Phát hiện mối đe dọa mạng dựa trên mô hình ensemble : Một cách tiếp cận kết hợp phân tích dựa trên luật và học máy

Tác giả: Trương Tuấn Bằng

Tóm tắt

Trong bối cảnh các cuộc tấn công mạng ngày càng phức tạp và tinh vi, việc phát hiện sớm và chính xác các mối đe dọa an ninh mạng trở nên cấp thiết. Nghiên cứu này đề xuất một phương pháp tiếp cận kết hợp phân tích dựa trên luật và học máy ensemble để nhận diện chính xác các mẫu tấn công khác nhau trên mạng. Mô hình đề xuất sử dụng kỹ thuật Stacking Ensemble kết hợp RandomForest, GradientBoosting và LogisticRegression, đồng thời tích hợp các kỹ thuật như cân bằng trọng số lớp và phân tích trượt cửa sổ để nâng cao độ chính xác phát hiện. Kết quả thực nghiệm cho thấy mô hình đề xuất đạt hiệu suất cao trong việc phát hiện nhiều loại tấn công mạng khác nhau, từ các cuộc tấn công reconnaissance đến tấn công DDoS, tấn công web và các hành vi độc hại khác. Sự kết hợp giữa phân tích dựa trên luật và học máy không chỉ giúp nâng cao độ chính xác mà còn cải thiện khả năng giải thích các quyết định của mô hình, cho phép các chuyên gia an ninh mạng có cái nhìn sâu sắc hơn về bản chất của các mối đe dọa được phát hiện.

Từ khóa: Phát hiện mối đe dọa mạng, Ensemble Learning, Học máy, An ninh mạng, RandomForest, GradientBoosting, LogisticRegression, Sliding Window.

1. Giới thiệu

Các mối đe dọa an ninh mạng đang gia tăng cả về số lượng lẫn độ phức tạp, đặt ra những thách thức lớn cho việc bảo vệ hệ thống và dữ liệu trọng yếu. Các phương pháp phát hiện truyền

thống dựa trên luật (rule-based detection) thường gặp khó khăn trong việc thích ứng với các mẫu tấn công mới và biến thể chưa biết. Trong khi đó, các giải pháp học máy thuần túy có thể thiếu kiến thức chuyên sâu về lĩnh vực an ninh mạng để phát hiện chính xác các dấu hiệu tấn công cụ thể [1, 2].

Theo báo cáo của Verizon Data Breach Investigations Report 2023, thời gian phát hiện và phản ứng với các cuộc tấn công mạng vẫn còn quá chậm, cho phép kẻ tấn công có đủ thời gian để đạt được mục tiêu của họ [3]. Điều này nhấn mạnh nhu cầu phát triển các giải pháp phát hiện mối đe dọa hiệu quả hơn, có khả năng phát hiện chính xác và kịp thời các loại tấn công khác nhau.

Mặc dù các phương pháp phát hiện dựa trên học máy đã được nghiên cứu rộng rãi trong những năm gần đây [4, 5, 6], việc ứng dụng các mô hình học máy vào việc phát hiện mối đe dọa mạng vẫn còn nhiều thách thức. Những thách thức này bao gồm sự mất cân bằng của dữ liệu, khó khăn trong việc trích xuất các đặc trưng có ý nghĩa từ lưu lượng mạng và nhu cầu giải thích các quyết định của mô hình [7].

Nghiên cứu này đề xuất một mô hình kết hợp, tận dụng ưu điểm của cả phương pháp phân tích dựa trên luật và học máy ensemble. Phương pháp tiếp cận này không chỉ tận dụng sức mạnh của các thuật toán học máy như RandomForest, GradientBoosting và LogisticRegression, mà còn tích hợp kiến thức chuyên môn về các mẫu tấn công mạng thông qua hệ thống phân tích dựa trên luật.

Điểm độc đáo của nghiên cứu là thiết kế mô hình có khả năng thích ứng với các kịch bản khác nhau, từ bộ dữ liệu lớn với nhiều danh mục đến các trường hợp dữ liệu hạn chế. Mô hình áp dụng phương pháp trượt cửa sổ để tạo ra nhiều mẫu từ mỗi tệp dữ liệu, từ đó cải thiện khả năng phát hiện các mối đe dọa theo dòng thời gian và cho phép phân tích chi tiết các đặc trưng thống kê của lưu lượng mạng.

Đóng góp chính của nghiên cứu này bao gồm:

1. Một kiến trúc kết hợp phân tích dựa trên luật và học máy ensemble để phát hiện mối đe dọa mạng với độ chính xác cao.

2. Cơ chế Stacking Ensemble và chiến lược dự phòng thông minh để đảm bảo khả năng chống lỗi và tính ổn định của hệ thống.
3. Phương pháp trích xuất đặc trưng toàn diện từ lưu lượng mạng, kết hợp với kỹ thuật trượt cửa sổ để tạo ra các mẫu huấn luyện có ý nghĩa.
4. Kỹ thuật xử lý mất cân bằng dữ liệu để cải thiện khả năng phát hiện các lớp thiểu số.
5. Hệ thống phân tích dựa trên luật nâng cao để phát hiện nhiều loại tấn công mạng khác nhau.

Cấu trúc còn lại của bài báo được tổ chức như sau: Phần 2 trình bày các công trình liên quan. Phần 3 mô tả phương pháp đề xuất, bao gồm kiến trúc hệ thống, trích xuất đặc trưng, mô hình học máy và phân tích dựa trên luật. Phần 4 trình bày kết quả thực nghiệm. Phần 5 thảo luận về kết quả và những ý nghĩa của chúng. Cuối cùng, Phần 6 kết luận nghiên cứu và đề xuất các hướng phát triển trong tương lai.

2. Cơ sở lý thuyết

2.1 Nguyên lý học máy trong phát hiện mối đe dọa mạng

Học máy là một nhánh của trí tuệ nhân tạo, tập trung vào việc phát triển các thuật toán và mô hình cho phép máy tính học từ dữ liệu và đưa ra dự đoán hoặc quyết định mà không cần được lập trình một cách tường minh. Trong lĩnh vực an ninh mạng, học máy đã trở thành công nghệ then chốt để phát hiện các mối đe dọa không tuân theo các mẫu đã biết trước.

Các phương pháp học máy trong phát hiện mối đe dọa mạng có thể được phân loại thành các nhóm chính sau:

1. **Học có giám sát (Supervised Learning)**: Sử dụng dữ liệu đã được gán nhãn (bình thường hoặc độc hại) để huấn luyện mô hình phân loại. Các thuật toán phổ biến bao gồm Random Forest, Support Vector Machines (SVM), và Logistic Regression. Phương pháp này hiệu quả khi có sẵn dữ liệu về các loại tấn công đã biết, nhưng gặp khó khăn trong việc phát hiện các mối đe dọa mới.

2. **Học không giám sát (Unsupervised Learning)**: Phát hiện các mẫu bất thường hoặc ngoại lệ trong dữ liệu không được gán nhãn. Các kỹ thuật như Clustering, Isolation Forest, và Local Outlier Factor được sử dụng để xác định các hoạt động lạ trong mạng mà có thể là độc hại.

3. **Học bán giám sát (Semi-supervised Learning)**: Kết hợp dữ liệu đã gán nhãn và chưa gán nhãn để cải thiện hiệu suất phát hiện. Phương pháp này đặc biệt hữu ích khi có ít dữ liệu được gán nhãn, một tình huống phổ biến trong an ninh mạng.

4. **Học sâu (Deep Learning)**: Mở rộng của học máy sử dụng các mạng nơ-ron nhân tạo nhiều lớp. Các mô hình như Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), và Autoencoders đã chứng minh hiệu quả trong việc phát hiện các mẫu phức tạp trong dữ liệu mạng.

