khẩu ứng dụng

Truy cập vào trang quản lý Google, tìm kiếm mật khẩu ứng dụng, tạo và gán một tên cho ứng dụng từ đây mật khẩu sẽ được tạo.

← App passwords

App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

Your app passwords

logalert

Created on Jun 22, last used on 10:19 AM



Hình 27. App password trên gmail của quản trị viên.

Cấu hình LOGALERT

File cấu hình của logalert được đặt tại /opt/logalert/config.json

```
[{"monitors": [
    {
        "name": "TOP TEN OSWAP ALERT",
        "path": "/var/log/kibana/kibana.log",
        "match": "TOP TEN OSWAP ALERT",
        "email": "22520442@gm.uit.edu.vn"
    }
],
"mail_settings": {
    "host": "smtp.gmail.com",
    "port": 465,
    "secure": true,
    "auth": {
        "user": "22520442@gm.uit.edu.vn",
        "pass": "ssxxclrktypobean"
    },
    "from": "22520442@gm.uit.edu.vn"
},
"sleep": 5,
"echo": 1,
"verbose": 2
}]
```

Hình 28. Cấu hình file config logalert.

File config.json để thiết lập một monitor – công cụ giám sát để gửi cảnh báo qua email khi phát hiện điều kiện cụ thể.

- “monitor”: danh sách các monitor được định nghĩa
 - “name”: tên monitor là “TOP TEN OWASP ALERT”.
 - “path”: Đường dẫn đến log được giám sát là “/var/log/kibana/kibana.log”.
 - “match”: Mẫu để khớp với nội dung log là “TOP TEN OWASP ALERT”.
 - “email”: địa chỉ của quản trị viên nhận cảnh báo
- “mail_settings”: Cài đặt mail server
 - “host”: máy chủ SMTP là “smtp.gmail.com”.
 - “port”: Cổng là 465.
 - “secure”: sử dụng kết nối bảo mật.
 - “auth”: Thông tin xác thực:
 - “user”: Email người dùng.
 - “pass”: mật khẩu ứng dụng đã thiết lập ở trên.

Cấu hình LOGALERT là SERVICE:

Bước 1: Tạo một service file

```
Nano logalert.service
```

Bước 2: Thực hiện cấu hình

```
[Unit]
Description=logalert
[Service]
WorkingDirectory=/opt/logalert/
ExecStart=/opt/logalert/logalert.bin
[Install]
WantedBy=multi-user.target
```

Bước 3: Khởi chạy dịch vụ

Systemctl start logalert

```
root@elk-Virtual-Platform:/opt/logalert# systemctl status syslog
● rsyslog.service - System Logging Service
  Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; preset: enabled)
  Active: active (running) since Sun 2025-07-06 08:03:10 +07; 1h 41min ago
TriggeredBy: ● syslog.socket
    Docs: man:rsyslogd(8)
           man:rsyslog.conf(5)
           https://www.rsyslog.com/doc/
 Main PID: 762 (rsyslogd)
   Tasks: 4 (limit: 4546)
  Memory: 1.5M (peak: 5.1M swap: 1.2M swap peak: 1.3M)
    CPU: 506ms
   CGroup: /system.slice/rsyslog.service
           └─762 /usr/sbin/rsyslogd -n -iNONE

Jul 06 08:03:08 elk-VMware-Virtual-Platform systemd[1]: Starting rsyslog.service - System Logging Service...
Jul 06 08:03:10 elk-VMware-Virtual-Platform rsyslogd[762]: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog' (fd 3) from systemd. [v8.231]
Jul 06 08:03:10 elk-VMware-Virtual-Platform rsyslogd[762]: rsyslogd's groupid changed to 102
Jul 06 08:03:10 elk-VMware-Virtual-Platform systemd[1]: Started rsyslog.service - System Logging Service.
Jul 06 08:03:10 elk-VMware-Virtual-Platform rsyslogd[762]: rsyslogd's userid changed to 102
Jul 06 08:03:10 elk-VMware-Virtual-Platform rsyslogd[762]: [origin software="rsyslogd" swVersion="8.2312.0" x-pid="762" x-info="https://www.rsyslog.c
Jul 06 08:03:16 elk-VMware-Virtual-Platform systemd[1]: rsyslog.service: Sent signal SIGHUP to main process 762 (rsyslogd) on client request.
Jul 06 08:03:16 elk-VMware-Virtual-Platform rsyslogd[762]: [origin software="rsyslogd" swVersion="8.2312.0" x-pid="762" x-info="https://www.rsyslog.c
[lines 1-22/22 (END)]
```

Hình 29. Dịch vụ Logalert đã khởi chạy thành công.

4.2.6. Router.

NAT outbound cho phép các máy trong mạng có thể đi ra Internet. Sau khi cấu hình NAT thành công. Máy Kali có thể kết nối Internet.

Bước 1: Xác minh các card mạng, xem đã được cài đặt đúng chưa.

```
Ls /etc/sysconfig/network-scripts/ifcfg-eth* | wc -l
```

Bước 2: Cấu hình NAT với bảng IP.

- Xóa bộ nhớ đệm và thiết lập default.

```
Iptables --flush
```

```
Iptable -table nat --flush
```

```
Iptables -delete-chain
```

- Xóa tất cả các chuỗi không có trong “filter” mặc định và bảng NAT.

```
Iptables -table nat -delete-chain
```

- Thiết lập IP Forwarding và Masquerading.

```
Iptables -table nat -append POSTROUTING -out-interface ens34 -j  
MASQUERADE
```

```
Iptables -append FORWARD -in-interface ens35 -j ACCEPT
```

- Cho phép chuyển tiếp gói qua Kernel.

```
Echo 1 > /proc/sys/net/ipv4/ip_forward
```

Bước 3: Kiểm tra bằng cách ping tới google.com

```
elk@elk-VMware-Virtual-Platform:~$ ping google.com
PING google.com (142.251.10.139) 56(84) bytes of data.
64 bytes from sd-in-f139.1e100.net (142.251.10.139): icmp_seq=1 ttl=128 time=62.3 ms
64 bytes from sd-in-f139.1e100.net (142.251.10.139): icmp_seq=2 ttl=128 time=29.9 ms
64 bytes from sd-in-f139.1e100.net (142.251.10.139): icmp_seq=3 ttl=128 time=32.2 ms
64 bytes from sd-in-f139.1e100.net (142.251.10.139): icmp_seq=4 ttl=128 time=30.5 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 29.937/38.742/62.340/13.650 ms
```

Hình 30. Ping tới google.com trên máy trong mạng nội bộ.

4.2.7. AI-based IDS.

4.2.7.1. Huấn luyện và đánh giá tính khả thi của mô hình:

- Nhóm sẽ sử dụng thuật toán random forest và tập dữ liệu CIC-IDS-2017 để huấn luyện mô hình trên Google Colab và đánh giá khả năng phát hiện tấn công.
- Import các thư viện cần thiết:

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
sns.set(style='darkgrid')
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import joblib
from imblearn.over_sampling import SMOTE
from sklearn.decomposition import IncrementalPCA

```

Hình 31. Các thư viện python cần thiết.

- Gộp các file .csv trong bộ dữ liệu CIC-IDS-2017 lại thành một file duy nhất

```

data_files = [
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Monday-WorkingHours.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Tuesday-WorkingHours.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Wednesday-workingHours.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Friday-WorkingHours-Morning.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv',
    '/content/drive/MyDrive/Colab Notebooks/IDS_IDS/input/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv'
]
data = pd.concat([pd.read_csv(f) for f in data_files], ignore_index=True)

```

Hình 32. Gộp dữ liệu trong dataset CIC-IDS-2017.

- Xử lý giá trị vô cực và không xác định bằng cách thay thế chúng bằng giá trị thiếu và in ra số lượng giá trị bị thiếu

```

# Xử lý giá trị vô cực và NaN
print(f'Initial missing values: {data.isna().sum().sum()}')
data.replace([np.inf, -np.inf], np.nan, inplace=True)
print(f'Missing values after processing infinite values: {data.isna().sum().sum()}')

Initial missing values: 353
Missing values after processing infinite values: 3128

```

Hình 33. In ra số lượng các giá trị bị thiếu.

- Xử lý giá trị bị thiếu

```

med_flow_bytes = data['Flow Bytes/s'].median()
med_flow_packets = data['Flow Packets/s'].median()

print('Median of Flow Bytes/s: ', med_flow_bytes)
print('Median of Flow Packets/s: ', med_flow_packets)

```

Median of Flow Bytes/s: 3715.0378579999997
Median of Flow Packets/s: 69.742244285

```

# Filling missing values with median
data['Flow Bytes/s'].fillna(med_flow_bytes, inplace = True)
data['Flow Packets/s'].fillna(med_flow_packets, inplace = True)

```

Hình 34. Thêm giá trị trung bình vào chỗ bị thiếu.

```

print('Number of \'Flow Bytes/s\' missing values:', data['Flow Bytes/s'].isna().sum())
print('Number of \'Flow Packets/s\' missing values:', data['Flow Packets/s'].isna().sum())

Number of 'Flow Bytes/s' missing values: 0
Number of 'Flow Packets/s' missing values: 0

```

Hình 35. Kiểm tra lại sau khi xử lý các giá trị bị thiếu.

- Gom các loại tấn công cụ thể thành nhóm chung ví dụ như nhóm SQL injection, XSS ... thành Web Attack

	count
Attack Type	
BENIGN	2096484
DoS	193748
DDoS	128016
Port Scan	90819
Brute Force	9152
Web Attack	2143
Bot	1953
Infiltration	36
Heartbleed	11

dtype: int64

Hình 36. In ra số lượng các loại tấn công.

- Loại bỏ cột ‘Label’ để tránh dữ thừa thông tin trong Dataframe

```

data.drop('Label', axis = 1, inplace = True)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Attack Number'] = le.fit_transform(data['Attack Type'])

print(data['Attack Number'].unique())

[0 2 4 5 8 6 1 7 3]

```

Hình 37. Chuyển đổi các nhãn tấn công dạng chữ thành dạng số.

```

# Printing corresponding attack type for each encoded value
encoded_values = data['Attack Number'].unique()
for val in sorted(encoded_values):
    print(f"{val}: {le.inverse_transform([val])[0]}")

0: BENIGN
1: Bot
2: Brute Force
3: DDoS
4: DoS
5: Heartbleed
6: Infiltration
7: Port Scan
8: Web Attack

```

Hình 38. In ra danh sách các loại tấn công tương ứng với mỗi giá trị số đã được mã hóa.

- Xác định và loại bỏ các cột không mang lại thông tin hữu ích cho việc huấn luyện mô hình, giúp giảm kích thước dữ liệu và cải thiện hiệu suất của mô hình

```

# Dropping columns with only one unique value
num_unique = data.nunique()
one_variable = num_unique[num_unique == 1]
not_one_variable = num_unique[num_unique > 1].index

dropped_cols = one_variable.index
data = data[not_one_variable]

print('Dropped columns:')
dropped_cols

Dropped columns:
Index(['Bwd PSH Flags', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk',
       'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk',
       'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate'],
      dtype='object')

```

Hình 39. Loại bỏ các cột chỉ có một giá trị duy nhất.

```

# Columns after removing non variant columns
data.columns

Index(['Destination Port', 'Flow Duration', 'Total Fwd Packets',
       'Total Backward Packets', 'Total Length of Fwd Packets',
       'Total Length of Bwd Packets', 'Fwd Packet Length Max',
       'Fwd Packet Length Min', 'Fwd Packet Length Mean',
       'Fwd Packet Length Std', 'Bwd Packet Length Max',
       'Bwd Packet Length Min', 'Bwd Packet Length Mean',
       'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s',
       'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min',
       'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max',
       'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std',
       'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Fwd URG Flags',
       'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s',
       'Bwd Packets/s', 'Min Packet Length', 'Max Packet Length',
       'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance',
       'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count',
       'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count',
       'Down/Up Ratio', 'Average Packet Size', 'Avg Fwd Segment Size',
       'Avg Bwd Segment Size', 'Fwd Header Length.1', 'Subflow Fwd Packets',
       'Subflow Fwd Bytes', 'Subflow Bwd Packets', 'Subflow Bwd Bytes',
       'Init_Win_bytes_forward', 'Init_Win_bytes_backward', 'act_data_pkt_fwd',
       'min_seg_size_forward', 'Active Mean', 'Active Std', 'Active Max',
       'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min',
       'Attack Type', 'Attack Number'],
      dtype='object')

```

Hình 40. Sau khi loại bỏ các cột không cần thiết.

- Đưa các features về cùng một phạm vi giá trị để cải thiện hiệu suất mô hình

```

# Standardizing the dataset
from sklearn.preprocessing import StandardScaler

features = data.drop('Attack Type', axis = 1)
attacks = data['Attack Type']

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

```

Hình 41. Chuẩn hóa dữ liệu.

- Giảm chiều dữ liệu xuống còn một nửa số lượng features ban đầu, giúp giảm thiểu độ phức tạp của mô hình, tăng tốc độ huấn luyện và cải thiện hiệu suất của mô hình. Sau đó in ra phần trăm thông tin được giữ lại sau khi giảm chiều, giúp đánh giá mức độ mát mẻ thông tin.

```

from sklearn.decomposition import IncrementalPCA

size = len(features.columns) // 2
ipca = IncrementalPCA(n_components = size, batch_size = 500)
for batch in np.array_split(scaled_features, len(features) // 500):
    ipca.partial_fit(batch)

print(f'information retained: {sum(ipca.explained_variance_ratio_):.2%}')

```

information retained: 99.08%

```

transformed_features = ipca.transform(scaled_features)
new_data = pd.DataFrame(transformed_features, columns = [f'PC{i+1}' for i in range(size)])
new_data['Attack Type'] = attacks.values

```

Hình 42. Giảm chiều dữ liệu.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC27	PC28	PC29	PC30	PC31	PC32	PC33
0	-2.390979	-0.054142	0.569875	0.608032	3.748798	0.193173	-0.016745	-0.214749	-0.249818	1.072820	...	0.707226	0.167883	-0.518800	1.451156	-0.156591	0.262873	0.001385
1	-2.913872	-0.069280	0.888999	1.462739	8.890884	0.517774	-0.061181	1.115094	1.970735	-2.750156	...	0.214982	-0.017288	-2.543196	2.112990	-0.649356	0.563292	0.027271
2	-2.449933	-0.055862	0.606199	0.704919	4.325623	0.227464	-0.021853	-0.068832	-0.002663	0.652011	...	0.656176	0.150637	-0.732336	1.525840	-0.208474	0.294626	0.004077
3	-2.914726	-0.069319	0.889864	1.464659	8.896365	0.515998	-0.061352	1.113251	1.971161	-2.746190	...	0.218626	-0.013960	-2.531800	2.114137	-0.646487	0.561665	0.027086
4	-1.538079	0.080213	-0.489878	0.320055	-0.525546	0.755004	0.100991	0.729182	-1.141042	-0.572120	...	-0.239837	-0.782527	0.348497	0.860799	-0.178431	-0.246678	-0.026624
...	
2522357	-2.304647	-0.047668	0.476010	0.385633	2.175687	-0.164691	-0.015810	-0.779535	-0.897969	2.692285	...	0.614358	0.472443	0.588583	0.787838	0.261563	-0.053321	-0.018226
2522358	-2.301679	-0.047573	0.474787	0.383356	2.155614	-0.166606	-0.015741	-0.782345	-0.900963	2.698772	...	0.617024	0.477510	0.587569	0.788610	0.262761	-0.053720	-0.018305
2522359	-2.301010	-0.047552	0.474510	0.382820	2.151095	-0.167023	-0.015725	-0.782862	-0.901615	2.700166	...	0.617562	0.478571	0.587203	0.788784	0.262980	-0.053784	-0.018320
2522360	-2.135438	-0.041386	0.357036	0.207592	1.803137	0.264473	0.006410	-0.297039	-0.613434	0.952894	...	0.657317	0.305405	1.751218	-0.857851	0.304591	-0.201020	-0.011825
2522361	-2.418212	-0.017525	0.400231	0.774027	2.496048	0.592719	0.070086	-0.411971	-2.333711	3.877034	...	0.560174	0.655739	-0.291241	-0.090932	0.227707	0.306353	0.020530

2522362 rows × 36 columns

Hình 43. Dữ liệu mới sau khi giảm chiều.

- Chọn lọc và cân bằng dữ liệu cho việc huấn luyện mô hình học máy, tập trung vào cột 'Attack Type' để tạo ra một tập dữ liệu cân bằng hơn, các loại tấn công được đại diện một cách đồng đều.

