BAN CƠ YẾU CHÍNH PHỦ
HỌC VIỆN KỸ THUẬT MẬT MÃ

TIỂU LUẬN
HỌC MÁY TRONG AN TOÀN THÔNG TIN

ĐỀ TÀI: PHÁT HIỆN MẠNG BẤT THƯỜNG VỚI XGBOOST

Giảng viên: TS. Nguyễn An Khương
Sinh viên: Nguyễn Minh Hưng (CHAT3P07)
Ngô Quang Khánh (CHAT3P09)

Lớp: CHAT3P
Niên khóa: 2024-2026

Tp. Hồ Chí Minh, 2025

ĐỀ TÀI: PHÁT HIỆN MẠNG BẤT THƯỜNG VỚI XGBOOST

❖ **ĐIỂM:**

❖ **NHẬN XÉT CỦA GIẢNG VIÊN:**

MỤC LỤC

A. MỞ ĐẦU

1. Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, các hệ thống mạng nội bộ ngày càng đóng vai trò quan trọng trong việc hỗ trợ hoạt động của doanh nghiệp, tổ chức, và cả quốc gia. Tuy nhiên, điều này cũng làm tăng nguy cơ các mối đe dọa an ninh mạng. Các hành vi bất thường trong mạng nội bộ, từ việc truy cập trái phép, lạm dụng tài nguyên, đến các cuộc tấn công nội bộ, đang trở thành thách thức lớn đối với các nhà quản trị hệ thống.

Trong bối cảnh đó, việc ứng dụng các phương pháp học máy, đặc biệt là các thuật toán mạnh mẽ như XGBoost (Extreme Gradient Boosting), đã mở ra nhiều cơ hội mới để phát hiện và ngăn chặn các hành vi bất thường một cách hiệu quả. Với khả năng xử lý dữ liệu lớn, tối ưu hóa tốc độ và độ chính xác, XGBoost là một công cụ tiềm năng để xây dựng các mô hình phát hiện mạng nội bộ bất thường.

Xuất phát từ nhu cầu cấp bách trong việc bảo vệ mạng nội bộ và khai thác sức mạnh của học máy, tôi đã chọn đề tài: "Phát hiện mạng nội bộ bất thường với XGBoost", nhằm đóng góp một phần vào việc tăng cường an ninh mạng trong các hệ thống hiện đại.

2. Mục tiêu và nhiệm vụ nghiên cứu

Mục tiêu: Nghiên cứu nhằm cung cấp một cách tiếp cận hiệu quả trong việc phát hiện các hành vi bất thường trong mạng nội bộ:

- Tối ưu hóa khả năng phát hiện sớm các mối đe dọa.
- Đưa ra góc nhìn khoa học về cách ứng dụng XGBoost trong lĩnh vực an ninh mạng.
- Tạo tiền đề cho các nghiên cứu và ứng dụng thực tiễn khác trong tương lai.

Nhiệm vụ:

- Xác định và làm rõ các khái niệm liên quan đến hành vi bất thường trong mạng nội bộ thường gặp.
- Phân tích các đặc điểm và dữ liệu mạng nội bộ, từ đó xây dựng bộ dữ liệu phù hợp để đào tạo mô hình.
- Ứng dụng XGBoost để xây dựng mô hình phát hiện bất thường và đánh giá hiệu quả của mô hình.
- Đề xuất một số phương pháp cải tiến nhằm nâng cao độ chính xác và tốc độ phát hiện hành vi bất thường.

3. Phương pháp nghiên cứu

Để đạt được các mục tiêu trên, bài tiểu luận sử dụng kết hợp các phương pháp nghiên cứu sau:

- **Phương pháp lịch sử**: Tìm hiểu các phương pháp truyền thống và hiện đại trong phát hiện bất thường mạng, cũng như quá trình phát triển của thuật toán XGBoost và các ứng dụng trong thực tế.
- **Phương pháp logic**: Phân tích, trừu tượng hóa và khái quát hóa từ dữ liệu thực tiễn, đánh giá mối quan hệ giữa các yếu tố ảnh hưởng đến hiệu quả phát hiện bất thường.
- **Phương pháp thực nghiệm**: Xây dựng và triển khai mô hình XGBoost trên bộ dữ liệu mẫu, từ đó đánh giá hiệu quả bằng các chỉ số khoa học như độ chính xác, độ nhạy, và F1-score.