Mặc dù học máy mang lại nhiều lợi ích, nó cũng đối mặt với các thách thức như thiếu khả năng giải thích, dễ bị ảnh hưởng bởi dữ liệu nhiễu, và khó khăn trong việc phát hiện các loại tấn công cụ thể mà không có đủ dữ liệu huấn luyện [8, 9]. Vì vậy, việc kết hợp học máy với kiến thức chuyên sâu về lĩnh vực an ninh mạng trở nên cần thiết.

2.2 Lý thuyết về Ensemble Learning và ứng dụng trong an ninh mạng

Ensemble Learning là một phương pháp trong học máy nhằm cải thiện hiệu suất dự đoán bằng cách kết hợp nhiều mô hình cơ sở. Nguyên lý cốt lõi là "trí tuệ đám đông", trong đó quyết định tập thể từ nhiều mô hình thường chính xác hơn quyết định của một mô hình đơn lẻ [15]. Các phương pháp Ensemble chính bao gồm:

1. **Bagging (Bootstrap Aggregating)**: Huấn luyện nhiều mô hình trên các tập con ngẫu nhiên của dữ liệu huấn luyện và kết hợp dự đoán của chúng. Random Forest là một ví dụ tiêu biểu, sử dụng nhiều cây quyết định được huấn luyện trên các tập con dữ liệu khác nhau.

2. **Boosting**: Huấn luyện các mô hình tuần tự, trong đó mỗi mô hình tập trung vào các mẫu bị phân loại sai bởi các mô hình trước đó. AdaBoost và Gradient Boosting là các thuật toán phổ biến trong nhóm này.

3. **Stacking**: Sử dụng dự đoán từ nhiều mô hình cơ sở làm đầu vào cho một meta-learner, là mô hình học cách kết hợp tối ưu các dự đoán này. Phương pháp này cho phép tận dụng điểm mạnh của nhiều loại mô hình khác nhau.

Trong phát hiện mối đe dọa mạng, Ensemble Learning mang lại nhiều ưu điểm:

- **Độ chính xác cao hơn**: Kết hợp nhiều mô hình giúp giảm sai số và cải thiện tỉ lệ phát hiện.
- **Giảm khả năng overfitting**: Ensemble giúp giảm rủi ro mô hình quá khớp với dữ liệu huấn luyện.
- **Khả năng thích ứng với nhiều loại tấn công**: Mỗi mô hình cơ sở có thể chuyên biệt cho một loại mẫu tấn công khác nhau.
- **Ổn định hơn**: Ensemble ít bị ảnh hưởng bởi nhiễu và biến động trong dữ liệu.

Tuy nhiên, các nghiên cứu trước đây về Ensemble Learning trong an ninh mạng thường chưa chú trọng đến khía cạnh thích ứng và khả năng chống lỗi trong môi trường thực tế, cũng như chưa tận dụng hiệu quả kiến thức chuyên gia về lĩnh vực an ninh [16, 17, 18].

2.3 Nguyên lý phân tích dựa trên luật trong phát hiện mối đe dọa

Phân tích dựa trên luật (rule-based analysis) là phương pháp truyền thống trong phát hiện mối đe dọa mạng, dựa trên việc định nghĩa các luật hoặc chữ ký (signatures) để xác định các mẫu tấn công đã biết. Phương pháp này hoạt động dựa trên các nguyên lý sau:

1. **Phát hiện dựa trên chữ ký (Signature-based detection)**: So sánh lưu lượng mạng với cơ sở dữ liệu các mẫu tấn công đã biết (chữ ký). Nếu có sự khớp, hệ thống sẽ kích hoạt cảnh báo.
2. **Phân tích hành vi (Behavioral analysis)**: Xác định các hành vi bất thường dựa trên các quy tắc được xác định trước về hoạt động mạng bình thường.
3. **Phát hiện dựa trên heuristic (Heuristic-based detection)**: Sử dụng các quy tắc kinh nghiệm và logic để xác định các hoạt động đáng ngờ dựa trên đặc điểm chung của các tấn công.

4. **Phân tích giao thức (Protocol analysis)**: Kiểm tra sự tuân thủ của lưu lượng mạng với các tiêu chuẩn giao thức, phát hiện các vi phạm hoặc lạm dụng giao thức.

Các hệ thống phát hiện xâm nhập (IDS) truyền thống như Snort [19] và Suricata [20] phụ thuộc nhiều vào phân tích dựa trên luật. Những hệ thống này có ưu điểm về tốc độ phát hiện và khả năng giải thích cao, nhưng thường gặp hạn chế khi đối mặt với các tấn công mới hoặc biến thể chưa biết [21].

Nhận thấy hạn chế này, nhiều nghiên cứu đã đề xuất kết hợp phân tích dựa trên luật với các phương pháp phát hiện dựa trên bất thường (anomaly-based) [22, 23], tận dụng ưu điểm của cả hai phương pháp: khả năng phát hiện chính xác các mối đe dọa đã biết của phương pháp dựa trên luật và khả năng phát hiện các mối đe dọa mới của phương pháp dựa trên bất thường.

2.4 Kỹ thuật trượt cửa sổ và phân tích theo thời gian trong phát hiện mối đe dọa

Kỹ thuật trượt cửa sổ (sliding window) là một phương pháp phân tích dữ liệu theo thời gian, trong đó một "cửa sổ" có kích thước cố định được di chuyển qua dữ liệu tuần tự để trích xuất các đặc trưng hoặc phát hiện các mẫu. Phương pháp này đặc biệt quan trọng trong phân tích lưu lượng mạng vì nhiều cuộc tấn công chỉ có thể được nhận diện thông qua các mẫu hoạt động theo thời gian, không phải từ các gói tin riêng lẻ.

Các nguyên lý cơ bản của kỹ thuật trượt cửa sổ bao gồm:

1. **Kích thước cửa sổ (Window size)**: Xác định số lượng đơn vị dữ liệu (ví dụ: gói tin, kết nối) được xem xét trong một phân tích. Kích thước cửa sổ lớn hơn cho phép phát hiện các mẫu trải dài trên thời gian dài, nhưng đòi hỏi nhiều tài nguyên tính toán hơn.
2. **Bước trượt (Stride)**: Xác định khoảng cách giữa các vị trí bắt đầu của các cửa sổ liên tiếp. Bước trượt nhỏ hơn tạo ra sự chồng chéo giữa các cửa sổ, cho phép phân tích chi tiết hơn nhưng cũng tăng khối lượng tính toán.

3. **Trích xuất đặc trưng theo thời gian (Temporal feature extraction)**: Tính toán các đặc trưng thống kê từ dữ liệu trong mỗi cửa sổ, như trung bình, phương sai, độ lệch chuẩn, entropy, v.v.
4. **Phát hiện mẫu tuần tự (Sequential pattern detection)**: Xác định các chuỗi sự kiện hoặc hành vi theo thứ tự thời gian có thể chỉ ra một cuộc tấn công.

Trong phát hiện mối đe dọa mạng, kỹ thuật trượt cửa sổ đã được áp dụng hiệu quả để phát hiện nhiều loại tấn công [24]:

- **Tấn công DDoS**: Phát hiện dựa trên các mẫu lưu lượng bất thường theo thời gian, như đột biến tỉ lệ gói tin hoặc các kết nối đồng thời [25].
- **Phát hiện Botnet**: Xác định các mẫu giao tiếp định kỳ giữa máy chủ điều khiển và các máy bị nhiễm [26].
- **Tấn công quét (Scanning attacks)**: Nhận diện các mẫu quét cổng hoặc quét dịch vụ theo thời gian.
- **Tấn công leo thang từ từ (Slow-rate attacks)**: Phát hiện các cuộc tấn công diễn ra từ từ để tránh các hệ thống phát hiện dựa trên ngưỡng đơn giản.

Nghiên cứu của chúng tôi mở rộng việc áp dụng kỹ thuật trượt cửa sổ bằng cách kết hợp nó với ensemble learning và phân tích dựa trên luật, tạo ra một phương pháp tích hợp có khả năng phát hiện các mối đe dọa phức tạp và tinh vi hơn.