```

class_counts = new_data['Attack Type'].value_counts()
selected_classes = class_counts[class_counts > 1950]
class_names = selected_classes.index
selected = new_data[new_data['Attack Type'].isin(class_names)]

dfs = []
for name in class_names:
    df = selected[selected['Attack Type'] == name]
    if len(df) > 2500:
        df = df.sample(n = 5000, random_state = 0)

    dfs.append(df)

df = pd.concat(dfs, ignore_index = True)
df['Attack Type'].value_counts()

```

Attack Type	count
BENIGN	5000
DoS	5000
DDoS	5000
Port Scan	5000
Brute Force	5000
Web Attack	2143
Bot	1953

dtype: int64

Hình 44. Chọn lọc dữ liệu.

- Sử dụng ‘SMOTE’ để tạo ra các mẫu dữ liệu tổng hợp cho các lớp thiểu số, cân bằng bộ dữ liệu và giúp mô hình học máy hoạt động hiệu quả hơn.

```

from imblearn.over_sampling import SMOTE

X = df.drop('Attack Type', axis=1)
y = df['Attack Type']

smote = SMOTE(sampling_strategy='auto', random_state=0)
X_upsampled, y_upsampled = smote.fit_resample(X, y)

blnc_data = pd.DataFrame(X_upsampled)
blnc_data['Attack Type'] = y_upsampled
blnc_data = blnc_data.sample(frac=1)

blnc_data['Attack Type'].value_counts()

```

Attack Type	count
Web Attack	5000
Bot	5000
Port Scan	5000
DoS	5000
DDoS	5000
Brute Force	5000
BENIGN	5000

dtype: int64

Hình 45. Cân bằng dữ liệu

- Chuẩn bị dữ liệu để huấn luyện mô hình bằng cách Chia dữ liệu để huấn luyện, trong đó 25% dữ liệu sẽ được phân bổ cho tập kiểm tra, 75% còn lại được sử dụng để huấn luyện

```
features = blnc_data.drop('Attack Type', axis = 1)
labels = blnc_data['Attack Type']

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size = 0.25, random_state = 0)
```

Hình 46. Chia dữ liệu để huấn luyện.

- Khởi tạo mô hình Random Forest 1 với độ sâu tối đa của cây là 4.
- Huấn luyện mô hình sử dụng dữ liệu huấn luyện X_train, y_train.
- Sử dụng mô hình đã huấn luyện để dự đoán nhãn cho dữ liệu kiểm tra (X_test)
- Đánh giá:
 - Thực hiện kiểm định chéo (cross-validation) 5 lần trên tập huấn luyện để đánh giá độ ổn định của mô hình.
 - In ra điểm cross-validation trung bình.
 - Tính toán và in ra độ chính xác (accuracy) trên tập kiểm tra.
 - In ra báo cáo phân loại chi tiết (precision, recall, f1-score) và ma trận nhầm lẫn (confusion matrix) để đánh giá hiệu suất trên từng lớp.

```
from sklearn.ensemble import RandomForestClassifier

rf1 = RandomForestClassifier(max_depth = 4, random_state = 0)
rf1.fit(X_train, y_train)
y_pred_rf = rf1.predict(X_test)

cv_rf1 = cross_val_score(rf1, X_train, y_train, cv = 5)
print('Random Forest Model 1')
print(f'\nCross-validation scores: ', ', '.join(map(str, cv_rf1)))
print(f'\nMean cross-validation score: {cv_rf1.mean():.2f}')
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
```

Hình 47. Khởi tạo và huấn luyện mô hình Random Forest 1.

```

Random Forest Model 1

Cross-validation scores: 0.9485714285714286, 0.9424761904761905, 0.9476190476190476, 0.9415238095238095, 0.9487619047619048

Mean cross-validation score: 0.95
Accuracy: 0.9419428571428572
      precision    recall   f1-score  support
BENIGN       0.98     0.81     0.89     1228
Bot          0.86     0.99     0.92     1252
Brute Force   0.97     0.96     0.97     1227
DDoS          0.86     0.98     0.91     1250
DoS           0.95     0.84     0.90     1259
Port Scan     1.00     1.00     1.00     1292
Web Attack    1.00     1.00     1.00     1242

accuracy          0.94     8750
macro avg        0.95     0.94     0.94     8750
weighted avg     0.95     0.94     0.94     8750

```

Hình 48. Kết quả huấn luyện mô hình 1.

- Tiếp tục khởi tạo mô hình Random Forest 2 giữ nguyên các thông số của mô hình 1 và chỉ thay đổi chiều sâu của cây thành 6

```

rf2 = RandomForestClassifier(max_depth = 6, random_state = 0)
rf2.fit(X_train, y_train)
y_pred_rf = rf2.predict(X_test)

cv_rf2 = cross_val_score(rf2, X_train, y_train, cv = 5)
print('Random Forest Model 2')
print(f'\nCross-validation scores: ', ', '.join(map(str, cv_rf2)))
print(f'\nMean cross-validation score: {cv_rf2.mean():.2f}')
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))

```

Hình 49. Khởi tạo và huấn luyện mô hình Random Forest 2.

```

Random Forest Model 2

Cross-validation scores: 0.9857142857142858, 0.9872380952380952, 0.9832380952380952, 0.9853333333333333, 0.9887619047619047

Mean cross-validation score: 0.99
Accuracy: 0.9838857142857143
      precision    recall   f1-score  support
BENIGN       0.98     0.96     0.97     1228
Bot          0.94     1.00     0.97     1252
Brute Force   1.00     0.96     0.98     1227
DDoS          0.98     0.99     0.99     1250
DoS           0.98     0.98     0.98     1259
Port Scan     1.00     1.00     1.00     1292
Web Attack    1.00     1.00     1.00     1242