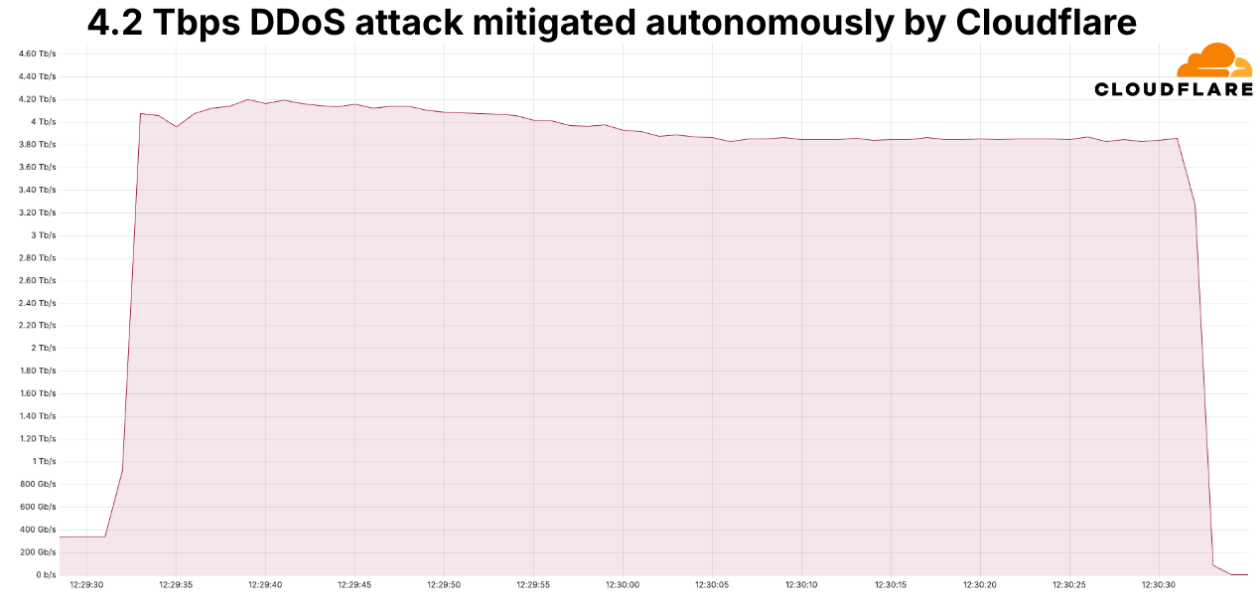
B. NỘI DUNG

CHƯƠNG I: TỔNG QUAN NGHIÊN CỨU

1.1 Các hoạt động bất thường trong mạng nội bộ

Hoạt động bất thường trong mạng nội bộ là một trong những vấn đề trọng tâm trong lĩnh vực an ninh mạng. Những hoạt động này thường liên quan đến các hành vi không phù hợp hoặc độc hại, có khả năng làm ảnh hưởng đến hiệu suất và tính an toàn của hệ thống. Trong đó, tấn công từ chối dịch vụ phân tán (DDoS) và quét cổng (port scanning) là hai biểu hiện phổ biến, thường xuyên xuất hiện trong các tình huống xâm nhập trái phép.

Tấn công DDoS (Distributed Denial of Service) là một trong những hình thức tấn công phổ biến nhất, đặc biệt trong các môi trường mạng có quy mô lớn. Mục đích chính của tấn công DDoS là làm gián đoạn hoạt động của hệ thống bằng cách gửi một lượng lớn yêu cầu đồng thời từ nhiều nguồn khác nhau. Điều này dẫn đến tình trạng máy chủ không thể xử lý kịp thời và ngừng cung cấp dịch vụ cho người dùng hợp lệ. Biểu hiện của DDoS thường bao gồm việc tăng đột biến lưu lượng mạng hoặc số lượng kết nối trong thời gian ngắn, sự chậm trễ hoặc ngừng hoạt động của máy chủ, và mức sử dụng tài nguyên hệ thống (CPU, RAM, băng thông) tăng bất thường.