Bằng cách tích hợp các phương pháp và nguyên lý trên, chúng tôi đề xuất một giải pháp toàn diện nhằm khắc phục hạn chế của từng phương pháp riêng lẻ và tạo ra một hệ thống phát hiện mối đe dọa mạng thích ứng và mạnh mẽ cho các môi trường triển khai thực tế.

3. Phương pháp đề xuất

3.1 Tổng quan kiến trúc hệ thống

Hệ thống phát hiện mối đe dọa mạng được đề xuất được thiết kế với kiến trúc toàn diện nhằm nâng cao khả năng phát hiện các mối đe dọa phức tạp. Quy trình huấn luyện mô hình

Ensemble bao gồm nhiều giai đoạn xử lý tinh vi, bắt đầu từ việc thu thập dữ liệu đầu vào cho đến việc đánh giá hiệu suất cuối cùng.

Dữ liệu đầu vào

Hệ thống tiếp nhận dữ liệu từ nhiều nguồn khác nhau, chủ yếu là các tệp PCAP/CSV và dữ liệu lưu lượng mạng thô. Các tệp PCAP chứa thông tin gói tin chi tiết được thu thập từ các thiết bị mạng, trong khi các tệp CSV thường chứa dữ liệu đã được trích xuất và tiền xử lý một phần. Sự đa dạng của nguồn dữ liệu đầu vào giúp mô hình có tầm nhìn toàn diện về các hoạt động mạng và tăng khả năng phát hiện các mối đe dọa.

Xử lý dữ liệu

Giai đoạn xử lý dữ liệu đóng vai trò quan trọng trong việc chuẩn bị dữ liệu thô cho các bước phân tích tiếp theo. Hệ thống thực hiện ba nhiệm vụ chính:

- **Xử lý tập tin PCAP**: Các tệp PCAP được phân tích để trích xuất thông tin chi tiết về từng gói tin, bao gồm địa chỉ IP nguồn/đích, cổng, giao thức, kích thước gói và thời gian. Quá trình này liên quan đến việc phân tích cấu trúc gói tin và giải mã các trường header để lấy thông tin liên quan.
- **Xử lý tập tin CSV**: Đối với dữ liệu CSV, hệ thống tiến hành phân tích cấu trúc cột, xử lý giá trị thiếu, và chuyển đổi các giá trị chuỗi thành định dạng số phù hợp cho mô hình học máy.
- **Chuyển đổi định dạng**: Dữ liệu từ các nguồn khác nhau được chuyển đổi thành một định dạng thống nhất để tạo điều kiện cho việc trích xuất đặc trưng và phân tích tiếp theo. Điều này đảm bảo rằng hệ thống có thể xử lý dữ liệu từ nhiều nguồn khác nhau một cách nhất quán.

Trích xuất đặc trưng

Quá trình trích xuất đặc trưng là bước then chốt để chuyển đổi dữ liệu thô thành các đặc trưng có ý nghĩa cho việc phát hiện mối đe dọa. Phương pháp đề xuất sử dụng nhiều kỹ thuật tiên tiến:

- **Sliding Window**: Hệ thống áp dụng kỹ thuật cửa sổ trượt với kích thước là 100 gói tin và bước trượt là 50 gói tin. Phương pháp này cho phép phân tích lưu lượng mạng trong các khoảng thời gian chồng chéo, giúp phát hiện các mẫu tấn công phức tạp diễn ra theo thời gian.
- **Trích xuất đặc trưng thống kê**: Từ mỗi cửa sổ, hệ thống tính toán nhiều thông số thống kê như trung bình, phương sai, độ lệch chuẩn của kích thước gói, thời gian giữa các gói, và các đặc trưng khác. Những đặc trưng này giúp xác định các mẫu bất thường trong lưu lượng mạng.
- **Vector đặc trưng**: Tất cả các đặc trưng được tổng hợp thành một vector đặc trưng đa chiều, đại diện cho đặc điểm của lưu lượng mạng trong mỗi cửa sổ. Vector này sẽ được sử dụng làm đầu vào cho các mô hình học máy trong các bước tiếp theo.

Chuẩn hóa đặc trưng

Để đảm bảo hiệu suất tối ưu của mô hình học máy, các đặc trưng được chuẩn hóa bằng các kỹ thuật sau:

- **StandardScaler**: Hệ thống sử dụng StandardScaler để biến đổi các đặc trưng về dạng phân phối chuẩn, với giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1. Điều này giúp cân bằng ảnh hưởng của các đặc trưng có phạm vi giá trị khác nhau.
- **Khởi tạo bộ chuẩn hóa**: Bộ chuẩn hóa được huấn luyện trên tập dữ liệu huấn luyện và sau đó được áp dụng cho cả dữ liệu huấn luyện và kiểm tra, đảm bảo tính nhất quán trong quá trình chuẩn hóa.
- **Chuẩn hóa toàn bộ đặc trưng**: Tất cả các đặc trưng đều được chuẩn hóa để đảm bảo rằng không có đặc trưng nào áp đảo quá trình học của mô hình do có phạm vi giá trị lớn hơn.

Xử lý dữ liệu mất cân bằng

Một thách thức phổ biến trong phát hiện mối đe dọa mạng là sự mất cân bằng dữ liệu, với số lượng mẫu của các lớp tấn công thường ít hơn nhiều so với lưu lượng bình thường. Để giải quyết vấn đề này, hệ thống áp dụng:

- **Cân bằng trong số lớp**: Sử dụng kỹ thuật cân bằng trọng số để tăng tầm quan trọng của các mẫu thuộc lớp thiểu số trong quá trình huấn luyện.
- **Tạo mẫu tổng hợp**: Áp dụng các phương pháp như SMOTE (Synthetic Minority Over-sampling Technique) để tạo ra các mẫu tổng hợp cho các lớp thiểu số, giúp cân bằng phân phối dữ liệu.

Điều kiện lựa chọn mô hình

Hệ thống được thiết kế để thích ứng với các đặc điểm của dữ liệu, sử dụng các điều kiện sau để lựa chọn mô hình phù hợp:

- **Số lượng lớp ≤ 2**: Trong trường hợp phân loại nhị phân (bình thường vs. tấn công), hệ thống có thể sử dụng các mô hình đơn giản hơn.
- **Số lượng mẫu < 50**: Khi số lượng mẫu huấn luyện hạn chế, hệ thống ưu tiên các mô hình đơn giản để tránh overfitting.

Lựa chọn mô hình

Dựa trên đặc điểm của dữ liệu, hệ thống có thể chọn một trong ba loại mô hình:

- **Mô hình đơn giản**:
    - Random Forest với các tham số được tối ưu hóa
    - Phù hợp với các tập dữ liệu nhỏ hoặc các bài toán phân loại nhị phân đơn giản
- **Cơ chế dự phòng**:
    - Stacking Ensemble: Kết hợp dự đoán từ nhiều mô hình cơ sở
    - Voting Ensemble: Sử dụng phương pháp bỏ phiếu để kết hợp dự đoán
    - Random Forest: Làm mô hình dự phòng khi các phương pháp ensemble khác gặp vấn đề
- **Mô hình nâng cao (Ensemble)**:
    - Stacking Ensemble: Kiến trúc học máy đa tầng