accuracy          0.98     8750
macro avg        0.98     0.98     0.98     8750
weighted avg     0.98     0.98     0.98     8750

```

Hình 50. Kết quả huấn luyện mô hình 2.

4.2.7.2. Đánh giá mô hình

1. Các tiêu chí đánh giá

Để đánh giá hiệu quả của mô hình AI-Based IDS trong việc phát hiện các cuộc tấn công mạng, nhóm nghiên cứu sử dụng các tiêu chí phổ biến trong lĩnh vực phân loại bao gồm Precision, Recall, F1-score và Accuracy, được tính toán dựa trên bốn tham số TP, TN, FP và FN :

- **Accuracy:** Tỷ lệ các mẫu được phân loại đúng trên tổng số mẫu. Đây là chỉ số đánh giá tổng thể độ chính xác của mô hình. Được tính bằng công thức:

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

Trong đó:

TP (True Positive): dự đoán đúng mẫu là tấn công.

TN (True Negative): dự đoán đúng mẫu là bình thường.

FP (False Positive): dự đoán nhầm mẫu bình thường là tấn công.

FN (False Negative): dự đoán nhầm mẫu tấn công là bình thường.

- **Precision:** Tỷ lệ các mẫu được mô hình dự đoán là tấn công mà thật sự là tấn công. Được tính bằng công thức:

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

- **Recall (độ bao phủ) :** Tỷ lệ các mẫu thật sự là tấn công mà mô hình dự đoán đúng. Được tính bằng công thức:

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

- **F1-score:** là chỉ số cân bằng giữa Precision và Recall. Được tính bằng công thức:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Ngoài ra nhóm còn sử dụng ma trận nhầm lẫn (Confusion Matrix) và đường cong đặc trưng hoạt động của bộ thu (ROC) để so sánh độ hiệu quả của hai mô hình được thử nghiệm trên cùng một tập dữ liệu chung.

Ma trận nhầm lẫn là một công cụ đo lường hiệu suất được sử dụng trong học có giám sát, đặc biệt là cho các vấn đề phân loại. Nó cung cấp một bản tóm tắt toàn diện về mức độ hiệu quả của một mô hình phân loại bằng cách so sánh các phân loại dự đoán với các phân loại thực tế cho một tập dữ liệu thử nghiệm. Hình ảnh trực quan này giúp hiểu rõ không chỉ tính chính xác tổng thể của mô hình mà còn cả các loại lỗi mà nó mắc phải (tức là mô hình "bị nhầm lẫn"). Nó đặc biệt hữu ích trong Học máy (ML) và Trí tuệ nhân tạo (AI) để đánh giá các mô hình được đào tạo cho các tác vụ như phân loại hình ảnh hoặc phát hiện đối tượng.

Ma trận nhầm lẫn thường được trình bày dưới dạng lưới vuông, trong đó mỗi hàng biểu diễn các trường hợp trong một lớp thực tế và mỗi cột biểu diễn các trường hợp trong một lớp dự đoán (hoặc ngược lại). Đối với một bài toán phân loại nhị phân đơn giản (hai lớp, ví dụ: Tích cực và Tiêu cực), ma trận có bốn ô:

True Positives (TP): Mô hình đã dự đoán đúng lớp dương.

True Negatives (TN): Mô hình đã dự đoán đúng lớp âm tính.

Kết quả dương tính giả (FP) (Lỗi loại I): Mô hình dự đoán không chính xác lớp dương tính (mô hình dự đoán lớp dương tính, nhưng lớp thực tế lại là lớp âm tính).

Âm tính giả (FN) (Lỗi loại II): Mô hình dự đoán không chính xác lớp âm tính (mô hình dự đoán lớp âm tính, nhưng lớp thực tế lại là dương tính).

Bốn thành phần này tạo thành cơ sở để tính toán nhiều số liệu hiệu suất khác nhau.

Đường cong ROC là hình ảnh trực quan về hiệu suất của mô hình trên tất cả các ngưỡng. Phiên bản dài của tên, đặc tính hoạt động của bộ thu, là một phần còn lại của công nghệ phát hiện radar trong Thế chiến II.

Đường cong ROC được vẽ bằng cách tính tỷ lệ dương tính thực (TPR) và tỷ lệ dương tính giả (FPR) ở mọi ngưỡng có thể (trong thực tế, ở các khoảng thời gian đã chọn), sau đó lập biểu đồ TPR trên FPR.

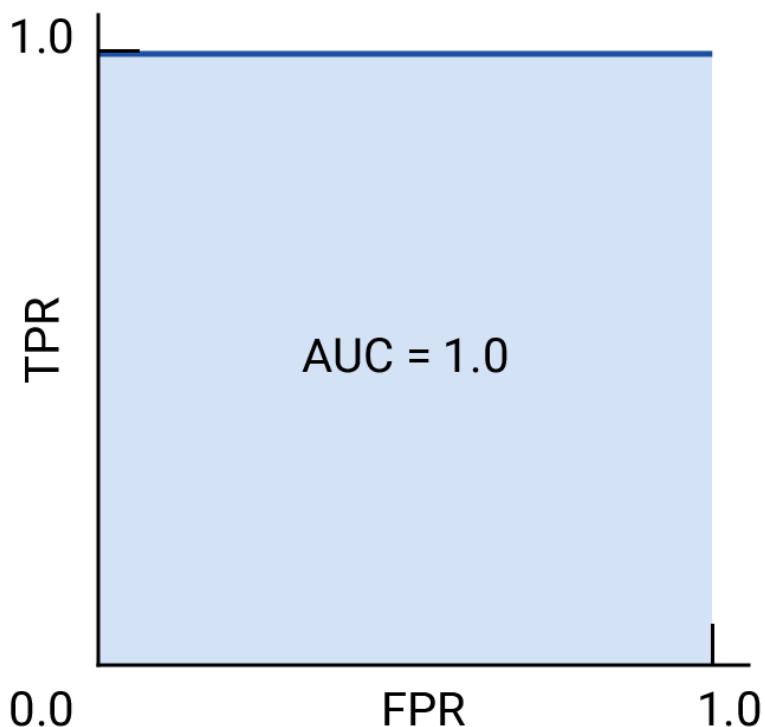
- **TPR (true positive rate):** Chỉ số này chính bằng độ phủ. Một số tài liệu thống kê còn gọi chúng là độ nhạy (sensitivity). Đây là tỷ lệ các trường hợp phân loại đúng dương tính trên tổng số các trường hợp thực tế là dương tính. Nó có tác dụng đánh giá mức độ dự báo chính xác của mô hình trên nhóm dương tính. Khi giá trị của nó càng cao, mô hình dự báo càng tốt trên nhóm dương tính. Nếu $TPR = 0.9$, chúng ta tin rằng 90% các mẫu thuộc nhóm dương tính đã được mô hình phân loại đúng.

$$TPR/\text{recall}/\text{sensitivity} = \frac{TP}{\text{total positive}}$$

- **FPR (false positive rate):** Tỷ lệ dự báo sai các trường hợp thực tế là âm tính thành thành dương tính trên tổng số các trường hợp thực tế là âm tính. Nếu giá trị của $FPR = 0.1$, mô hình đã dự báo sai 10% trên tổng số các trường hợp là âm tính. Một mô hình có FPR càng thấp thì mô hình càng chuẩn xác vì sai số của nó trên nhóm âm tính càng thấp. Phần bù của FPR là độ đặc hiệu (specificity) đo lường tỷ lệ dự báo đúng các trường hợp âm tính trên tổng số các trường hợp thực tế là âm tính.

$$FPR = 1 - \text{specificity} = \frac{FP}{\text{total negative}}$$

Một mô hình hoàn hảo, ở một ngưỡng nào đó có TPR là 1.0 và FPR là 0.0, có thể được biểu thị bằng một điểm tại $(0, 1)$ nếu tất cả các ngưỡng khác bị bỏ qua hoặc bằng cách sau:



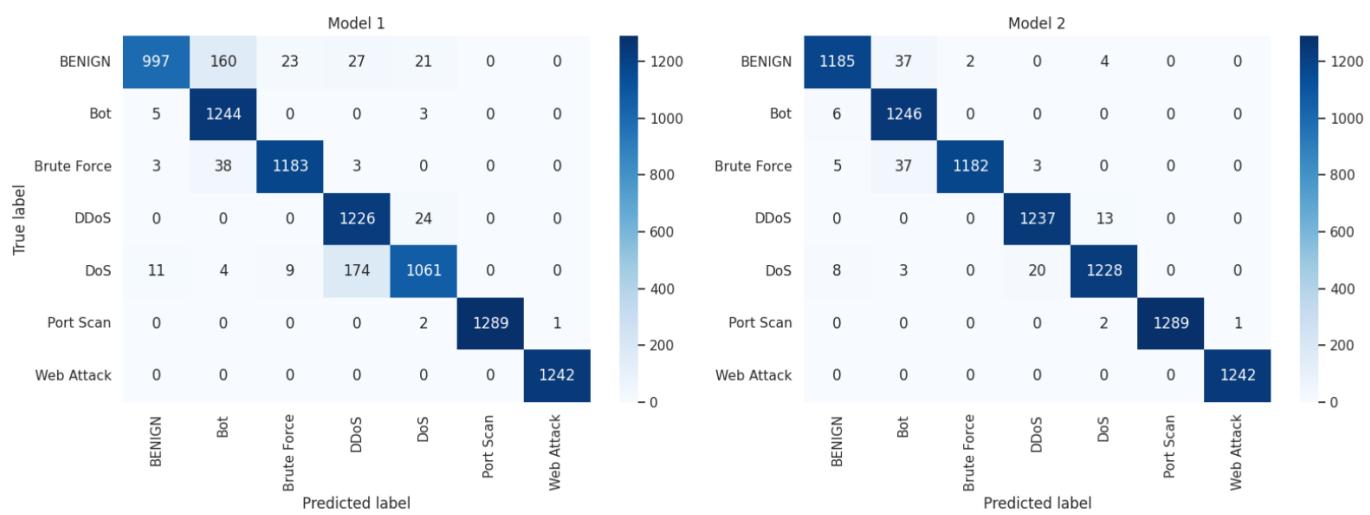
Hình 51. ROC và AUC của một mô hình hoàn hảo giả định.

Diện tích dưới đường cong ROC (AUC) đại diện cho xác suất mô hình, nếu được cung cấp một ví dụ dương tính và âm tính được chọn ngẫu nhiên, sẽ xếp hạng dương tính cao hơn âm tính.

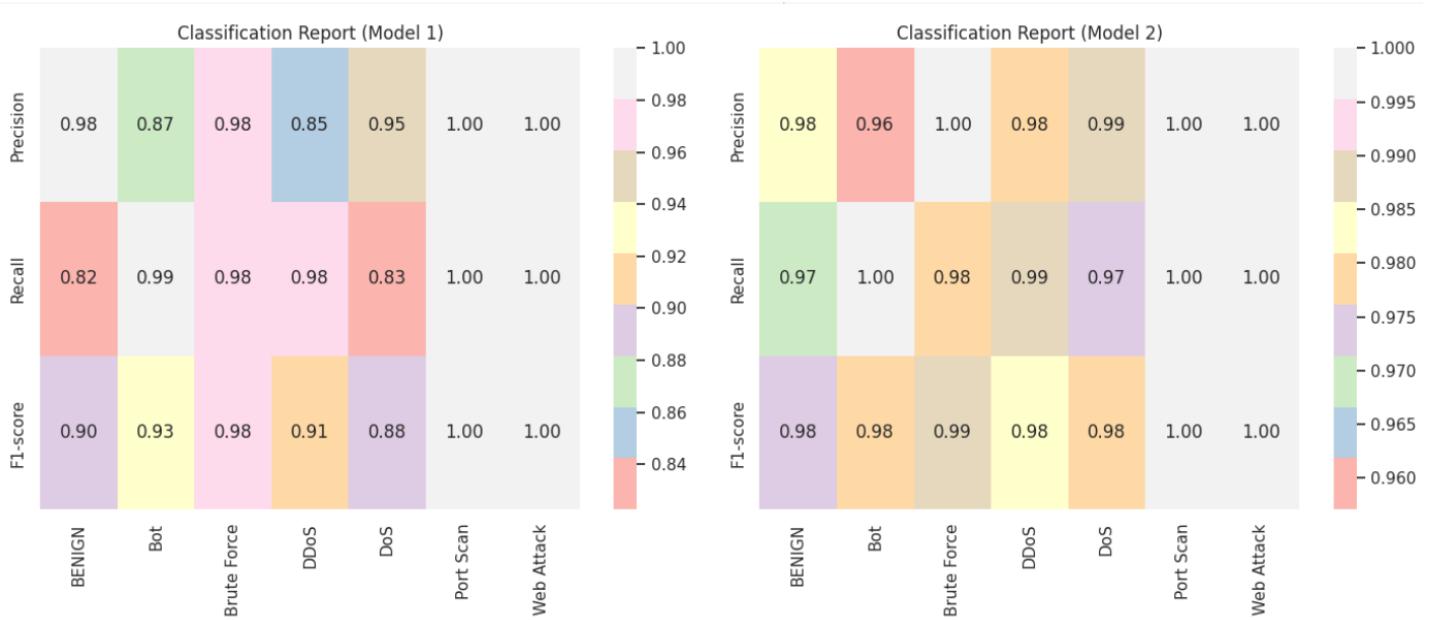
Mô hình hoàn hảo ở trên, chứa một hình vuông có các cạnh dài 1, có diện tích dưới đường cong (AUC) là 1.0. Điều này có nghĩa là có 100% khả năng mô hình sẽ xếp hạng chính xác một ví dụ dương tính được chọn ngẫu nhiên cao hơn một ví dụ âm tính được chọn ngẫu nhiên. Nói cách khác, khi xem xét mức độ phân tán của các điểm dữ liệu bên dưới, AUC cho biết xác suất mô hình sẽ đặt một hình vuông được chọn ngẫu nhiên ở bên phải một hình tròn được chọn ngẫu nhiên, không phụ thuộc vào vị trí đặt ngưỡng.

2. Kết quả đánh giá

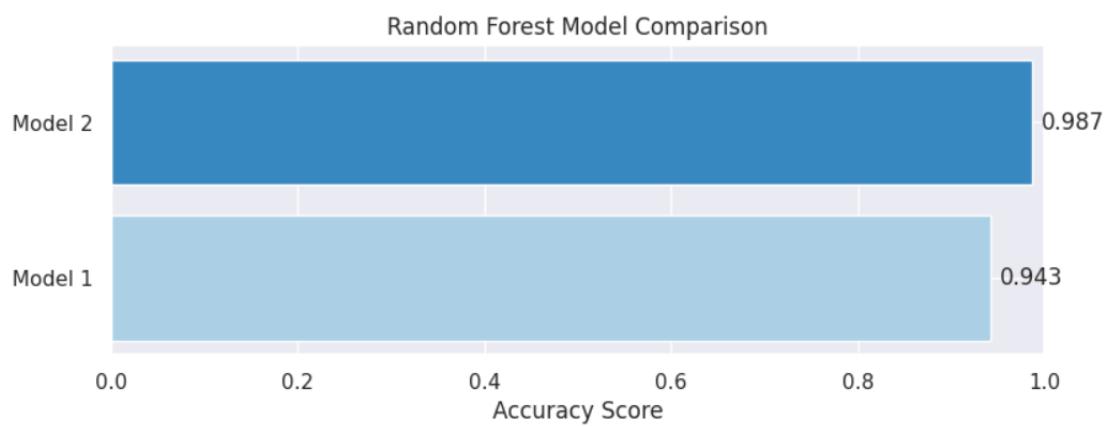
So sánh hai mô hình:



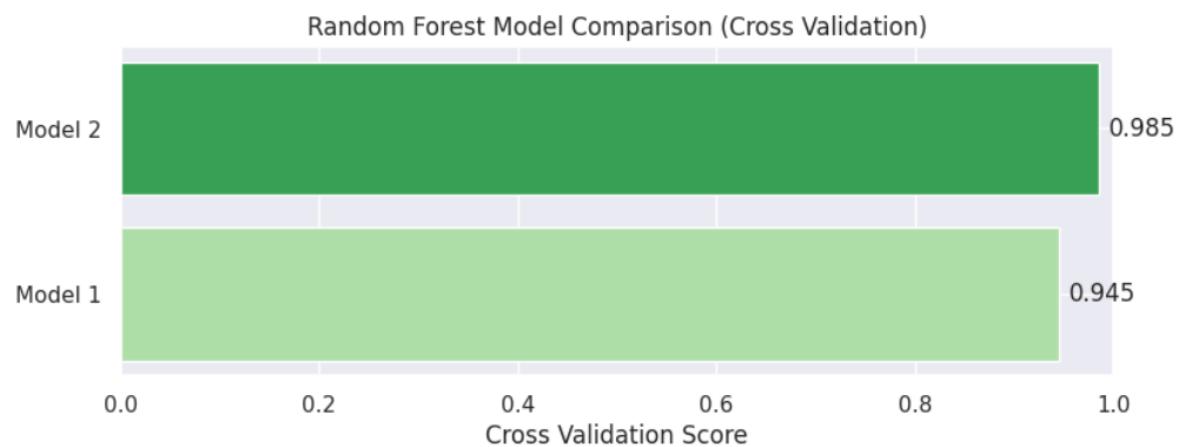
Hình 52. Bảng so sánh ma trận nhầm lẫn (confusion matrix) của hai mô hình.



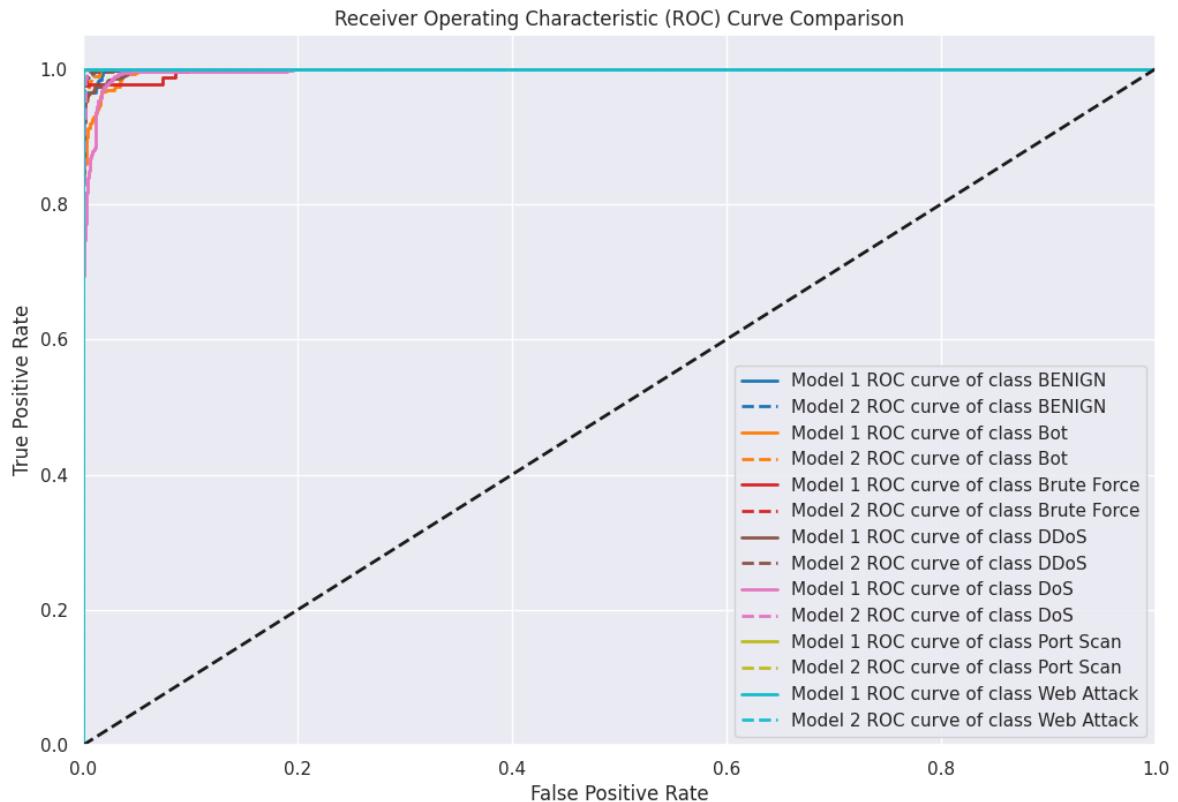
Hình 53. Bảng trực quan hóa chỉ số hiệu suất quan trọng của hai mô hình .



Hình 54. Bảng so sánh độ chính xác của hai mô hình.



Hình 55. Bảng so sánh chỉ số Cross Validation Score của hai mô hình.



Hình 56. Bảng so sánh ROC Curve của hai mô hình

Nhận xét kết quả huấn luyện của hai mô hình:

	Acuracy	Precision	Recall	F1-Score	Cross Validation Score
Model 1	0.942857	0.95	0.94	0.94	0.945791
Model 2	0.986743	0.98	0.98	0.98	0.987238

- Hiệu suất tổng thể (Accuracy và Cross-validation Score): Biểu đồ so sánh độ chính xác và cross-validation score cho thấy rõ ràng Model 2 có hiệu suất tổng thể cao hơn đáng kể so với Model 1.
- Hiệu suất trên từng lớp (Classification Report Heatmaps):
 - Mô hình 1 (`max_depth = 4`) hoạt động khá tốt trên các lớp lớn như BENIGN, Brute Force, Port Scan, Web Attack, nhưng có hiệu suất

thấp hơn trên các lớp như DoS (đặc biệt là Recall thấp) và DDoS (Precision thấp)

- Mô hình 2 (`max_depth = 6`) cho thấy sự cải thiện đáng kể trên hầu hết các lớp so với Model 1. Các chỉ số Precision, Recall và F1-score cao hơn đồng đều trên các lớp, bao gồm cả các lớp mà Model 1 gặp khó khăn như DoS và DDoS.
- Ma trận nhầm lẫn (Confusion Matrix Heatmaps): Ma trận nhầm lẫn giúp làm rõ các lỗi phân loại.
 - Mô hình 1 cho thấy số lượng mẫu bị phân loại sai lớn hơn, đặc biệt là việc nhầm lẫn giữa BENIGN và các loại DoS, cũng như giữa các loại DoS với nhau.
 - Mô hình 2 có số lượng mẫu bị phân loại sai ít hơn đáng kể trên hầu hết các cặp lớp. Các giá trị ngoài đường chéo chính nhỏ hơn nhiều so với Model 1, cho thấy mô hình ít bị nhầm lẫn giữa các lớp hơn.

Nhận xét chung: Mô hình 2 thể hiện hiệu suất vượt trội hơn hẳn so với mô hình 1 trên cả các chỉ số tổng thể (Accuracy, Cross-validation) và hiệu suất chi tiết trên từng lớp (Precision, Recall, F1-score) được thể hiện qua báo cáo phân loại và ma trận nhầm lẫn. Việc tăng độ sâu tối đa của cây từ 4 lên 6 đã giúp mô hình học được các ranh giới quyết định phức tạp hơn và phân loại các loại tấn công, đặc biệt là các loại DoS và DDoS, chính xác hơn.

3. Thu thập dữ liệu thực tế trong Lab để huấn luyện mô hình:

Bộ dữ liệu CIC-IDS-2017 được sử dụng ở trên chỉ dùng để đánh giá tính khả thi và độ hiệu quả của mô hình. Để mô hình có thể phát hiện các cuộc tấn công thực tế một cách chính xác thì nhóm đã thu thập dữ liệu lưu lượng mạng trong Lab bao gồm lưu lượng mạng lành tính (BENIGN), lưu lượng mạng tấn công DoS/DDoS và lưu lượng mạng tấn công Data Exfiltration, lưu thành file .csv và đưa vào mô hình để huấn luyện.

```

feature_logger.py > $ Flow
 1 import time
 2 import json
 3 import logging
 4 import argparse
 5 import numpy as np
 6 import pandas as pd
 7 from scapy.all import sniff, IP, TCP, UDP
 8 from collections import defaultdict
 9 from threading import Thread, Lock
10 import os
11
12
13 CONFIG = {
14     "INTERFACE": "ens34",
15     "FEATURES_PATH": "optimal_features.json", # File chứa danh sách đặc trưng
16     "OUTPUT_CSV_PATH": "/var/log/feature_log.csv", # File CSV đầu ra
17     "FLOW_TIMEOUT": 10,
18     "PACKET_THRESHOLD_TO_LOG": 20 # Ghi log flow khi đạt 20 gói tin để có đủ thống kê
19 }
20
21 class Flow:
22     def __init__(self, flow_id):
23         self.flow_id = flow_id; self.packets = []; self.start_time = time.time(); self.last_seen = time.time(); self.flags = defaultdict(int)
24     def add_packet(self, packet, direction):
25         self.packets.append({'pkt': packet, 'dir': direction}); self.last_seen = time.time()
26     def if_packet_is_tcp(self, packet):
27         flags_str = str(packet[TCP].flags)
28         if 'F' in flags_str: self.flags['FIN Flag Count'] += 1
29         if 'S' in flags_str: self.flags['SYN Flag Count'] += 1
30         if 'R' in flags_str: self.flags['RST Flag Count'] += 1
31         if 'P' in flags_str: self.flags['PSH Flag Count'] += 1
32         if 'A' in flags_str: self.flags['ACK Flag Count'] += 1
33         if 'U' in flags_str: self.flags['UNG Flag Count'] += 1
34         if 'E' in flags_str: self.flags['ECE Flag Count'] += 1
35         if 'C' in flags_str: self.flags['CWE Flag Count'] += 1
36     def calculate_features(self, feature_names):
37         if len(self.packets) < 2: return None

```

Hình 57. Script python thu thập dữ liệu mạng.