Hình 1. Vào ngày 21 tháng 10 năm 2024, hệ thống của Cloudflare đã tự động phát hiện và giảm thiểu một cuộc tấn công DDoS 4,2 Tbps kéo dài khoảng một phút. [2]

Vào ngày 21 tháng 10 năm 2024, Cloudflare đã tự động phát hiện và giảm thiểu một cuộc tấn công DDoS với lưu lượng đỉnh điểm lên đến 4,2 Tbps, kéo dài khoảng một phút ở

hình 1. Đây là một trong những cuộc tấn công có quy mô lớn nhất từng được ghi nhận, vượt qua cả cuộc tấn công 3,8 Tbps mà Cloudflare đã ngăn chặn trước đó vào đầu tháng 9 năm 2024. Các cuộc tấn công DDoS với lưu lượng cao như vậy thường nhằm mục tiêu làm cạn kiệt băng thông mạng và tài nguyên hệ thống của nạn nhân, dẫn đến gián đoạn dịch vụ và ảnh hưởng đến người dùng hợp pháp. Việc Cloudflare có thể tự động phát hiện và giảm thiểu hiệu quả các cuộc tấn công này cho thấy khả năng phòng thủ mạnh mẽ của họ trong việc bảo vệ hạ tầng mạng và dịch vụ trực tuyến.

Khả năng tự động phát hiện và giảm thiểu các cuộc tấn công DDoS với quy mô lớn của Cloudflare là kết quả của việc triển khai các hệ thống phòng thủ tiên tiến, có khả năng xử lý hàng tỷ gói tin mỗi giây mà không cần sự can thiệp của con người. Điều này đảm bảo rằng các dịch vụ trực tuyến được bảo vệ liên tục, giảm thiểu tác động tiêu cực đến người dùng và doanh nghiệp.

Bên cạnh tấn công DDoS, quét cổng (port scanning) cũng là một hành vi bất thường đáng chú ý. Đây là kỹ thuật thăm dò các cổng mạng của hệ thống để xác định những cổng đang mở hoặc những dịch vụ có thể bị khai thác. Mục tiêu chính của quét cổng là thu thập thông tin để chuẩn bị cho các cuộc tấn công tiếp theo, như khai thác lỗ hổng bảo mật hoặc truy cập trái phép. Hoạt động này thường biểu hiện qua sự gia tăng đột biến số lượng yêu cầu kết nối đến các cổng khác nhau trong thời gian ngắn, hoặc xuất hiện nhiều địa chỉ IP lạ gửi yêu cầu không theo hành vi thông thường. Các trường hợp như "portsweep" (quét nhiều cổng trên một máy tính) hay "satan" (kết hợp quét cổng và khai thác lỗ hổng) trong NSL-KDD là minh chứng rõ ràng cho dạng tấn công này.

Các phần mềm và công cụ thực hiện các hành vi trên thường rất đa dạng, từ những công cụ hợp pháp nhưng bị lạm dụng như Nmap (dùng để quét cổng) đến các phần mềm chuyên dụng cho tấn công mạng như LOIC (Low Orbit Ion Cannon) thường được sử dụng trong các cuộc tấn công DDoS. Ngoài ra, những công cụ như Hping và Metasploit cũng thường xuyên bị khai thác để thực hiện các hành vi độc hại này.