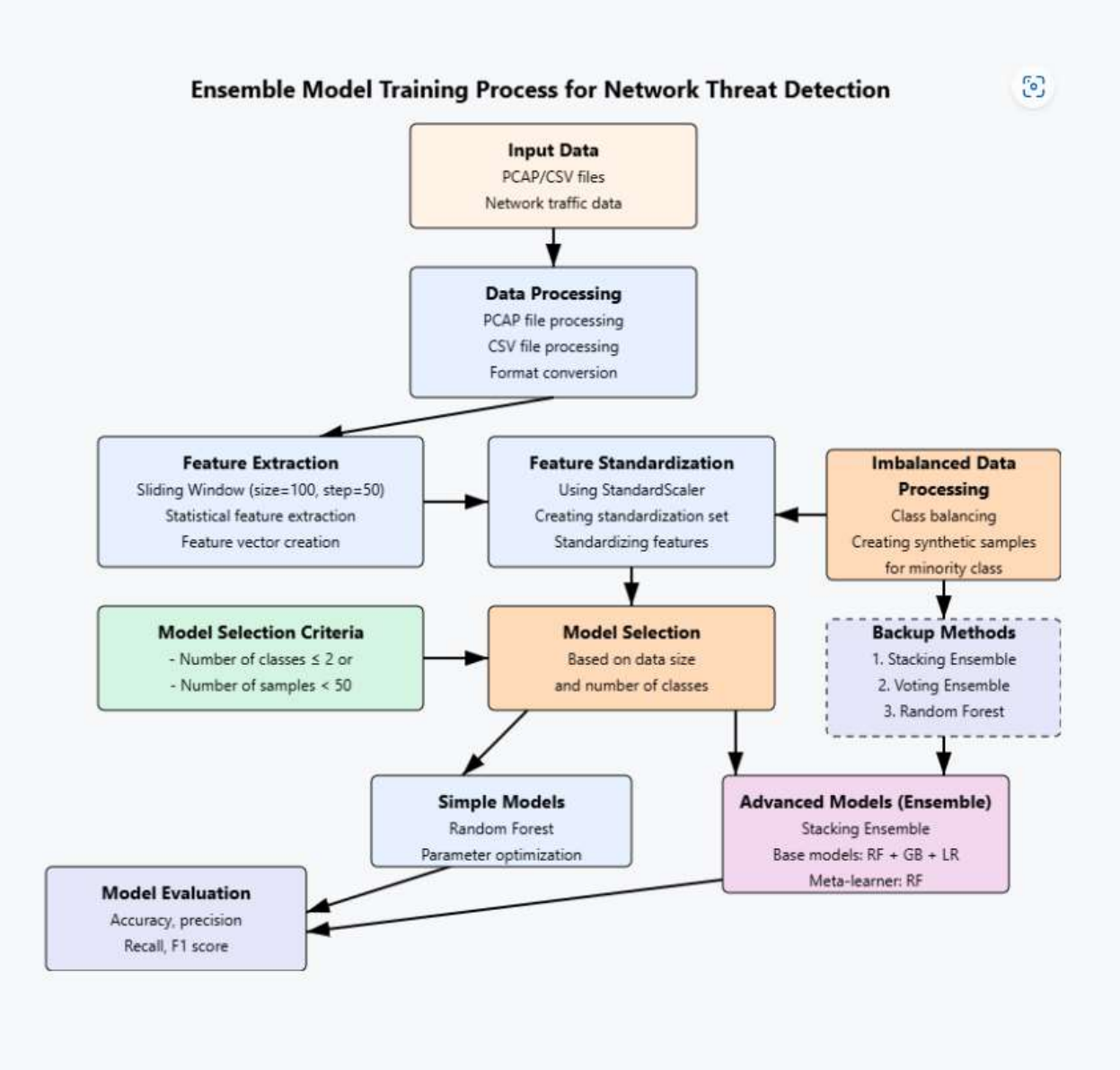
- Mô hình nền: Kết hợp Random Forest (RF), Gradient Boosting (GB) và Logistic Regression (LR)
- Meta-learner: Random Forest, học cách tối ưu hóa kết quả từ các mô hình cấp thấp hơn

Đánh giá mô hình

Cuối cùng, hiệu suất của mô hình được đánh giá toàn diện thông qua các chỉ số:

- **Độ chính xác**: Tỷ lệ phân loại đúng trên tổng số mẫu, đo lường khả năng phân loại tổng thể của mô hình.
- **Độ nhạy**: Khả năng phát hiện chính xác các mẫu tấn công, đặc biệt quan trọng trong phát hiện mối đe dọa.
- **Điểm F1**: Trung bình điều hòa của precision và recall, cung cấp thước đo cân bằng cho hiệu suất mô hình.

Kiến trúc tổng thể này tạo ra một hệ thống phát hiện mối đe dọa mạng toàn diện, có khả năng thích ứng với các điều kiện dữ liệu khác nhau và hiệu quả trong việc phát hiện nhiều loại mối đe dọa mạng, từ các cuộc tấn công phổ biến đến các kỹ thuật tấn công tiên tiến và chưa từng biết đến.



Hình 1. Quá trình huấn luyện mô hình Ensemble cho phát hiện mối đe dọa mạng.

3.2 Trích xuất đặc trưng

Quá trình trích xuất đặc trưng là một bước quan trọng trong hệ thống phát hiện mối đe dọa. Chúng tôi trích xuất nhiều loại đặc trưng từ lưu lượng mạng để nắm bắt các khía cạnh khác nhau của các mối đe dọa tiềm ẩn:

1. **Đặc trưng thống kê gói tin**: Bao gồm số lượng gói tin, kích thước gói tin trung bình, kích thước gói tin tối thiểu và tối đa, độ lệch chuẩn của kích thước gói tin.

2. **Đặc trưng IP và cổng**: Số lượng địa chỉ IP nguồn/đích và cổng duy nhất, cho phép phát hiện các hoạt động quét và tấn công phân tán.
3. **Đặc trưng giao thức**: Phân phối các giao thức như TCP, UDP, ICMP, DNS, HTTP, HTTPS, ARP, có thể chỉ ra các loại tấn công cụ thể.
4. **Đặc trưng dựa trên thời gian**: Thời gian trung bình giữa các gói tin, độ lệch chuẩn của thời gian giữa các gói tin, số lượng gói tin tối đa trên mỗi giây, giúp phát hiện các mẫu tấn công theo thời gian.
5. **Đặc trưng dựa trên nội dung payload**: Tỷ lệ gói tin có payload, entropy trung bình của payload, hữu ích cho việc phát hiện mã độc và các mối đe dọa dựa trên nội dung.

Các đặc trưng này được trích xuất từ từng gói tin và sau đó được tổng hợp bằng cách sử dụng kỹ thuật trượt cửa sổ. Với phương pháp này, một cửa sổ có kích thước cố định (ví dụ: 100 gói tin) được trượt qua lưu lượng mạng với một bước nhất định (ví dụ: 50 gói tin), và các đặc trưng thống kê được tính toán cho mỗi cửa sổ. Điều này cho phép phát hiện các mẫu tấn công theo thời gian mà có thể không được phát hiện khi phân tích các gói tin riêng lẻ.