```
(ids_env) ad@Suricata:~/Downloads/ids$ sudo /home/ad/Downloads/ids/ids_env/bin/python3 feature_logger.py -i ens34 --label BENIGN --output my_dataset.csv
[sudo] password for ad:
2025-07-06 14:56:48,624 - Feature Logger initialized. Logging features with label 'BENIGN' to my_dataset.csv
2025-07-06 14:56:48,624 - Starting Feature Logger on interface ens34...
2025-07-06 14:57:04,667 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 39098, 6) with label 'BENIGN'
2025-07-06 14:57:15,594 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 39098, 6) with label 'BENIGN'
2025-07-06 14:57:21,716 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 39102, 6) with label 'BENIGN'
2025-07-06 14:57:21,981 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 39098, 6) with label 'BENIGN'
2025-07-06 14:57:32,745 - Logged features for flow ('18.81.56.131', 67, '18.81.56.254', 68, 17) with label 'BENIGN'
2025-07-06 14:57:33,299 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 41306, 6) with label 'BENIGN'
2025-07-06 14:57:44,494 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 41306, 6) with label 'BENIGN'
2025-07-06 14:57:47,096 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 41306, 6) with label 'BENIGN'
2025-07-06 14:57:50,258 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 41306, 6) with label 'BENIGN'
2025-07-06 14:57:52,763 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 39098, 6) with label 'BENIGN'
2025-07-06 14:58:12,603 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:14,594 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:15,998 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:21,551 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:22,443 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:23,505 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 41306, 6) with label 'BENIGN'
2025-07-06 14:58:26,878 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:33,853 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:36,627 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:36,718 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 55638, 6) with label 'BENIGN'
2025-07-06 14:58:44,412 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 40432, 6) with label 'BENIGN'
2025-07-06 14:58:44,415 - Logged features for flow ('192.168.192.129', 80, '192.168.57.200', 40448, 6) with label 'BENIGN'
```

Hình 58. Thu thập dữ liệu mạng trong Lab.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Average Packets	Total Length	Destination Port	Flow Bytes/s	Flow Duration	Idle Min	idle IAT	Min Flow IAT	1/Tot. Flow	Win_byt/s	Fwd IAT	Tot. Flow	ATime	Max	AttackType
2	486.1	9722	80	33767.0103	108302.5932	0	0.00939927	0	0.00984431	1514	26886	0.1090586	0.0093994	0.03684431	BENIGN
3	1138.55	22771	80	422957.6773	99317.5922	0	0.00931632	0	0.00976025	1514	191	0.010116	0.009531632	0.09762025	BENIGN
4	66.77	13334	40594	4695263.387	28360.56996	0	0.00917572	0	0.00960803	1514	349	0.0327783	0.0172220	0.00960803	BENIGN
5	1029.86	21297	80	86177.22448	22148.1232	0	0.00263037	0	0.0099554	1514	191	0.0043726	0.00263037	0.0099554	BENIGN
6	1092.05	21841	80	201569.54	10492.6593	0	0.000417687	0	0.00295932	1514	191	0.0093952	0.0004177	0.00295932	BENIGN
7	1179.7	23594	40594	26124771.19	9031.285776	0	0.001419231	0	0.0082744	1514	550	0.0269654	0.00141923	0.0082744	BENIGN
8	1248.9	24978	80	141363.483	176693.4398	0	0.001038627	0	0.16253258	1514	4443	0.1697339	0.010038623	0.16253258	BENIGN
9	751.15	15023	40594	326547.73	44625.23223	0	0.009472575	0	0.04129312	1514	617	0.0409733	0.002472283	0.04129312	BENIGN
10	864.45	17289	40594	1843391.17	9378.91065	0	0.00057425	0	0.00251627	1514	684	0.0109107	0.00057425	0.00251627	BENIGN
11	1251.2	25024	40594	1908574.97	13111.35292	0	0.00016017	0	0.00115992	1514	818	0.0030422	0.00116012	0.00115992	BENIGN
12	1177.95	23599	40594	1329841.35	17674.44611	0	0.001724406	0	0.02875781	1514	851	0.0327637	0.00172441	0.02875781	BENIGN
13	737.5	14750	40594	602811.915	24468.66035	0	0.001769806	0	0.02882385	1514	918	0.0332663	0.0176981	0.02882385	BENIGN
14	198.5	3970	40594	553784.136	7169.246674	1.38E-05	6.92E-05	1.38E-05	0.00282533	482	982	0.0013142	6.92E-05	0.00028253	BENIGN
15	144.45	2899	40594	342559.134	8433.580399	2.53E-05	7.24E-05	2.53E-05	0.00272751	494	982	0.001375	7.24E-05	0.00027251	BENIGN
16	70.4	1408	40594	164918.876	8537.530898	0	5.98E-05	0	0.00036287	140	982	0.0011363	5.98E-05	0.00036287	BENIGN
17	1076.6	21592	80	162366.61	9982.585907	0	0.00206538	0	0.0390731	1514	4452	0.0398126	0.0020654	0.0390731	BENIGN
18	235.7	4714	40594	128258.523	36753.8929	2.74E-05	0.001544187	2.74E-05	0.02893653	1008	985	0.0239336	0.00154419	0.02893653	BENIGN
19	174.95	3499	40594	336178.438	10408.16307	2.31E-05	0.000136074	2.31E-05	0.00282317	482	1019	0.0025854	0.00013607	0.00282317	BENIGN
20	225.25	4505	40594	509911.092	8834.838867	2.83E-05	7.13E-05	2.83E-05	0.00030899	497	1019	0.0013554	7.13E-05	0.00030899	BENIGN
21	138.4	2768	80	378911.014	7305.145264	3.08E-05	7.06E-05	3.08E-05	0.00028283	1514	4452	0.0013413	7.06E-05	0.00028283	BENIGN
22	1039.9	20798	40594	337195.44	61679.36325	0	0.005522766	0	0.03764272	1514	1019	0.1049325	0.00552277	0.03764272	BENIGN
23	174.9	3498	40594	419406.42	8340.358734	2.38E-07	2.38E-07	0.00223216	480	1086	0.0015054	7.92E-05	0.00022316	BENIGN	
24	1135.85	22717	40594	2718692.16	8355.855942	0	5.77E-05	0	0.00053382	1514	1086	0.001096	5.77E-05	0.00053382	BENIGN
25	248.9	4978	40594	100126.339	49717.18788	0	0.003919074	0	0.04055357	1514	1086	0.0744624	0.003919074	0.04055357	BENIGN
26	133	2660	40594	153654.437	17311.57303	0.00010014	0.000808352	0.0001001	0.02893469	170	1119	0.0153587	0.00080835	0.02893469	BENIGN
27	1119.15	22383	80	643881.255	34762.62093	0	0.001756367	0	0.02898288	1514	4452	0.033371	0.00175637	0.02898288	BENIGN
28	868.3	17366	40594	2246292.58	7730.960846	0	0.000180219	0	0.00201535	1514	1186	0.0034242	0.00018022	0.00201535	BENIGN
29	781.75	15635	40594	214602.257	72855.71098	0	0.004176077	0	0.03708099	1514	1153	0.0793455	0.00417608	0.03708099	BENIGN
30	1304.5	26090	80	360568.819	72375.89299	0	0.00447666	0	0.0726788	1514	4452	0.0850565	0.00447666	0.0726788	BENIGN

Hình 59. Dữ liệu mạng được đánh dấu và lưu vào trong file .csv.

Tạo module AI-Based IDS trong Lab mô phỏng:

Nhóm đã viết một script Python ‘realtime_ids_detector.py’ có thể chạy độc lập, có chức năng bắt và phân tích traffic mạng trong thời gian thực, sau đó sử dụng

một mô hình học máy đã được huấn luyện trước để xác định các hành vi tấn công tinh vi.

- Class Flow đóng vai trò như một hồ sơ cho mỗi cuộc hội thoại mạng. Mỗi khi một luồng mới được phát hiện, một đối tượng Flow sẽ được tạo ra để theo dõi nó.
- Khi được tạo, nó sẽ ghi lại thời điểm bắt đầu (start_time), khởi tạo một danh sách rỗng để chứa các gói tin (packets), và một bộ đếm để thống kê các cờ TCP (flags).
- Mỗi khi một gói tin thuộc về luồng này được bắt, hàm này sẽ được gọi. Nó thêm gói tin vào danh sách, cập nhật lại thời gian cuối cùng nhìn thấy (last_seen), và nếu gói tin là TCP, nó sẽ phân tích và tăng bộ đếm cho các cờ tương ứng (SYN, ACK, FIN...)
- Tính toán Đặc trưng (calculate_features) được gọi khi một luồng kết thúc. Nó thực hiện các phép tính thống kê phức tạp trên danh sách các gói tin đã thu thập để tạo ra một vector đặc trưng

```

24 class Flow:
25     def __init__(self, flow_id):
26         self.flow_id = flow_id; self.packets = []; self.start_time = time.time(); self.last_seen = time.time(); self.flags = defaultdict(int)
27     def add_packet(self, packet, direction):
28         self.packets.append((('pkt': packet, 'dir': direction)); self.last_seen = time.time()
29         if packet.haslayer(TCP):
30             flags_str = str(packet[TCP].flags)
31             if 'F' in flags_str: self.flags['FIN Flag Count'] += 1
32             if 'S' in flags_str: self.flags['SYN Flag Count'] += 1
33             if 'R' in flags_str: self.flags['RST Flag Count'] += 1
34             if 'P' in flags_str: self.flags['PSH Flag Count'] += 1
35             if 'A' in flags_str: self.flags['ACK Flag Count'] += 1
36             if 'U' in flags_str: self.flags['URG Flag Count'] += 1
37             if 'E' in flags_str: self.flags['ECE Flag Count'] += 1
38             if 'C' in flags_str: self.flags['CWE Flag Count'] += 1
39     def calculate_features(self, feature_names):
40         if len(self.packets) < 2: return None
41         feature_dict = {}; fwd_packets = [p for p in self.packets if p['dir'] == 'fwd']; bwd_packets = [p for p in self.packets if p['dir'] == 'bwd']
42         feature_dict['Flow Duration'] = (self.last_seen - self.start_time) * 1e6
43         first_packet = self.packets[0]['pkt']
44         feature_dict['Destination Port'] = first_packet.dport if hasattr(first_packet, 'dport') else 0
45         bwd_pkt_lengths = [len(p) for p in bwd_packets]; all_pkt_lengths = [len(p['pkt']) for p in self.packets]
46         feature_dict['Bwd Packet Length Max'] = max(bwd_pkt_lengths) if bwd_pkt_lengths else 0; feature_dict['Fwd Packet Length Max'] = max([len(p) for p in fwd_packets])
47         feature_dict['Average Packet Size'] = np.mean(all_pkt_lengths) if all_pkt_lengths else 0; feature_dict['Bwd Packet Length Min'] = min(bwd_pkt_lengths) if bwd_pkt_lengths else 0
48         feature_dict['Min Packet Length'] = min(all_pkt_lengths) if all_pkt_lengths else 0
49         init_win_bwd = next(([p[TCP].window for p in bwd_packets if p.haslayer(TCP)], 0)); feature_dict['Init Win Bwd'] = init_win_bwd
50         init_win_fwd = next(([p[TCP].window for p in fwd_packets if p.haslayer(TCP)], 0)); feature_dict['Init Win Bytes Forward'] = init_win_fwd
51         duration_sec = self.last_seen - self.start_time; feature_dict['Flow Bytes/s'] = sum(all_pkt_lengths) / duration_sec if duration_sec > 0 else 0
52         feature_dict['Min Seg Size Forward'] = min([p[IP].ihl * 4 for p in fwd_packets if p.haslayer(IP)]) if any(p.haslayer(IP) for p in fwd_packets) else 0
53         feature_dict['Fwd IAT Total'] = sum(np.diff(p.time for p in fwd_packets)) if len(fwd_packets) > 1 else 0
54         feature_dict['Bwd IAT Min'] = min(np.diff(p.time for p in bwd_packets)) if len(bwd_packets) > 1 else 0
55         timestamps = [p['pkt'].time for p in self.packets]; feature_dict['Idle Min'] = min(np.diff(timestamps)) if len(timestamps) > 1 else 0
56         feature_dict['Total Backward Packets'] = len(bwd_packets); feature_dict['Total Fwd Packets'] = len(fwd_packets)
57         feature_dict['Total Length of Bwd Packets'] = sum(bwd_pkt_lengths); feature_dict['Total Length of Fwd Packets'] = sum(len(p) for p in fwd_packets)
58         feature_dict['Bwd Packet Length Std'] = np.std(bwd_pkt_lengths) if bwd_pkt_lengths else 0; feature_dict['Flow Packets/s'] = len(all_pkt_lengths) / duration_sec if duration_sec > 0 else 0
59         feature_dict['Flow TAT Std'] = np.std(np.diff(timestamps)) if len(timestamps) > 1 else 0; feature_dict['Bwd Duration Mean'] = np.mean(bwd_pkt_lengths)
60

```

Hình 60. Class Flow - Quản lý trạng thái luồng mạng.

- Class IDSDetector là trung tâm điều phối, chịu trách nhiệm cho toàn bộ quá trình từ việc bắt gói tin đến đưa ra cảnh báo.
- Các chức năng chính:

- **Xác định luồng (_get_flow_key)**: Nhận một gói tin làm đầu vào và tạo ra một "định danh luồng" (flow key) duy nhất dựa trên 5 yếu tố (5-tuple): IP nguồn, IP đích, cổng nguồn, cổng đích và giao thức.
- **Xử lý Gói tin (process_packet)**: lọc bỏ các gói tin không phải là TCP/UDP và sử dụng _get_flow_key để xác định gói tin này thuộc về luồng nào. Sau khi thêm gói tin, nó sẽ kiểm tra xem luồng đó đã đạt đến ngưỡng về số lượng gói tin (PACKET_THRESHOLD_FOR_ML) chưa. Nếu đã đạt, nó sẽ ngay lập tức gửi luồng đó đi phân tích.
- **Xử lý Luồng Timeout (check_flow_timeouts)**: Hàm này chạy trên một luồng (thread) riêng biệt, hoạt động song song với luồng bắt gói tin. Cứ mỗi 10 giây nó sẽ lặp qua tất cả các luồng đang hoạt động và kiểm tra xem có luồng nào bị bỏ quên hay không. Nếu một luồng bị timeout, nó sẽ được gửi đi phân tích để đảm bảo không bỏ sót các cuộc tấn công có lưu lượng thấp hoặc các luồng đã kết thúc tự nhiên.
- **Phân tích và Dự đoán (_analyze_and_predict)**: nhận một đối tượng Flow đã kết thúc (do đạt ngưỡng hoặc timeout), sau đó gọi **flow.calculate_features()** để tính toán vector đặc trưng. Vector đặc trưng này sau đó được chuẩn hóa bằng đối tượng scaler đã tải. Cuối cùng, dữ liệu đã chuẩn hóa được đưa vào mô hình **self.model.predict()** để đưa ra dự đoán. Nếu kết quả dự đoán không phải là BENIGN thì gọi hàm **_log_alert** để tạo cảnh báo.
- **Ghi Cảnh báo (_log_alert)**: Khi một cuộc tấn công được phát hiện, tạo log cảnh báo chi tiết dưới định dạng JSON chứa các thông tin quan trọng như timestamp, IP/cổng nguồn và đích, loại tấn công được phát hiện (signature), và tên của engine phát hiện, giúp việc tích hợp vào các hệ thống SIEM như ELK Stack trở nên dễ dàng.

```

66     class IDSVerifier:
67         def __init__(self, config):
68             self.config = config
69             self.model = joblib.load(config["MODEL_PATH"])
70             self.scaler = joblib.load(config["SCALER_PATH"])
71             with open(config["FEATURES_PATH"], "r") as f:
72                 self.feature_names = json.load(f)
73             self.flows = {}
74             self.ip_packet_count = defaultdict(int)
75             self.ip_packet_timestamp = defaultdict(float)
76             self.portscan_tracker = defaultdict(lambda: {"ports": set(), "first_seen": time.time()})
77             self.lock = Lock()
78             logging.basicConfig(format="%(levelname)s:%(name)s:%(message)s", level=logging.DEBUG, handlers=[logging.FileHandler(config["LOG_FILE_PATH"])]
79             logging.info("IDS Detector initialized successfully.")
80
81     def _get_flow_key(self, packet):
82         if IP in packet and (TCP in packet or UDP in packet):
83             ip_pair = tuple(sorted((packet[IP].src, packet[IP].dst)))
84             port_pair = tuple(sorted((packet[TCP].sport, packet[TCP].dport)))
85             proto = packet[IP].proto
86             return (ip_pair[0], port_pair[0], ip_pair[1], port_pair[1], proto)
87         return None
88
89     def _log_alert(self, signature, packet, category="intrusion_detection", metadata=None):
90         alert = {
91             "@timestamp": datetime.now(tz.tzutc()).isoformat(),
92             "event": {"kind": "alert", "category": category, "type": "network"},
93             "source": {"ip": packet[IP].src, "port": packet.sport if packet.haslayer(TCP) or packet.haslayer(UDP) else 0},
94             "destination": {"ip": packet[IP].dst, "port": packet.dport if packet.haslayer(TCP) or packet.haslayer(UDP) else 0},
95             "network": {"protocol": "tcp" if packet.haslayer(TCP) else "udp"},
96             "ids": {"engine": "ML+Hybrid-IDS", "signature": signature}
97         }
98         if metadata:
99             alert['metadata'] = metadata
100        logging.warning(json.dumps(alert))
101

```

Hình 61. Class *IDSVerifier* – Bộ não của hệ thống.

Ngăn chặn tấn công bằng Iptables trên máy Switch:

Nhóm đã viết script Python ‘active_response_agent.py’ có thể chạy độc lập dùng để phản ứng trước cuộc tấn công được phát hiện bởi AI-Based IDS bằng cách gửi lệnh chặn IP kẻ tấn công đến Iptables của máy Switch.

- Khi được khởi tạo, agent sẽ thiết lập một kết nối đến máy chủ Elasticsearch bằng các thông tin được định nghĩa trong “CONFIG”.
- Tạo một hàng đợi (block_queue), là nơi các IP cần chặn sẽ được xếp hàng chờ xử lý đồng thời tạo ra một tập hợp (set) tên là blocked_ips để lưu lại các IP đã bị chặn, nhằm tránh thực hiện các hành động trùng lặp.

```

class ActiveResponseAgent:
    def __init__(self, config):
        self.config = config
        self.es_client = Elasticsearch(
            self.config["ES_HOSTS"],
            basic_auth=(self.config["ES_USER"], self.config["ES_PASS"])
        )
        self.blocked_ips = set()
        self.block_queue = Queue()
        logging.info("Active Response Agent initialized.")

```

Hình 62. Hàm khởi tạo.

- Hàm **run()** giúp chương trình hoạt động liên tục, cứ mỗi 15 giây, nó sẽ thực hiện một truy vấn đến Elasticsearch để tìm tất cả các bản ghi có **event.kind: "alert"** xuất hiện trong vòng 30 giây qua. Khi tìm thấy các cảnh báo mới,

nó không thực hiện hành động chặn ngay lập tức. Thay vào đó, nó gọi hàm **process_alert()** để phân tích và đưa IP của kẻ tấn công vào **block_queue**.

```
def run(self):
    worker_thread = Thread(target=self.block_worker, daemon=True)
    worker_thread.start()
    logging.info("Blocker worker thread started.")

    while True:
        try:
            logging.info(f"Querying Elasticsearch for new alerts...")
            query = { "query": { "bool": { "filter": [ { "range": { "@timestamp": { "gte": self.config['QUERY_TIME_RANGE'] }}} ], "term": { "event.kind": "attack" } } } }
            response = self.es_client.search(index=self.config["ES_INDEX_PATTERN"], body=query)

            hits = response.get('hits', {}).get('hits', [])
            if hits:
                logging.warning(f"Found {len(hits)} new alert(s).")
                for hit in hits:
                    self.process_alert(hit)
            else:
                logging.info("No new alerts found.")
        except Exception as e:
            logging.error(f"An error occurred in the main loop: {e}", exc_info=True)

        time.sleep(self.config['LOOP_SLEEP_INTERVAL'])
```

Hình 63. Hàm *run()*.

- Hàm **block_worker()** chạy trên một luồng (thread) riêng biệt và độc lập ngay từ khi agent khởi động.
- Nó liên tục lắng nghe hàng đợi **block_queue**. Hàm **block_queue.get()** sẽ tự động tạm dừng luồng nếu hàng đợi trống, giúp tiết kiệm tài nguyên CPU. Khi có một IP mới được đưa vào hàng đợi, luồng worker này sẽ thức dậy, lấy IP đó ra và thực hiện toàn bộ quy trình chặn thông qua hàm **block_ip()**.

```
def block_worker(self):
    """Luồng công nhân (worker) lắng nghe từ hàng đợi và thực hiện chặn."""
    while True:
        src_ip = self.block_queue.get() # Lệnh này sẽ chờ cho đến khi có mục mới trong queue
        if src_ip is None: # Tín hiệu để dừng
            break

        # Kiểm tra lại lần nữa trước khi chặn
        if src_ip in self.config["WHITELIST_IPS"] or src_ip in self.blocked_ips:
            continue

        ssh = None
        try:
            logging.info(f"Worker processing IP: {src_ip}")
            ssh = paramiko.SSHClient()
            ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            ssh.connect(
                self.config["SWITCH_IP"],
                username=self.config["SWITCH_USER"],
                password=self.config["SWITCH_PASS"],
                timeout=10
            )
            cmd = f"sudo iptables -I FORWARD 1 -s {src_ip} -j DROP"
            stdin, stdout, stderr = ssh.exec_command(cmd)
            error = stderr.read().decode().strip()
            if error:
                logging.error(f"IPTables error for {src_ip}: {error}")
            else:
                logging.warning(f"ACTION: Successfully blocked source IP: {src_ip}")
                self.blocked_ips.add(src_ip)
        except Exception as e:
            logging.error(f"SSH error for {src_ip}: {e}", exc_info=False)
        finally:
            if ssh: ssh.close()
            self.block_queue.task_done()
```

Hình 64. Hàm *block_worker()*.

- Hàm process_alert() đóng vai trò như một bộ tiền xử lý, nhận một bản ghi cảnh báo từ Elasticsearch và kiểm tra nguồn gốc của cảnh báo đó.
- Trích xuất chính xác trường source.ip từ cấu trúc log tương ứng của mỗi nguồn và đưa IP đó vào hàng đợi luồng worker xử lý.

```

def process_alert(self, hit):
    source = hit.get('_source', {})
    src_ip = source.get('source', {}).get('ip') or \
             source.get('suricata', {}).get('eve', {}).get('src_ip')

    if src_ip and src_ip not in self.blocked_ips:
        logging.info(f"Adding IP {src_ip} to block queue.")
        self.block_queue.put(src_ip)

```

Hình 65. Hàm process_alert() .

- Kiểm tra IP đầu vào với danh sách trắng (WHITELIST_IPS) và danh sách các IP đã bị chặn (blocked_ips) để đảm bảo không chặn nhầm hoặc lặp.
- Khi cần chặn IP, script tạo một phiên kết nối SSH hoàn toàn mới đến máy Switch, sau đó thực thi lệnh sudo iptables -I FORWARD 1 -s {src_ip} -j DROP để chèn một luật chặn vào vị trí đầu tiên của chuỗi FORWARD, đảm bảo luật này có hiệu lực ngay lập tức.
- Sau khi thực thi xong thì đóng kết nối SSH.

```

# Kiểm tra lại lần nữa trước khi chặn
if src_ip in self.config["WHITELIST_IPS"] or src_ip in self.blocked_ips:
    continue

ssh = None
try:
    logging.info(f"Worker processing IP: {src_ip}")
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(
        self.config["SWITCH_IP"],
        username=self.config["SWITCH_USER"],
        password=self.config["SWITCH_PASS"],
        timeout=10
    )
    cmd = f"sudo iptables -I FORWARD 1 -s {src_ip} -j DROP"
    stdin, stdout, stderr = ssh.exec_command(cmd)
    error = stderr.read().decode().strip()
    if error:
        logging.error(f"IPTables error for {src_ip}: {error}")
    else:
        logging.warning(f"ACTION: Successfully blocked source IP: {src_ip}")
        self.blocked_ips.add(src_ip)
except Exception as e:
    logging.error(f"SSH error for {src_ip}: {e}", exc_info=False)
finally:
    if ssh: ssh.close()
    self.block_queue.task_done()

```

Hình 66. Code thực hiện chặn IP.

```

2025-07-03 10:11:15,523 - INFO - Querying Elasticsearch for new alerts...
2025-07-03 10:11:15,527 - INFO - POST http://192.168.193.153:9200/filebeat-*/_search [status:200 duration:0.003s]
2025-07-03 10:11:15,527 - WARNING - Found 9 new alert(s).
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Worker processing IP: 192.168.192.129
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Connected (version 2.0, client OpenSSH_8.9p1)
2025-07-03 10:11:15,609 - INFO - Authentication (password) successful!
2025-07-03 10:11:16,090 - WARNING - ACTION: Successfully blocked source IP: 192.168.192.129

```

Hình 67. Kết quả chạy Script để chặn IP kẻ tấn công.

4.3. Mô phỏng tấn công.

4.3.1. Tấn công Cross-site scripting, SQL injection và Brute force mật khẩu.

Ngữ cảnh: Trong mạng nội bộ, có một máy đang host một trang web DVWA chứa nhiều lỗ hổng có thể bị khai thác. Attacker sử dụng script python để tấn công vào trang web đó. Hệ thống mạng có tích hợp Suricata sẽ phát hiện các cuộc tấn công đó và filebeat sẽ thu thập log của Suricata về Elasticsearch để phân tích hành vi tấn công. Đồng thời gửi Email về cho quản trị viên.

Kiểm thử:

Phía Attacker: viết script tấn công bằng python nhắm vào website DVWA trên máy webserver trong mạng nội bộ.

- **Tấn công Cross-site Scripting:**

```
~/Desktop/xss.txt [Read Only] - Mousepad
File Edit Search View Document Help
+ C x ↻ ⌛ 🔍 ⌂ ⌂ ⌂
1 <script>alert(1)</script>
2 "><script>alert(1)</script>
3 <img src=x onerror=alert(1)>
4 "><img src=x onerror=alert(1)>
5 <svg/onload=alert(1)>
6 <iframe src="javascript:alert(1)">
7

def xss_attack(target):
    with open("xss.txt", "r") as f:
        for i in f.readlines():
            r = s.get(f"{target}/DVWA/vulnerabilities/xss_r/?name={i}", cookies={"security": "low"})
```

Hình 68. Script python tấn công XSS.

Nội dung file "xss.txt" chứa mã JavaScript độc hại (script alert, onerror, onload, iframe) nhằm thực thi các lệnh như hiển thị cảnh báo hoặc tải nội dung không mong muốn. Dữ liệu được truyền vào bao gồm thông tin giả lập về lỗ hổng (DWA/vulnerabilities) và cookie với giá trị "security": "low", có thể được sử dụng để khai thác thêm.

- **Tấn công SQL injection:**

```
1 ' OR 1=1 --
2 ' OR '1'='1
3 ' OR '1'='1' --
4 ' OR '1'='1' #
5 admin' --
6 admin' #
7 admin' or '1'='1
8 1' or '1' = '1
9 1 OR 1=1
10 1' or 1=1--
11 1') or ('1'='1

def sqli_attack(target):
    with open("sql.txt", "r") as f:
        for i in f.readlines():
            r = s.get(f"{target}/DVWA/vulnerabilities/sql/?id={i}&Submit=Submit", cookies={"security": "low"})
```

Hình 69. Script python tấn công SQL injection.

Đọc từng dòng trong file, chứa các payload SQL injection (như ' OR 1=1 --, ' OR '1'='1',...). Gửi yêu cầu đến URL được xây dựng từ target với đường dẫn /DVWA/vulnerabilities/sql/?id= cùng payload và cookie "security": "low". Mục tiêu là khai thác lỗ hổng SQL injection trong tham số id.

- **Tấn công Brute force mật khẩu:**

```

23 def password_attack(target):
24     target_ip = target.split("http://")[1]
25     print("[*] Starting brute-force attack... ")
26     os.system(f"hydra -l admin -P /usr/share/wordlists/rockyou.txt {target_ip} http-post-form '/DWWA/login.php:username='USER'&password='PASS'&Login=Login:F=Login failed'")
27

```

Hình 70. Script tấn công brute force mật khẩu.

Hàm password_attack(target) dùng hydra để brute force mật khẩu cho tài khoản "admin" trên form login /DWWA/login.php, sử dụng danh sách từ "rockyou.txt" và kiểm tra dựa trên thông báo lỗi "Login failed".

Phía người quản trị: Thiết lập các Rule cảnh báo trên Suricata, kiểm tra log trên Suricata và Elasticsearch và cảnh báo qua email.

- Rule cảnh báo của Suricata: *nano /var/lib/suricata/rules/suri.rules*

```

GNU nano 7.2
/var/lib/suricata/rules/suri.rules
#XSS detect
alert http any any --> any any (msg:"XSS Attempt - <script> tag"; flow:to_server,established; content:"<script>"; nocase; http_uri; classtype:web-application-attack; sid:205; rev:1)
alert http any any --> any any (msg:"XSS Attempt - <script> in body"; flow:to_server,established; content:"<script>"; nocase; http_client_body; classtype:web-application-attack; sid:206; rev:1)
alert http any any --> any any (msg:"XSS Attempt - DOM event onload"; flow:to_server,established; content:"onload"; nocase; http_uri; sid:202; rev:1)
alert http any any --> any any (msg:"XSS Attempt - DOM event onclick"; flow:to_server,established; content:"onclick"; nocase; http_uri; sid:203; rev:1)
alert http any any --> any any (msg:"XSS Attempt - DOM event onerror"; flow:to_server,established; content:"onerror"; nocase; http_uri; sid:204; rev:1)
alert http any any --> any any (msg:"XSS Attempt - javascript URI"; flow:to_server,established; content:"javascript:"; nocase; http_uri; sid:205; rev:1)
alert http any any --> any any (msg:"XSS Attempt - encoded script"; flow:to_server,established; content:"%3Cscript%"; nocase; http_uri; sid:206; rev:1)
alert http any any --> any any (msg:"XSS Attempt - <img onerror>"; flow:to_server,established; content:<img>; nocase; http_uri; content:"onerror"; nocase; http_header; sid:207; rev:1)
alert http any any --> any any (msg:"XSS Attempt - document.cookie leak"; flow:to_server,established; content:"document.cookie"; nocase; http_uri; sid:208; rev:1)
alert http any any --> any any (msg:"XSS Attempt - alert()"; flow:to_server,established; content:"alert("; nocase; http_uri; sid:209; rev:1)

#DDos detect
alert icmp any any --> any any (msg:"[DoS] ICMP Ping of Death - oversized packet"; itype:8; dsiz:>1000; sid:300; rev:1)
alert icmp any any --> any any (msg:"[DoS] ICMP Flood attack detected"; itype:8; threshold: type both, track_by_src, count 20, seconds 5; sid:301; rev:1)
alert icmp any any --> any any (msg:"[DoS] ICMP with TCP flags - suspicious spoof (hping3)"; itype:8; dsiz:>100; sid:302; rev:2)