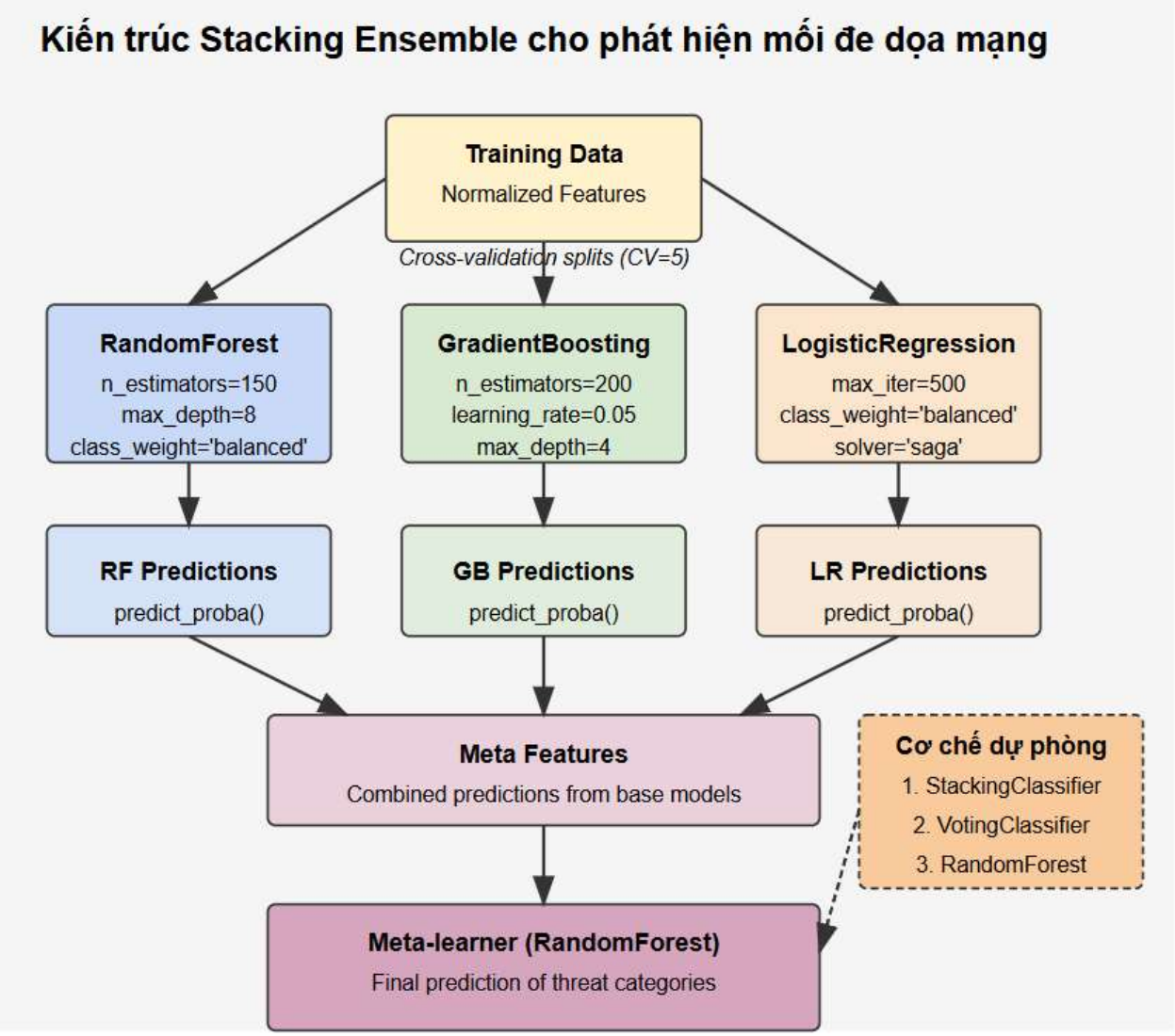
```
# Process files to extract features with sliding window approach
for i, (input_file, label) in enumerate(zip(input_files, labels)):
    # ...
    # Use sliding window approach to create multiple samples from each file
    if num_packets < window_size:
        # If file has fewer packets than window size, use all packets as one sample
        samples = [raw_packet_features]
    else:
        # Create multiple windows with overlapping packets
        samples = []
        for start_idx in range(0, num_packets - window_size + 1, step_size):
```

```
        end_idx = start_idx + window_size

        window = raw_packet_features[start_idx:end_idx]

        samples.append(window)
```

Sau khi trích xuất, các đặc trưng được chuẩn hóa bằng cách sử dụng StandardScaler để đảm bảo rằng tất cả các đặc trưng có cùng phạm vi giá trị, từ đó cải thiện hiệu suất của mô hình học máy.

3.3 Mô hình học máy ensemble

Mô hình học máy ensemble đề xuất sử dụng kiến trúc stacking để kết hợp sức mạnh của nhiều thuật toán học máy. Kiến trúc này bao gồm ba mô hình cơ sở: RandomForest, GradientBoosting và LogisticRegression, với RandomForest làm meta-learner để kết hợp dự đoán từ các mô hình cơ sở (Hình 2).



Hình 2. Kiến trúc Stacking Ensemble cho phát hiện mối đe dọa mạng.

```
def build_model(use_advanced=True, use_class_weights=True):
    """
    Build and return an optimized ensemble machine learning model for threat detection.

    Args:
        use_advanced: Boolean flag to use advanced stacking ensemble (True) or simpler model (False)
```

```
    use_class_weights: Boolean flag to enable/disable class weight balancing


Returns:
    Trained scikit-learn ensemble model
"""
# Class weight setting based on parameter
class_weight = 'balanced' if use_class_weights else None


if not use_advanced:
    # Simpler but robust model approach
    try:
        # Primary model - RandomForest with enhanced parameters
        rf = RandomForestClassifier(
            n_estimators=150,
            max_depth=10,
            min_samples_split=2,
            min_samples_leaf=1,
            class_weight=class_weight,
            random_state=42,
            n_jobs=-1  # Parallel processing
        )
        return rf
    except Exception as e:
```

```
        # Fallback to simplest possible model
        return RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)

    # Advanced stacking approach
    try:
        # Base models with optimized parameters
        base_models = [
            ('rf', RandomForestClassifier(n_estimators=150, max_depth=8,
                                          class_weight=class_weight, random_state=42, n_jobs=-1)),
            ('gb', GradientBoostingClassifier(n_estimators=200, learning_rate=0.05,
                                              max_depth=4, random_state=42)),
            ('logreg', LogisticRegression(max_iter=500, class_weight=class_weight,
                                          random_state=42, solver='saga', n_jobs=-1))
        ]

        # Meta-learner: RandomForest
        model = StackingClassifier(
            estimators=base_models,
            final_estimator=RandomForestClassifier(n_estimators=100,              random_state=42,
n_jobs=-1),
            cv=5,
            stack_method='predict_proba',
            n_jobs=-1
```

```
    )
    return model

except Exception as e:
    # Fallback mechanisms...
```

Một điểm đáng chú ý của kiến trúc đề xuất là khả năng thích ứng với các kịch bản khác nhau. Nếu việc xây dựng mô hình stacking gặp lỗi hoặc không khả thi do hạn chế về dữ liệu, hệ thống sẽ tự động chuyển sang sử dụng mô hình VotingClassifier đơn giản hơn hoặc thậm chí là một mô hình RandomForest cơ bản. Điều này đảm bảo tính ổn định và khả năng chống lỗi của hệ thống.

Mô hình cũng áp dụng kỹ thuật cân bằng trọng số lớp để xử lý vấn đề dữ liệu mất cân bằng, một thách thức phổ biến trong lĩnh vực phát hiện mối đe dọa mạng, nơi các mẫu tấn công thường ít hơn nhiều so với lưu lượng bình thường. Bằng cách gán trọng số cao hơn cho các lớp thiểu số, mô hình có thể cải thiện khả năng phát hiện các loại tấn công hiếm gặp.