#Brute-force detect
alert http any any --> any any (msg:"[Brute Force] Multiple POST requests to /login.php"; flow:to_server,established; content:"POST"; http_method; content:"Content-Type: application/x-www-form-urlencoded"); sid:401; rev:1)
#alert http any any --> any any (msg:"[Brute Force] Login failed detected in response"; flow:from_server,established; content:"Login failed"; http_response); sid:402; rev:1)
alert http any any --> any any (msg:"[Brute Force] Suspicious User-Agent (Hydra)"; content:"User-Agent:3a] hydra"; http_header; sid:402; rev:1)

#SQLi detect
alert http any any --> any any (msg:"[SQLi] Suspicious SQL keyword in URL"; flow:to_server,established; content:"' OR '1='1"; nocase; http_uri; sid:501; rev:1)
alert http any any --> any any (msg:"[SQLi] UNION SELECT detected in URL"; flow:to_server,established; content:"UNION SELECT"; nocase; http_uri; sid:502; rev:1)
alert http any any --> any any (msg:"[SQLi] SQL injection attempt in POST login"; flow:to_server,established; content:"POST"; http_method; content:"/login.php"); sid:503; rev:1)
alert http any any --> any any (msg:"[SQLi] Potential brute-force SQLi from same source"; flow:to_server,established; content:"id="; http_uri; content:"Content-Type: application/x-www-form-urlencoded"); sid:504; rev:1)
alert http any any --> any any (msg:"[SQLi] Blind SQL Injection - sleep() function detected"; flow:to_server,established; content:"sleep("; nocase; http_header); sid:505; rev:1)


```

Hình 71. Rule cảnh báo trên Suricata.

- Kiểm tra log Suricata sau cuộc tấn công: *tail -f /var/log/suricata/fast.log*

```

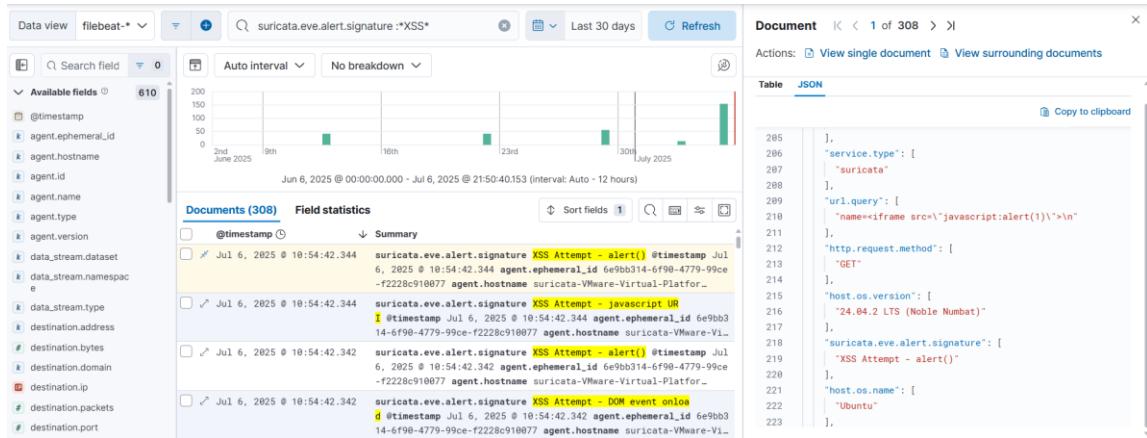
suricata@suricata-Virtual-Platform:~$ sudo tail -f /var/log/suricata/fast.log
07/06/2025-10:54:42.342753 [**] [1:209:1] XSS Attempt - alert() [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60917 -> 192.168.100.4:80
07/06/2025-10:54:42.344228 [**] [1:205:1] XSS Attempt - javascript URI [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60917 -> 192.168.100.4:80
07/06/2025-10:54:42.344228 [**] [1:209:1] XSS Attempt - alert() [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60917 -> 192.168.100.4:80
07/06/2025-10:54:48.667899 [**] [1:500:1] [SQLi] Suspicious SQL keyword in URL [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:48.668836 [**] [1:500:1] [SQLi] Suspicious SQL keyword in URL [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:48.678921 [**] [1:500:1] [SQLi] Suspicious SQL keyword in URL [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:48.673393 [**] [1:504:1] [SQLi] Potential brute-force SQLi from same source [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:48.676846 [**] [1:500:1] [SQLi] Suspicious SQL keyword in URL [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:48.685589 [**] [1:505:1] [SQLi] Blind SQL Injection - sleep() function detected [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60922 -> 192.168.100.4:80
07/06/2025-10:54:54.606346 [**] [1:400:2] [Brute Force] Multiple POST requests to /Login.php [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.100.1:60944 -> 192.168.100.4:80

```

Hình 72. Suricata đã phát hiện thành công và ghi log lại.

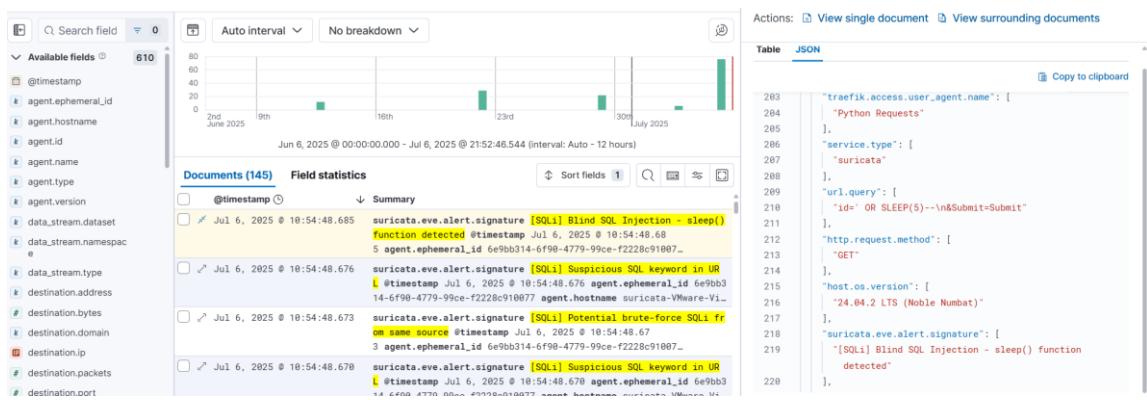
- Kiểm tra log trên Elasticsearch:

- Log của tấn công XSS:



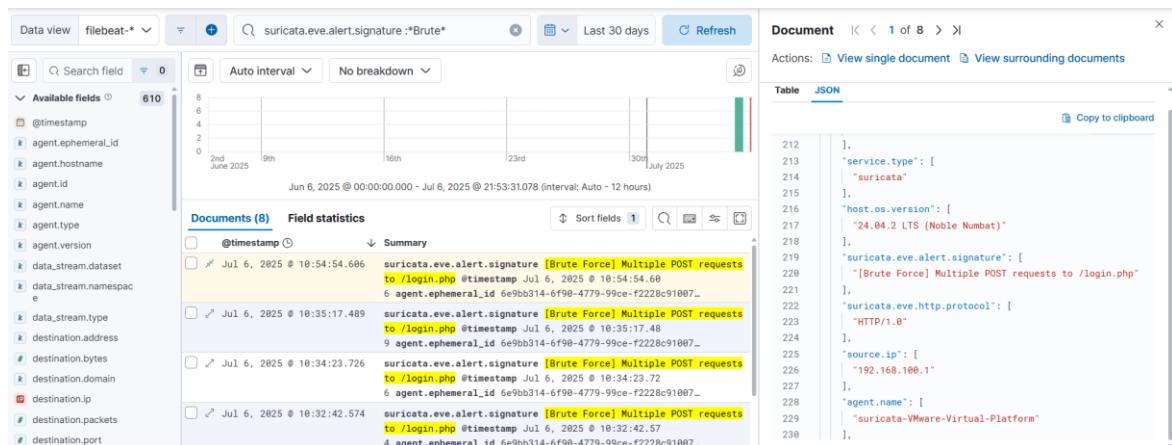
Hình 73. Log của cuộc tấn công XSS được lưu trữ trên Elasticsearch.

- Log của tấn công SQL injection:



Hình 74. Log của cuộc tấn công SQL injection được lưu trữ trên Elasticsearch.

- Log của tấn công Brute force:



Hình 75. Log của cuộc tấn công Brute force được lưu trữ trên Elasticsearch.

- Kiểm tra gmail của quản trị viên:

LogAlert for TOP TEN OSWAP ALERT: /var/log/kibana/kibana.log

H 22520442@gm.uit.edu.vn to me 10:54 AM (11 hours ago) ⌂ ⌂ ⌂

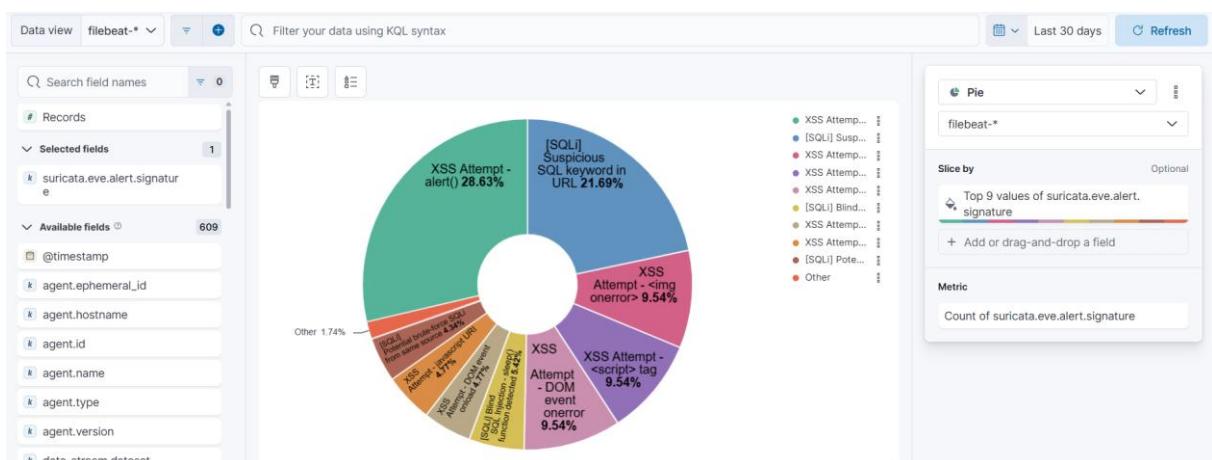
LogAlert Name: TOP TEN OSWAP ALERT
Date/Time: Sun Jul 06 2025 10:54:51 GMT+0700 (Indochina Time)
Hostname: elk-VMware-Virtual-Platform
File Path: /var/log/kibana/kibana.log

Matched Lines:
("service":{"node":{"roles":["background_tasks","ui"]}}, "ecs":{"version":"8.11.0"}, "@timestamp":"2025-07-06T10:54:49.230+07:00", "message":"Server log: 20 log entries in the last 1 min. Alert when ≥ 1 ; TOP TEN OSWAP ALERT is active ;; 20 log entries have matched the following conditions: destination.address equals 192.168.100.4;,[View alert details](/app/observability/alerts/c8b1354f-e3b8-4491-a2a4-b28186b126ae)," , "log": {"level": "WARN", "logger": "plugins.actions.server-log"}, "process": {"pid": 1062, "uptime": 10293.934628478}, "trace": {"id": "4e34f47611684bb60087fc74d7483775"}, "transaction": {"id": "d475ea1a159a49dd"}}

End of alert.

Hình 76. Người quản trị đã nhận được Gmail cảnh báo sau khi có cuộc tấn công.

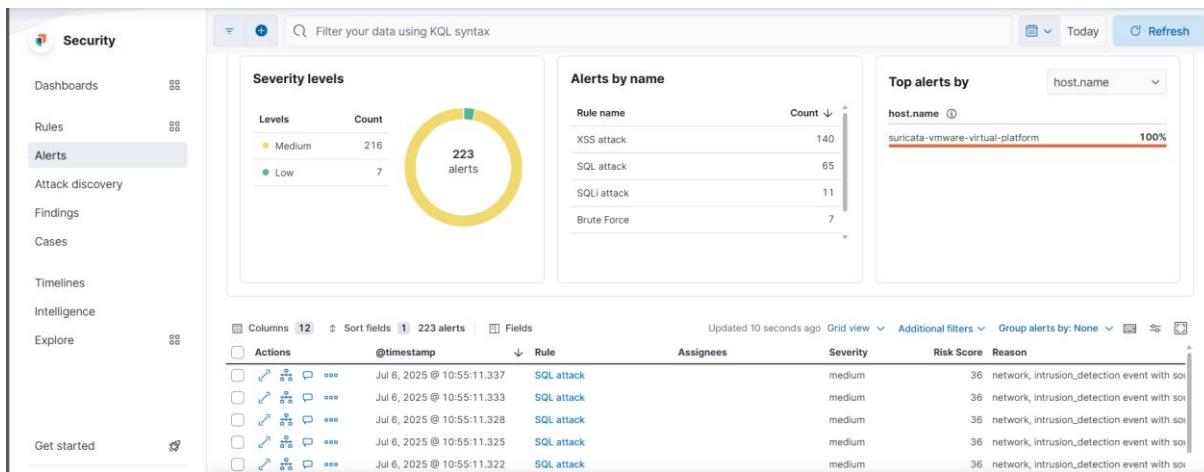
- Tạo Dashboard để trực quan hóa lượng log từ các cuộc tấn công.



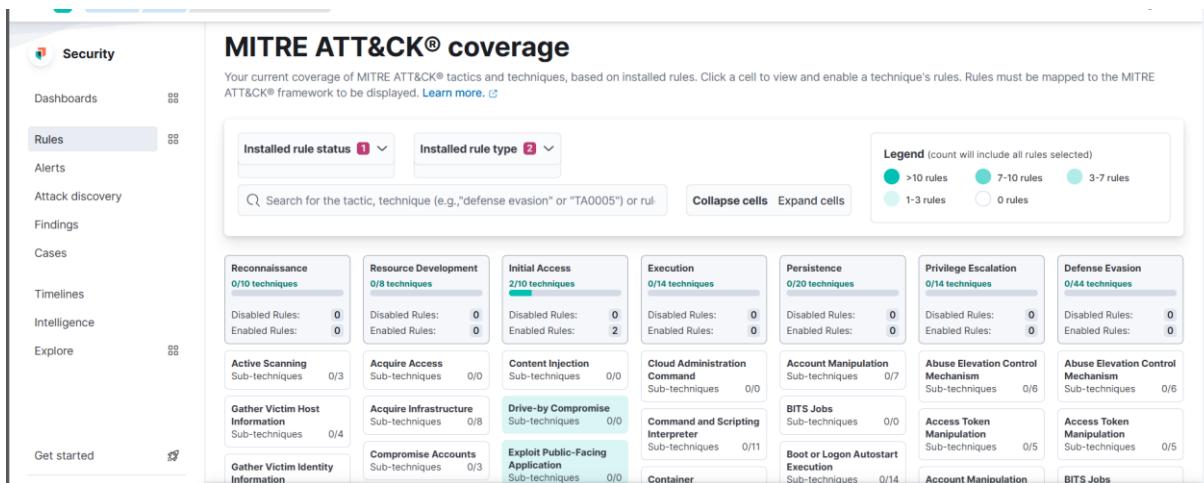
Hình 77. Giao diện tạo Dashboard trên Kibana.

Để tạo được biểu đồ của các cuộc tấn công, cần một giá trị được ghi nhận là đặc trưng của cuộc tấn công đó qua số lượng log đã gửi về Elasticsearch. Ở đây trường suricata.eve.alert.signature là trường đặc trưng được chọn. Biểu đồ sẽ thống kê % các cuộc tấn công xảy ra và so sánh kết quả qua biểu đồ.

- Đánh giá rủi ro và ánh xạ với các kỹ thuật trong MITRE ATT&CK.



Hình 78. Chức năng hỗ trợ người quản trị đánh giá mức độ rủi ro của các cuộc tấn công.



Hình 79. Các kỹ thuật trong bảng MITRE ATT&CK.

Elasticsearch cũng hỗ trợ ánh xạ các log của các cuộc tấn công vào kỹ thuật của MITRE ATT&CK. Ở bảng trên 2 cuộc tấn công XSS và SQLI được đánh xạ đến 2 kỹ thuật là Drive-by Compromise và Exploit Public-Facing Application.

4.3.2. Tấn công malware mô phỏng WannaCry.

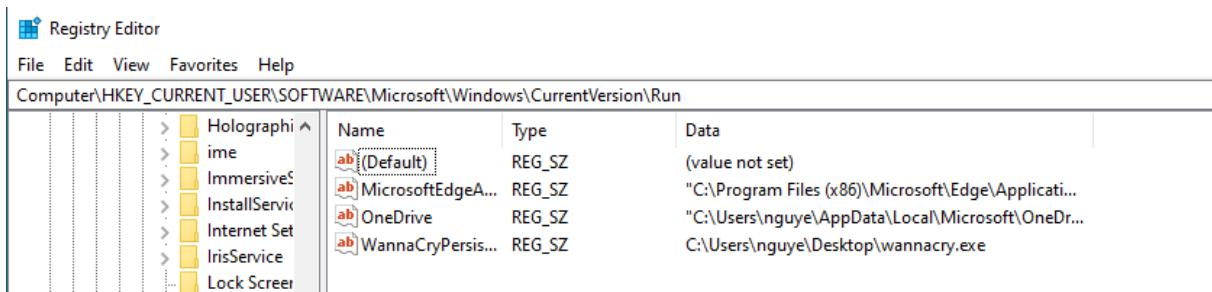
Ngữ cảnh: Một máy người dùng trong mạng nội bộ sử dụng win10 nhận được 1 email có chứa một file mã độc được gửi từ attacker. Người dùng đó tải về và bị nhiễm mã độc WannaCry. Sysmon được cài đặt trên máy người dùng để phát hiện các hành vi bất thường như tạo tiến trình lạ, ghi vào Registry,... Để đưa được log của sysmon về Elasticsearch, người quản trị sử dụng công cụ Winlogbeat.

- Hoạt động của Malware



Hình 80. Mã độc được khởi chạy và máy nạn nhân đang bị tấn công.

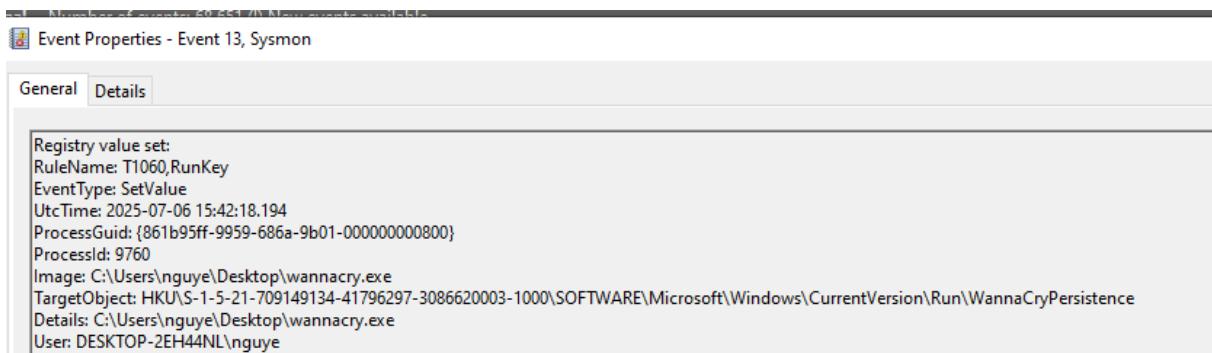
Sau khi chạy mã độc lên, thì sẽ hiện một GUI để thông báo đến nạn nhân là cần tiền chuộc để có được key giải mã các file. Ta có thể thấy các file đã được mã hóa và không thể đọc được.



Hình 81. Registry của máy nạn nhân sau khi chạy mã độc.

Ngoài ra mã độc còn để lại một Registry key WannaCryPersistence trong Registry của nạn nhân. Khi nạn nhân khởi động lại máy sẽ khởi chạy file mã độc lên.

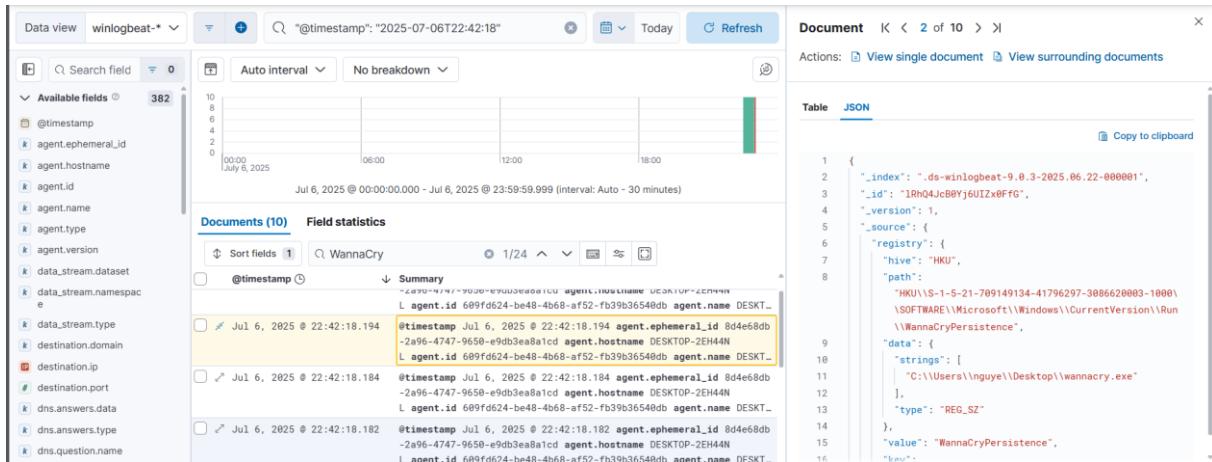
- Kiểm tra log của sysmon



Hình 82. Log của công cụ giám sát sysmon.

Từ log của công cụ Sysmon ta có thể thấy được mã độc đã tạo 1 Registry Key vào thư mục .../Run của Registry của máy nạn nhân.

- **Kiểm tra log trên Elasticsearch**



Hình 83. Log của Win10 đã được Winlogbeat gửi về Elasticsearch.

Ta có thể thấy log đã được Winlogbeat gửi về Elasticsearch để phân tích và điều tra. Một số đặc điểm của mã độc cũng được ghi lại trong log.

4.3.3. Kiểm thử hệ thống AI-based IDS.

Kịch bản 1: Phát hiện và ngăn chặn tấn công DoS/DDoS

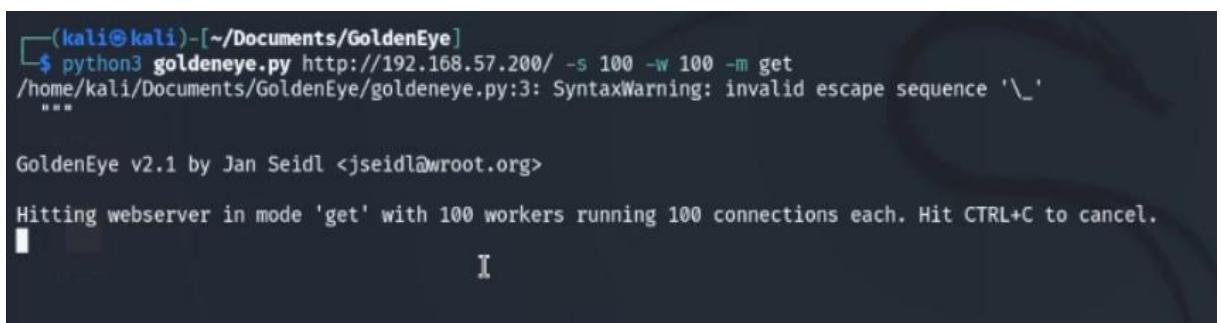
- Ngữ cảnh: Attacker sử dụng các công cụ tấn công DoS/DDoS để làm gián đoạn hoặc ngừng khả năng cung cấp dịch vụ của Server, từ đó có thể sử dụng các cuộc tấn công khác để khai thác lúc Server bị sập .
- Chạy Script phát hiện và cảnh báo tấn công AI-Based IDS.