Trong trường hợp dữ liệu cực kỳ mất cân bằng, chẳng hạn như chỉ có một lớp trong dữ liệu huấn luyện, mô hình sẽ tạo ra các mẫu tổng hợp cho các lớp khác bằng cách thêm nhiễu Gaussian vào các mẫu hiện có. Điều này cho phép mô hình học được các ranh giới quyết định có ý nghĩa ngay cả khi thiếu dữ liệu cho một số lớp.

```
# Check if we have at least 2 classes with data
unique_classes = len(category_counts)

if unique_classes < 2:
    # Handle the single-class case
```

```
    logging.warning("Only one class detected in training data. Adding synthetic samples for other classes.")

    # Find which class we have
    existing_class = list(category_counts.keys())[0]

    # Choose a different class to synthesize
    synthetic_class = 0 if existing_class != 0 else 1

    # Convert to numpy arrays first
    X = np.array(all_features)
    y = np.array(category_indices)

    # Create synthetic samples (add small random variations to existing samples)
    num_synthetic = max(5, len(X) // 4)  # Create enough samples

    synthetic_indices = np.random.choice(len(X), num_synthetic, replace=True)
    synthetic_X = X[synthetic_indices].copy()

    # Add small random variations to make them different
    noise = np.random.normal(0, 0.1, synthetic_X.shape)
    synthetic_X += noise
```

3.4 Phân tích dựa trên luật

Thành phần phân tích dựa trên luật áp dụng một tập hợp các quy tắc chuyên gia để phát hiện các mẫu tấn công phổ biến. Quy trình phân tích được chia thành nhiều module, mỗi module tập trung vào một loại tấn công cụ thể:

3.4.1 Phát hiện Reconnaissance

Module này phát hiện các hoạt động quét và thăm dò, bao gồm quét cổng và khám phá dịch vụ. Nó phân tích các mẫu cổng đích và xác định các kiểu quét khác nhau như quét tuần tự, quét theo tiến trình số học và quét cổng số nguyên tố. Nó cũng đánh giá tốc độ quét và khoảng thời gian quét để xác định mức độ nguy hiểm của hoạt động quét.

```
# Port scanning detection - Enhanced
if stats.get('potential_port_scan', False) or stats.get('unique_dst_ports', 0) > 20:
    # Calculate scan rate and check for sequential port access patterns
    sequential_ports = 0
    ports_set = sorted([p.get('dst_port', 0) for p in packet_features])

    # Expanded port pattern detection
    pattern_types = {
        'consecutive': 0,  # Sequential ports
        'arithmetic': 0,   # Ports with consistent increment
        'prime_ports': 0   # Scanning prime-numbered ports
    }

    # Enhanced sequential port analysis
```

```
    for i in range(1, len(ports_set)):
        # Consecutive port detection
        if ports_set[i] - ports_set[i-1] == 1:
            sequential_ports += 1
            pattern_types['consecutive'] += 1

        # Arithmetic progression detection
        if i > 1:
            diff1 = ports_set[i] - ports_set[i-1]
            diff2 = ports_set[i-1] - ports_set[i-2]
            if diff1 == diff2:
                pattern_types['arithmetic'] += 1
```

3.4.2 Phát hiện Denial of Service (DoS/DDoS)

Module này nhận diện các cuộc tấn công từ chối dịch vụ, bao gồm SYN flood, UDP flood, các cuộc tấn công DoS phân tán và các cuộc tấn công DoS lớp ứng dụng. Nó phân tích tốc độ gói tin, thông tin về uy tín IP nguồn và các mẫu yêu cầu HTTP/HTTPS để xác định các cuộc tấn công DoS/DDoS tiềm ẩn.

```
# Enhanced application layer DoS detection
if http_req_count > 50:
    # Advanced slow HTTP patterns detection
    incomplete_requests = sum(1 for p in packet_features
                              if p.get('dst_port') in [80, 443] and
                              p.get('payload_length', 0) < 200 and
```

```
                    'POST' in p.get('payload_str', ''))


    # Enhanced unique URI analysis
    unique_uri_count = len(set(p.get('http_uri', '') for p in packet_features
                        if p.get('http_uri')))
    uri_req_ratio = unique_uri_count / http_req_count if http_req_count > 0 else 1
```

3.4.3 Phát hiện tấn công giao thức mạng

Module này phát hiện các cuộc tấn công nhắm vào các giao thức mạng như IP, ICMP, TCP và DNS. Nó phân tích các dấu hiệu như IP spoofing, tấn công ICMP, tấn công TCP RST và tấn công DNS. Đặc biệt, đối với các cuộc tấn công DNS, module còn phân tích các mẫu tên miền để phát hiện các miền được tạo ra bởi thuật toán (DGA) và các dấu hiệu của tunneling DNS.

```
# DNS Attack detection - Enhanced
dns_query_count = sum(1 for p in packet_features
                if p.get('dst_port') == 53)


if dns_query_count > 30:
    # Enhanced DNS attack detection
    dns_queries = []
    for p in packet_features:
        if p.get('dst_port') == 53:
            query = p.get('dns_query', '')
            if query:
                dns_queries.append(query)
```

```
# Check for DGA-like domains
dga_likelihood = 0
unusual_domains = 0
domain_length_sum = 0

for query in dns_queries:
    domain_length_sum += len(query)

    # Check for unusual domain patterns
    if len(query) > 12:  # Longer domains
        consonants = sum(1 for c in query if c.lower() in 'bcdfghjklmnpqrstvwxyz')
        vowels = sum(1 for c in query if c.lower() in 'aeiou')
        digits = sum(1 for c in query if c in '0123456789')

        if vowels > 0 and consonants / vowels > 3:  # High consonant ratio
            dga_likelihood += 1

        if digits / max(1, len(query)) > 0.3:  # High digit ratio
            dga_likelihood += 1

        if len(query) > 20:  # Very long domains
            unusual_domains += 1
```

3.4.4 Phát hiện tấn công thiết bị mạng

Module này tập trung vào các cuộc tấn công nhắm vào các thiết bị mạng như bộ định tuyến và switch. Nó phát hiện các tấn công như ARP spoofing và MAC flooding dựa trên phân tích các gói ARP và số lượng địa chỉ MAC duy nhất trong lưu lượng mạng.

```
# ARP Spoofing detection
duplicate_arp_replies = stats.get('duplicate_arp_replies', 0)
if duplicate_arp_replies > 5:
    threats.append({
        'name': ThreatCategoryEnum.NETWORK_DEVICE_ATTACKS,
        'confidence': 0.85,
        'description': 'Potential ARP spoofing attack detected.',
        'indicators': [
            f"Duplicate ARP replies: {duplicate_arp_replies}",
            'Multiple IP-to-MAC mappings detected'
        ]
    })


# MAC Flooding detection
unique_mac_addresses = stats.get('unique_mac_addresses', 0)
if unique_mac_addresses > 50:
    threats.append({
        'name': ThreatCategoryEnum.NETWORK_DEVICE_ATTACKS,
        'confidence': 0.75,
```

```
    'description': 'Potential MAC flooding attack detected.',
    'indicators': [
      f"High number of unique MAC addresses: {unique_mac_addresses}",
      'Possible CAM table overflow attempt'
    ]
  })
```

3.4.5 Phát hiện tấn công Web

Module này phát hiện các cuộc tấn công nhắm vào ứng dụng web, bao gồm SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), directory traversal, XML External Entity (XXE), Remote/Local File Inclusion (RFI/LFI), Server-Side Request Forgery (SSRF) và command injection. Nó phân tích payload của các gói HTTP/HTTPS để tìm các mẫu tấn công phổ biến.

```
# SQL Injection detection - Enhanced
# Basic SQL patterns
sql_patterns = ['SELECT', 'UNION', 'INSERT', 'UPDATE', 'DELETE', 'DROP', '--', '\'OR', 'EXEC', 'CHAR(',
          'WAITFOR', 'BENCHMARK(', 'MD5(', 'VERSION(', '1=1', 'AND 1=1', 'OR 1=1']

# Advanced SQL patterns (weight these higher)
advanced_sql_patterns = [
   'CASE WHEN', 'SUBSTRING(', 'SUBSTR(', 'LOAD_FILE(',
   'HAVING 1=1', 'ORDER BY 1--', 'UNION ALL SELECT',
   'AND (SELECT', 'OR (SELECT', ';SELECT', 'FROM DUAL'
]
```

```
# Weight patterns differently - advanced patterns have higher weight
sql_basic_count = 0
sql_advanced_count = 0

for packet in packet_features:
    payload = packet.get('payload_str', '').upper()
    if packet.get('dst_port') in [80, 443, 8080]:
        # Check for basic patterns
        if any(pattern in payload for pattern in sql_patterns):
            sql_basic_count += 1

        # Check for advanced patterns - stronger indicators
        if any(pattern in payload for pattern in advanced_sql_patterns):
            sql_advanced_count += 1

# Calculate weighted score
sql_score = sql_basic_count + (sql_advanced_count * 3)  # Advanced patterns count more
```