```
ad@Suricata:~/Downloads/ids$ sudo ./ids_env/bin/python3 realtime_ids_detector.py
INFO:root:IDS Detector initialized successfully.
INFO:root:Starting Hybrid IDS Detector on interface ens34...
```

Hình 84. Chạy Script phát hiện tấn công

- Dùng công cụ GoldenEye để thực hiện tấn công DoS đến máy Server.
- Tạo 100 tiến trình với 100 luồn để gửi gói tin đến server nhằm mục đích gây lụt hệ thống [13].



```
(kali㉿kali)-[~/Documents/GoldenEye]
$ python3 goldeneye.py http://192.168.57.200/ -s 100 -w 100 -m get
/home/kali/Documents/GoldenEye/goldeneye.py:3: SyntaxWarning: invalid escape sequence '\_'
"""

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>
Hitting webserver in mode 'get' with 100 workers running 100 connections each. Hit CTRL+C to cancel.
```

Hình 85. Sử dụng GoldenEye để tấn công.

- Module AI-Based IDS bắt được luồng dữ liệu và phân tích, dự đoán đó là cuộc tấn công DoS đồng thời gửi cảnh báo đến Kibana và lệnh chặn IP đến Iptables của máy Switch.

```

INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41646, 6). Total packets: 8
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41900, 6). Total packets: 5
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41872, 6). Total packets: 5
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41882, 6). Total packets: 5
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41934, 6). Total packets: 4
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41890, 6). Total packets: 5
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41932, 6). Total packets: 4
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40462, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40472, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40500, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40510, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41920, 6). Total packets: 5
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40374, 6). Total packets: 14
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41658, 6). Total packets: 9
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40428, 6). Total packets: 14
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 41730, 6). Total packets: 9
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57
.200', 40406, 6). Total packets: 14
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57

```

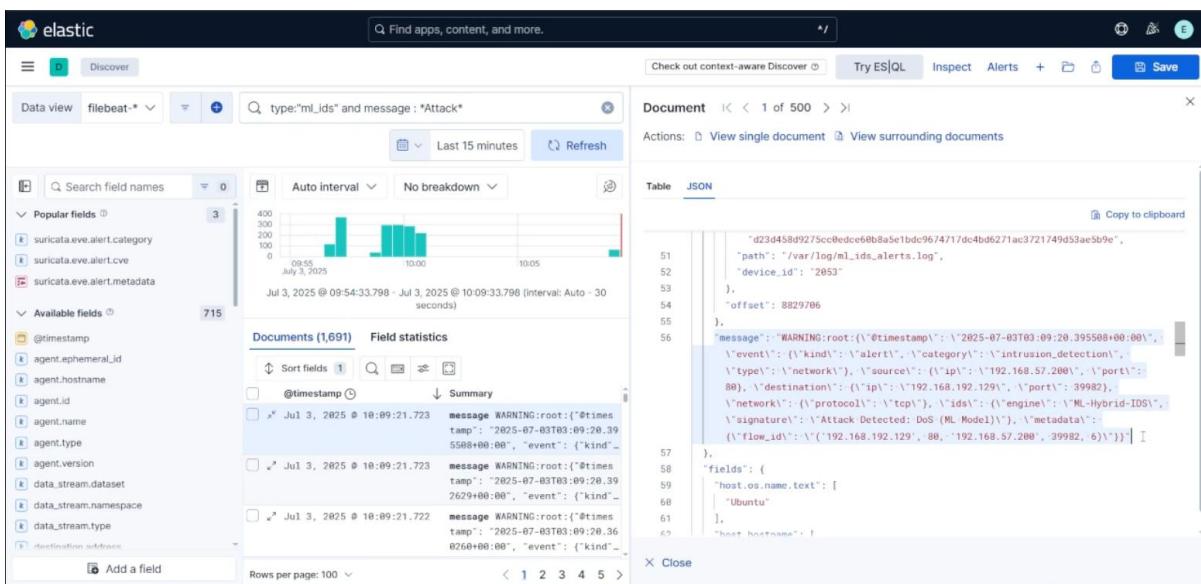
Hình 86. Module bắt được luồng gói tin tấn công DoS.

```

ad@Suricata: ~/Downloads/ids
INFO:root:DEBUG: Model prediction for flow ('192.168.192.129', 80, '192.168.57.200', 3882, 6)
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57.200', 42308, 6). Total packets: 3
INFO:root:DEBUG: Model prediction for flow ('192.168.192.129', 80, '192.168.57.200', 38824, 6) is: DoS
WARNING:root:{ "@timestamp": "2025-07-03T03:09:08.496962+00:00", "event": { "kind": "alert", "category": "intrusion_detection", "type": "network", "source": { "ip": "192.168.192.129", "port": 38824}, "destination": { "ip": "192.168.57.200", "port": 80}, "network": { "protocol": "tcp"}, "ids": { "engine": "ML-Hybrid-IDS", "signature": "Attack Detected: DoS (ML Model)"}, "metadata": { "flow_id": "(192.168.192.129", 80, '192.168.57.200', 38824, 6)}}
WARNING:root:{ "@timestamp": "2025-07-03T03:09:08.496057+00:00", "event": { "kind": "alert", "category": "intrusion_detection", "type": "network", "source": { "ip": "192.168.192.129", "port": 38840}, "destination": { "ip": "192.168.57.200", "port": 80}, "network": { "protocol": "tcp"}, "ids": { "engine": "ML-Hybrid-IDS", "signature": "Attack Detected: DoS (ML Model)"}, "metadata": { "flow_id": "(192.168.192.129", 80, '192.168.57.200', 38840, 6)}}
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57.200', 42248, 6). Total packets: 3
INFO:root:DEBUG: Model prediction for flow ('192.168.192.129', 80, '192.168.57.200', 38868, 6) is: DoS
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 80, '192.168.57.200', 42294, 6). Total packets: 3
WARNING:root:{ "@timestamp": "2025-07-03T03:09:08.514896+00:00", "event": { "kind": "alert", "category": "intrusion_detection", "type": "network", "source": { "ip": "192.168.192.129", "port": 38868}, "destination": { "ip": "192.168.57.200", "port": 80}, "network": { "protocol": "tcp"}, "ids": { "engine": "ML-Hybrid-IDS", "signature": "Attack Detected: DoS (ML Model)"}, "metadata": { "flow_id": "(192.168.192.129", 80, '192.168.57.200', 38868, 6)}}

```

Hình 87. Đưa ra cảnh báo về tấn công.



Hình 88. Cảnh báo tấn công trên Kibana.

```

2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:00,507 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:15,523 - INFO - Querying Elasticsearch for new alerts...
2025-07-03 10:11:15,527 - INFO - POST http://192.168.193.153:9200/filebea
t-*/_search [status:200 duration:0.003s]
2025-07-03 10:11:15,527 - WARNING - Found 9 new alert(s).
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.57.200 to block queue.
2025-07-03 10:11:15,527 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Worker processing IP: 192.168.192.129
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:11:15,528 - INFO - Connected (version 2.0, client OpenSSH_8
.9p1)
2025-07-03 10:11:15,609 - INFO - Authentication (password) successful!
2025-07-03 10:11:16,090 - WARNING - ACTION: Successfully blocked source I
P: 192.168.192.129

```

Hình 89. Chạy script gửi lệnh chặn ip của kẻ tấn công đến máy switch.