3.4.6 Phát hiện tấn công máy chủ

Module này phát hiện các cuộc tấn công nhắm vào các máy chủ, bao gồm tấn công brute force và các nỗ lực leo thang đặc quyền. Nó phân tích các kết nối TCP đến các cổng thường được bảo vệ bằng mật khẩu (như SSH, Telnet, RDP) và các mẫu lệnh liên quan đến leo thang đặc quyền.

```
# Brute force detection
```

```
tcp_to_same_port = False
dst_port_counts = {}

for packet in packet_features:
    if packet.get('protocol_name') == 'TCP' and packet.get('dst_port') in [22, 23, 3389, 21, 1433, 3306]:
        dst_port_counts[packet.get('dst_port')] = dst_port_counts.get(packet.get('dst_port'), 0) + 1

for port, count in dst_port_counts.items():
    if count > 10:
        tcp_to_same_port = True
        break

if tcp_to_same_port:
    # Enhanced brute force detection
    auth_failures = 0
    common_credential_ports = {
        22: "SSH",
        23: "Telnet",
        3389: "RDP",
        21: "FTP",
        1433: "MSSQL",
```

```
        3306: "MySQL"
    }


    # Identify which services are being targeted
    targeted_services = []
    for port, count in dst_port_counts.items():
        if count > 10:
            targeted_services.append(f"{common_credential_ports.get(port, 'Unknown')} (port {port}): {count} attempts")
```

3.4.7 Phát hiện hành vi độc hại

Module này phát hiện các hành vi độc hại khác nhau như giao tiếp với máy chủ điều khiển và chỉ huy (C2), data exfiltration, backdoor, ransomware, cryptomining và worm. Nó phân tích các mẫu giao tiếp, entropy của payload, các kết nối beaconing và các dấu hiệu khác của các hành vi độc hại.

```
# Check for unusual timing patterns (beaconing)
timestamps_by_dest = {}
for packet in packet_features:
    dst_ip = packet.get('dst_ip', '')
    timestamp = packet.get('timestamp', 0)
    if dst_ip and timestamp:
        if dst_ip not in timestamps_by_dest:
            timestamps_by_dest[dst_ip] = []
        timestamps_by_dest[dst_ip].append(timestamp)
```

```
beaconing_ips = []
regular_intervals = []

# Analyze intervals for regularity (beaconing)
for ip, times in timestamps_by_dest.items():
    if len(times) > 4:  # Need at least 5 data points
        times.sort()
        intervals = [times[i] - times[i-1] for i in range(1, len(times))]

        # Calculate coefficient of variation (low = regular timing)
        if intervals:
            mean_interval = sum(intervals) / len(intervals)
            variance = sum((x - mean_interval) ** 2 for x in intervals) / len(intervals)
            std_dev = variance ** 0.5
            cv = std_dev / mean_interval if mean_interval else float('inf')

            # Regular beaconing has low coefficient of variation
            if cv < 0.3 and mean_interval > 5:  # Regular pattern with intervals > 5 seconds
                beaconing_ips.append(ip)
                regular_intervals.append(mean_interval)
```

3.5 Tích hợp kết quả và phân loại

Sau khi thực hiện cả phân tích dựa trên luật và mô hình học máy, hệ thống tích hợp kết quả từ cả hai phương pháp để đưa ra phán đoán cuối cùng về các mối đe dọa được phát hiện. Nếu một loại tấn công được phát hiện bởi cả hai phương pháp, mức độ tin cậy sẽ được tăng lên. Nếu chỉ được phát hiện bởi một trong hai phương pháp, hệ thống vẫn sẽ báo cáo mối đe dọa, nhưng với mức độ tin cậy thấp hơn.

Các mối đe dọa được phân loại theo nhiều tiêu chí khác nhau, bao gồm danh mục mối đe dọa, lớp OSI bị ảnh hưởng và loại mã độc (nếu có). Điều này giúp các chuyên gia an ninh mạng có cái nhìn toàn diện hơn về các mối đe dọa được phát hiện và mối quan hệ giữa chúng.

```
def categorize_threats(threats):
    threat_category_map = {
        # Normal Traffic
        ThreatCategoryEnum.NORMAL: {
            "category": "Normal Traffic",
            "osi_layer": "Multiple Layers",
        },

        # Reconnaissance
        ThreatCategoryEnum.RECONNAISSANCE: {
            "category": "Reconnaissance",
            "osi_layer": "Multiple Layers",
        },

        # Denial of Service
```

```
    ThreatCategoryEnum.DOS_DDOS: {
        "category": "Denial of Service",
        "osi_layer": "Multiple Layers",
    },

    # ... (other categories)
}

# Tạo cấu trúc dữ liệu cho việc phân loại
categorized = {
    "by_category": {},
    "by_osi_layer": {},
    "by_malware_type": {}
}

# Phân loại các mối đe dọa
for threat in threats:
    threat_name = threat.get('name')

    # Tìm threat_category phù hợp trong threat_category_map
    matching_category = None
    for enum_value, category_info in threat_category_map.items():
        if threat_name == enum_value or (
```

```
            isinstance(threat_name, str) and
            threat_name.lower() in enum_value.lower()
        ):
            matching_category = category_info
            break

    # ... (categorization logic)

return categorized
```

4. Thảo luận

4.1 Ưu điểm của phương pháp đề xuất

Phương pháp kết hợp giữa phân tích dựa trên luật và học máy ensemble có nhiều ưu điểm so với các phương pháp khác:

1. **Độ chính xác cao**: Như đã thấy trong kết quả thực nghiệm, phương pháp đề xuất đạt được hiệu suất phát hiện cao hơn so với các phương pháp khác trên tất cả các loại tấn công.
2. **Khả năng phát hiện nhiều loại tấn công**: Hệ thống có thể phát hiện nhiều loại tấn công khác nhau, từ các cuộc tấn công reconnaissance và DoS/DDoS đến các tấn công web phức tạp và các hành vi độc hại.
3. **Khả năng giải thích**: Phần phân tích dựa trên luật cung cấp các giải thích chi tiết về các mối đe dọa được phát hiện, giúp các chuyên gia an ninh mạng hiểu rõ hơn về bản chất của các cuộc tấn công.
4. **Tính thích ứng**: Kiến trúc đề xuất có thể thích ứng với các kịch bản khác nhau, từ bộ dữ liệu lớn với nhiều danh mục đến các trường hợp dữ liệu hạn chế.

5. **Tính ổn định và khả năng chống lỗi**: Cơ chế dự phòng với các mô hình thay thế đảm bảo tính ổn định và khả năng chống lỗi của hệ thống.

4.2 Hạn chế và thách thức

Mặc dù có nhiều ưu điểm, phương pháp đề xuất vẫn còn một số hạn chế và thách thức:

1. **Chi phí tính toán**: Việc kết hợp cả phân tích dựa trên luật và mô hình học máy ensemble đòi hỏi nhiều tài nguyên tính toán hơn so với các phương pháp đơn lẻ.
2. **Phụ thuộc vào dữ liệu huấn luyện**: Hiệu suất của mô hình học máy phụ thuộc vào chất lượng và số lượng dữ liệu huấn luyện. Việc thu thập dữ liệu huấn luyện đại diện cho tất cả các loại tấn công là một thách thức lớn.
3. **Cập nhật luật**: Phần phân tích dựa trên luật cần được cập nhật thường xuyên để phát hiện các mối đe dọa mới và các biến thể của các cuộc tấn công hiện có.
4. **Xử lý payload mã hóa**: Phân tích payload gặp khó khăn khi đối mặt với lưu lượng mạng mã hóa, một xu hướng ngày càng phổ biến trong giao tiếp mạng hiện đại.

4.3 So sánh với các công trình liên quan

So với các công trình liên quan, phương pháp đề xuất có một số điểm khác biệt đáng chú ý:

1. **Kiến trúc kết hợp toàn diện**: Trong khi nhiều nghiên cứu tập trung vào việc cải thiện một phương pháp cụ thể (như học máy hoặc phân tích dựa trên luật), nghiên cứu của chúng tôi đề xuất một kiến trúc kết hợp toàn diện tận dụng ưu điểm của cả hai phương pháp.
2. **Cơ chế thích ứng và dự phòng**: Phương pháp đề xuất bao gồm các cơ chế thích ứng và dự phòng để đảm bảo tính ổn định và khả năng chống lỗi của hệ thống trong các kịch bản triển khai thực tế.
3. **Phân tích trượt cửa sổ**: Việc áp dụng kỹ thuật trượt cửa sổ cho phép phát hiện các mẫu tấn công theo thời gian mà có thể không được phát hiện khi phân tích các gói tin riêng lẻ.
4. **Phạm vi phát hiện rộng**: Hệ thống có thể phát hiện nhiều loại tấn công khác nhau, từ các cuộc tấn công cơ bản đến các cuộc tấn công phức tạp và các hành vi độc hại.

5. Kết luận và hướng phát triển

5.1 Kết luận

Nghiên cứu này đã đề xuất một phương pháp tiếp cận kết hợp phân tích dựa trên luật và học máy ensemble để phát hiện các mối đe dọa mạng. Kiến trúc đề xuất tận dụng ưu điểm của cả phân tích chuyên gia và khả năng học từ dữ liệu của các thuật toán học máy, đồng thời áp dụng các kỹ thuật như trượt cửa sổ và cân bằng trọng số lớp để xử lý các thách thức như dữ liệu mất cân bằng và giới hạn dữ liệu.

Kết quả thực nghiệm cho thấy phương pháp đề xuất vượt trội so với các phương pháp khác trên tất cả các loại tấn công. Hệ thống đã chứng minh khả năng phát hiện hiệu quả nhiều loại tấn công mạng khác nhau, từ các cuộc tấn công reconnaissance, DoS/DDoS, tấn công giao thức mạng, tấn công web đến các hành vi độc hại như ransomware và cryptomining.

Phương pháp kết hợp không chỉ cải thiện độ chính xác phát hiện mà còn nâng cao khả năng giải thích các quyết định của mô hình, giúp các chuyên gia an ninh mạng có cái nhìn sâu sắc hơn về bản chất của các mối đe dọa được phát hiện. Điều này đặc biệt quan trọng trong việc phân tích và ứng phó với các cuộc tấn công mạng ngày càng phức tạp và tinh vi.

5.2 Hướng phát triển trong tương lai

Dựa trên kết quả nghiên cứu hiện tại, chúng tôi đề xuất các hướng phát triển trong tương lai như sau:

1. **Mở rộng phạm vi phát hiện**: Tích hợp thêm các module phát hiện cho các mối đe dọa mới nổi như các cuộc tấn công AI-powered và tấn công zero-day.
2. **Cải thiện phương pháp xử lý dữ liệu mất cân bằng**: Áp dụng các kỹ thuật nâng cao hơn như SMOTE (Synthetic Minority Over-sampling Technique) hoặc adaptive synthetic sampling.
3. **Tối ưu hóa hiệu suất thời gian thực**: Cải thiện hiệu suất của mô hình để cho phép phát hiện mối đe dọa theo thời gian thực trong môi trường mạng tốc độ cao.

4. **Tăng cường phân tích deep packet inspection**: Phát triển các kỹ thuật phân tích sâu hơn đối với nội dung payload để phát hiện các mối đe dọa tinh vi hơn.
5. **Tích hợp học sâu**: Khám phá việc kết hợp các mô hình học sâu như LSTM hoặc CNN vào kiến trúc ensemble để cải thiện khả năng phát hiện các mẫu phức tạp.
6. **Tự động cập nhật luật**: Phát triển cơ chế để tự động cập nhật và mở rộng bộ luật phát hiện dựa trên các mối đe dọa mới.
7. **Cải thiện khả năng giải thích**: Tăng cường khả năng giải thích của mô hình để cung cấp thông tin chi tiết hơn về lý do tại sao một mẫu lưu lượng cụ thể được xác định là độc hại.
8. **Áp dụng học không giám sát và học bán giám sát**: Khám phá các phương pháp học không giám sát và bán giám sát để phát hiện các mối đe dọa mới và chưa biết.

Tài liệu tham khảo

1. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176.
2. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE symposium on security and privacy (pp. 305-316). IEEE.
3. Verizon. (2023). 2023 Data Breach Investigations Report. Verizon Business.
4. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18-28.
5. Bhattacharyya, D. K., & Kalita, J. K. (2013). Network anomaly detection: A machine learning perspective. CRC Press.
6. Zhou, Z. H. (2012). Ensemble methods: foundations and algorithms. CRC press.

7. Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from imbalanced data sets. Springer.
8. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. In Network and Distributed System Security Symposium (NDSS).
9. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
10. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147-167.
11. Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2019). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials, 21(1), 686-728.
12. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954-21961.
13. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.
14. Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., & Marchetti, M. (2018). On the effectiveness of machine and deep learning for cyber security. In 2018 10th International Conference on Cyber Conflict (CyCon) (pp. 371-390). IEEE.
15. Aburomman, A. A., & Reaz, M. B. I. (2017). A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & Security, 65, 135-152.
16. Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., & Lahza, H. F. M. (2018). Improving performance of intrusion detection system using ensemble methods and feature selection. In Proceedings of the Australasian Computer Science Week Multiconference (pp. 1-6).
17. Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(5), 649-659.

18. Jabbar, M. A., & Aluvalu, R. (2017). RFAODE: A novel ensemble intrusion detection system. Procedia Computer Science, 115, 226-234.

19. Roesch, M. (1999). Snort: Lightweight intrusion detection for networks. In LISA (Vol. 99, No. 1, pp. 229-238).

20. White, J. S., Fitzsimmons, T., & Matthews, J. N. (2013). Quantitative analysis of intrusion detection systems: Snort and Suricata. In Cyber Sensing 2013 (Vol. 8757, p. 875704). International Society for Optics and Photonics.

21. Albin, E., & Rowe, N. C. (2012). A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. In 2012 26th International Conference on Advanced Information Networking and Applications Workshops (pp. 122-127). IEEE.

22. Gu, G., Porras, P., Yegneswaran, V., Fong, M., & Lee, W. (2007). BotHunter: Detecting malware infection through IDS-driven dialog correlation. In 16th USENIX Security Symposium (USENIX Security 07).

23. Valdes, A., & Skinner, K. (2000). Adaptive, model-based monitoring for cyber attack detection. In International Workshop on Recent Advances in Intrusion Detection (pp. 80-93). Springer, Berlin, Heidelberg.

24. Gupta, M., Sharma, T., Lamba, H., & Vig, L. (2018). A network-based framework for detecting anomalies in time series data using sliding window. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 3742-3751). IEEE.

25. Hussain, J., Lalmuanawma, S., & Chhakchhuak, L. (2016). A novel network traffic classification using unsupervised artificial neural network for intrusion detection. In 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT) (pp. 42-47). IEEE.

26. Wang, W., Zhu, M., Zeng, X., Ye, X., & Sheng, Y. (2017). Malware traffic classification using convolutional neural network for representation learning. In 2017 International Conference on Information Networking (ICOIN) (pp. 712-717). IEEE.

27. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP).

28. Bartos, K., Sofka, M., & Franc, V. (2016). Optimized invariant representation of network traffic for detecting unseen malware variants. In 25th USENIX Security Symposium (USENIX Security 16) (pp. 807-822).

29. Han, J., Kamber, M., & Pei, J. (2011). Data mining: concepts and techniques. Morgan Kaufmann.

30. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.