```

ad@switch:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 74448 packets, 179M bytes)
pkts bytes target prot opt in     out     source          destination
Chain FORWARD (policy ACCEPT 358K packets, 307M bytes)
pkts bytes target prot opt in     out     source          destination
 670K  94M ACCEPT   all  --  ens38  ens40  0.0.0.0/0      0.0.0.0/0
 446K  829M ACCEPT  all  --  ens40  ens38  0.0.0.0/0      0.0.0.0/0

Chain OUTPUT (policy ACCEPT 1193K packets, 946M bytes)
pkts bytes target prot opt in     out     source          destination
ad@switch:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 74470 packets, 179M bytes)
pkts bytes target prot opt in     out     source          destination
Chain FORWARD (policy ACCEPT 358K packets, 307M bytes)
pkts bytes target prot opt in     out     source          destination
 2913  214K DROP    all  --  *       *       192.168.192.129  0.0.0.0/0
 673K  95M ACCEPT   all  --  ens38  ens40  0.0.0.0/0      0.0.0.0/0
 449K  830M ACCEPT  all  --  ens40  ens38  0.0.0.0/0      0.0.0.0/0

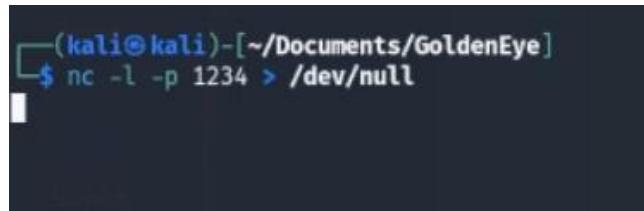
Chain OUTPUT (policy ACCEPT 1202K packets, 947M bytes)
pkts bytes target prot opt in     out     source          destination
ad@switch:~$ 

```

Hình 90. Rule chặn IP của kẻ tấn công được thêm vào Iptables của máy Switch.

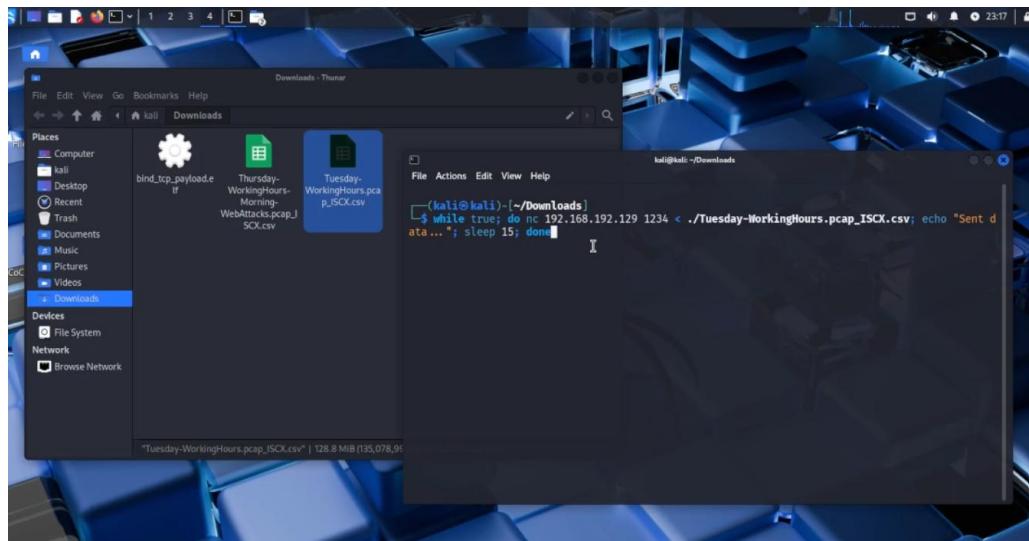
Kịch bản 2: Phát hiện và ngăn chặn tấn công Data Exfiltration

- Ngữ cảnh: Attacker đã chiếm quyền điều khiển của máy Server và tiến hành đánh cắp dữ liệu nhạy cảm.
- Thực hiện: Trên máy Attacker, mở kết nối nghe trên cổng 1234



Hình 91. Dùng netcat để tạo server nhận file từ máy Victim.

- Trên máy Server dùng netcat để chuyển file đến cho máy Attacker



Hình 92. Thực hiện chuyển file đến cho máy Attacker trên máy Server.

- Trên máy giám sát phát hiện lưu lượng mạng bất thường, phân tích và dự đoán tấn công đồng thời gửi log đến Kibana và gửi lệnh chặn IP đến Iptables của máy Switch.

```

ad@Suricata: ~/Downloads/ids
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 10
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 11
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 12
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 14
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 15
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 16
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 17
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 18
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 19
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 20
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 21
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 22
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 23
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 24
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 25
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 26
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 27

```

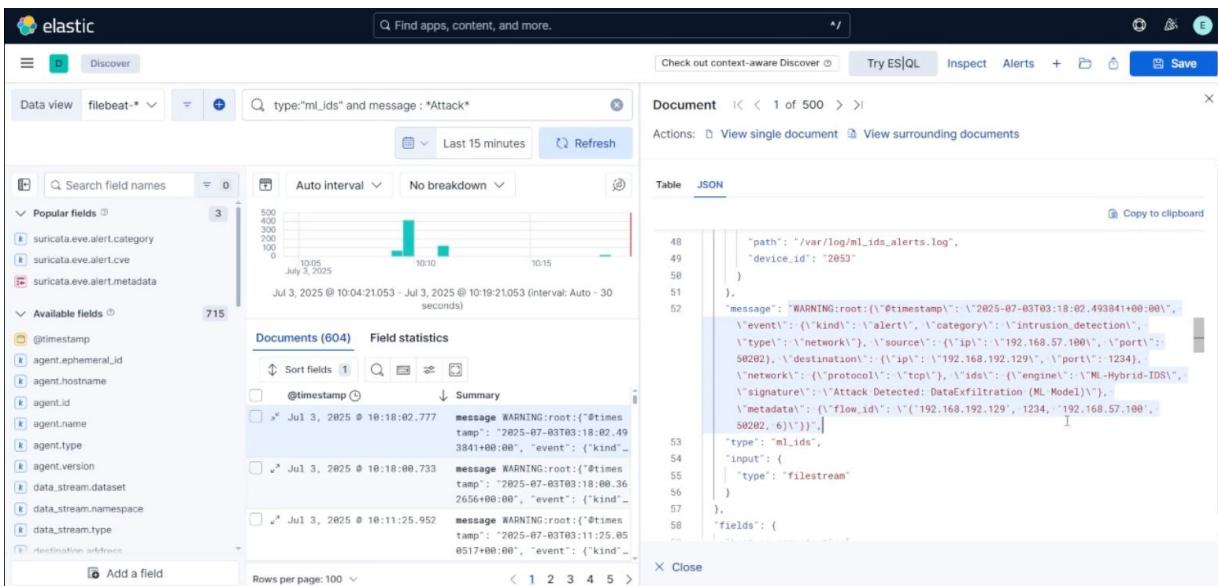
Hình 93. Module bắt được luồng dữ liệu của cuộc tấn công.

```

ad@Suricata: ~/Downloads/ids
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 10
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 11
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 12
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 13
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 14
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 15
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 16
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 17
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 18
INFO:root:DEBUG: Model prediction for flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6) is: DataExfiltration
WARNING:root:{ "timestamp": "2025-07-03T03:18:00.362656+00:00", "event": { "Kind": "alert", "category": "intrusion_detection", "type": "network"}, "source": { "ip": "192.168.57.100", "port": 50202}, "destination": { "ip": "192.168.192.129", "port": 1234}, "network": { "protocol": "tcp"}, "ids": { "engine": "ML-Hybrid-IDS", "signature": "Attack Detected: DataExfiltration (ML Model)"}, "metadata": { "flow id": "192.168.192.129", "id": 192.168.57.100, "port": 50202, "proto": "tcp", "src_ip": "192.168.57.100", "src_port": 50202, "dst_ip": "192.168.192.129", "dst_port": 1234}}}
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 19
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 20
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 21
INFO:root:DEBUG: Packet added to flow ('192.168.192.129', 1234, '192.168.57.100', 50202, 6). Total packets: 22

```

Hình 94. Mô hình phát hiện và gửi cảnh báo đến Kibana.



Hình 95. Cảnh báo trên Kibana.

```
ad@Suricata:~/Downloads/ids$ sudo ./ids_env/bin/python3 active_response_agent.py
2025-07-03 10:20:22,550 - INFO - Active Response Agent initialized.
2025-07-03 10:20:22,551 - INFO - Blocker worker thread started.
2025-07-03 10:20:22,551 - INFO - Querying Elasticsearch for new alerts...
2025-07-03 10:20:22,555 - INFO - POST http://192.168.193.153:9200/filebeat-*/_search [status:200 duration:0.004s]
2025-07-03 10:20:22,555 - INFO - No new alerts found.
2025-07-03 10:20:37,571 - INFO - Querying Elasticsearch for new alerts...
2025-07-03 10:20:37,573 - INFO - POST http://192.168.193.153:9200/filebeat-*/_search [status:200 duration:0.002s]
2025-07-03 10:20:37,573 - INFO - No new alerts found.
2025-07-03 10:20:52,586 - INFO - Querying Elasticsearch for new alerts...
2025-07-03 10:20:52,589 - INFO - POST http://192.168.193.153:9200/filebeat-*/_search [status:200 duration:0.003s]
2025-07-03 10:20:52,590 - WARNING - Found 3 new alert(s).
2025-07-03 10:20:52,590 - INFO - Adding IP 192.168.57.100 to block queue.
2025-07-03 10:20:52,590 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:20:52,590 - INFO - Adding IP 192.168.192.129 to block queue
.
2025-07-03 10:20:52,590 - INFO - Worker processing IP: 192.168.192.129
2025-07-03 10:20:52,607 - INFO - Connected (version 2.0, client OpenSSH_8.9p1)
2025-07-03 10:20:52,671 - INFO - Authentication (password) successful!
2025-07-03 10:20:53,209 - WARNING - ACTION: Successfully blocked source IP: 192.168.192.129
```

Hình 96. Chạy Script gửi lệnh chặn IP kẻ tấn công đến máy Switch.

```
ad@switch:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 74545 packets, 179M bytes)
pkts bytes target     prot opt in     out     source               destination
Chain FORWARD (policy ACCEPT 358K packets, 307M bytes)
pkts bytes target     prot opt in     out     source               destination
 685K  95M ACCEPT    all  --  ens38  ens40  0.0.0.0/0
 466K  966M ACCEPT   all  --  ens40  ens38  0.0.0.0/0
Chain OUTPUT (policy ACCEPT 1233K packets, 1085M bytes)
pkts bytes target     prot opt in     out     source               destination
ad@switch:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 74579 packets, 179M bytes)
pkts bytes target     prot opt in     out     source               destination
Chain FORWARD (policy ACCEPT 358K packets, 307M bytes)
pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *      192.168.192.129  0.0.0.0/0
 688K  96M ACCEPT    all  --  ens38  ens40  0.0.0.0/0
 470K 1101M ACCEPT   all  --  ens40  ens38  0.0.0.0/0
Chain OUTPUT (policy ACCEPT 1239K packets, 1220M bytes)
pkts bytes target     prot opt in     out     source               destination
ad@switch:~$
```

Hình 97. Rule chặn IP của kẻ tấn công được thêm vào Iptables trên máy Switch.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận.

Nghiên cứu đã chứng minh được tầm quan trọng của bộ công cụ tích hợp gồm Suricata và ELK stack. Hai công cụ này đã hỗ trợ cho nhau để giúp kỹ sư an ninh mạng giám sát được các cuộc tấn công nguy hiểm đối với mô hình mạng doanh nghiệp của mình. Từ công cụ Suricata với khả năng phát hiện và ngăn chặn các cuộc tấn công, ELK stack sẽ hỗ trợ được khả năng phân tích, truy vết, tìm kiếm và lập các báo cáo giúp người quản trị có thể đối phó và đưa ra các biện pháp kịp thời với các cuộc tấn công ngày càng hiện đại.

Bên cạnh đó, việc tích hợp mô hình trí tuệ nhân tạo vào bộ công cụ trên đã tạo nên sự khác biệt trong cách giám sát mô hình mạng. Mô hình trí tuệ nhân tạo đã giúp Suricata giám sát toàn diện hơn, nhanh chóng và hiệu quả đối với các cuộc tấn công hiện đại.

2. Hướng phát triển.

Tiếp tục nghiên cứu phát triển khả năng giám sát và phản ứng của mô hình IDS AI-based. Giúp mô hình có thể thông minh hơn phát hiện nhiều cuộc tấn công phức tạp hơn.

Áp dụng thêm các chức năng của bộ công cụ ELK stack vào khả năng phân tích log. Tạo ra nhiều biểu đồ, ánh xạ đầy đủ vào bảng MITRE ATT&CK.

Phát triển thêm một module AI để phân loại các cuộc tấn công từ số lượng log không lồ đã bắt được từ Elasticsearch.

TÀI LIỆU THAM KHẢO

- [1] Muhammad Bilal Azam, "A SECURITY INFORMATION AND," 2025. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/895242/Azam_Bilal.pdf?sequence=2.
- [2] Anphat, Admin, "Suricata - Giải pháp bảo mật mạng hiệu quả, linh hoạt," 11 10 2024. [Online]. Available: <https://www.anphat.vn/tin-cong-nghe/suricata-giai-phap-bao-mat-mang-hieu-qua-linh-hoat>.
- [3] Peter Manev, "The Other Side of Suricata," 06 10 2021. [Online]. Available: <https://www.stamus-networks.com/blog/the-other-side-of-suricata>. [Accessed 07 07 2025].
- [4] huyhoc1310, "ELK Stack - 3 anh em siêu nhân trong quản lý logs," 18 2 2020. [Online]. Available: <https://viblo.asia/p/elk-stack-3-anh-em-sieu-nhan-trong-quan-ly-logs-oOVIYLAZ8W>. [Accessed 07 07 2025].
- [5] Elastic Admin, "Filebeat overview," 01 01 2025. [Online]. Available: <https://www.elastic.co/guide/en/beats/filebeat/8.18/filebeat-overview.html#:~:text=Filebeat%>. [Accessed 07 07 2025].
- [6] Truong Văn Qui, "Filebeat: Thu thập log hiệu quả cho hệ thống của bạn," 05 05 2025. [Online]. Available: <https://viblo.asia/p/filebeat-thu-thap-log-hieu-qua-cho-he-thong-cua-ban-3RIL5xG8JbB>. [Accessed 07 07 2025].
- [7] S. Moderator, "MITRE ATT&CK Framework là gì?," 15 01 2022. [Online]. Available: <https://whitehat.vn/threads/mitre-att-ck-framework-la-gi-mang-ve-cho-me-duoc-khong.16187/>. [Accessed 07 05 2025].
- [8] scikit-learn.org, "RandomForestClassifier," [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>. [Accessed 01 07 2025].
- [9] C. Ducre, "How to Install Elastic Stack on Ubuntu 22.04 LTS," 18 08 2023. [Online]. Available: <https://blog.devops.dev/how-to-install-elastic-stack-on-ubuntu-22-04-lts-18c3d9120494>. [Accessed 05 07 2027].
- [10] nattycoder, "Elastic-Stack-Deployment-with-Filebeat-and-Suricata," 07 02 2024. [Online]. Available: <https://github.com/nattycoder/Elastic-Stack-Deployment-with-Filebeat-and-Suricata/blob/main/filebeat-config/filebeat-setup.md>. [Accessed 04 06 2025].
- [11] Elastic Admin, "Download Winlogbeat," 24 06 2025. [Online]. Available: <https://www.elastic.co/downloads/beats/winlogbeat>. [Accessed 07 07 2025].
- [12] jhuckaby, "LogAlert," 28 09 2023. [Online]. Available: <https://github.com/jhuckaby/logalert>. [Accessed 18 06 2025].

- [13] jseidl, "GoldenEye," 15 10 2014. [Online]. Available: <https://github.com/jseidl/GoldenEye>. [Accessed 30 06 2025].
- [14] T. Nguyễn, "Random Forest algorithm," 07 07 2022. [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_model/random_forest.html. [Accessed 30 06 2025].
- [15] Wikipedia, "Random forest," 27 06 2025. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest. [Accessed 06 06 2025].
- [16] F. ISMAIL and M. MOHAMMED , "Security Information and Event Management:," 2023. [Online]. Available: <https://github.com/fathiismail/SIEM-with-Suricata-and-ELK-stack/blob/main/README.md>.