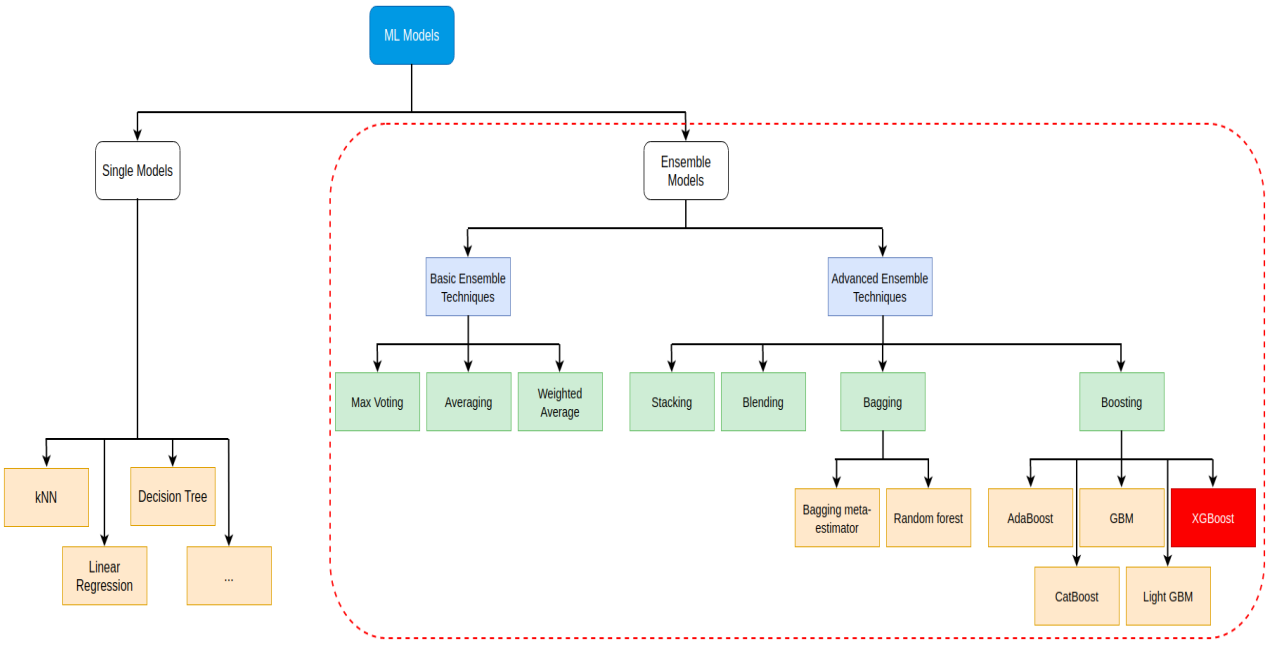
Việc nhận diện và xử lý kịp thời các hoạt động bất thường như tấn công DDoS hay quét cổng là vô cùng cần thiết để bảo vệ hệ thống mạng nội bộ. Những hành động này không chỉ giúp ngăn chặn các cuộc tấn công tiềm tàng mà còn đảm bảo tính liên tục trong hoạt động của hệ thống, giảm thiểu rủi ro và thiệt hại về tài chính cũng như uy tín cho tổ chức. Sự hiểu biết về các biểu hiện của những hoạt động này, cùng với các công cụ và phương pháp phát hiện hiệu quả, là bước quan trọng trong việc xây dựng một hệ thống mạng an toàn và đáng tin cậy.

1.2 Tổng quan về học máy – machine learning

Học máy (Machine Learning - ML) là một lĩnh vực thuộc trí tuệ nhân tạo (AI), tập trung vào việc phát triển các hệ thống có khả năng tự học và cải thiện hiệu suất dựa trên dữ liệu mà không cần lập trình cụ thể. Thay vì thực hiện các hành vi theo các quy tắc được lập trình trước, hệ thống học máy phân tích dữ liệu, rút ra quy luật, và áp dụng quy luật này để giải quyết các vấn đề.

Học máy được phân chia thành ba nhóm chính, bao gồm học có giám sát (supervised learning), học không giám sát (unsupervised learning), và học tăng cường (reinforcement learning). Trong học có giám sát, dữ liệu huấn luyện bao gồm cả đầu vào và đầu ra kỳ vọng, từ đó mô hình học cách ánh xạ giữa chúng. Ngược lại, học không giám sát chỉ sử dụng dữ liệu đầu vào, mục tiêu là tìm ra các mẫu hoặc cấu trúc ẩn trong dữ liệu. Học tăng cường lại là một phương pháp khác biệt, nơi hệ thống học cách đưa ra quyết định thông qua việc tối ưu hóa phần thưởng nhận được từ môi trường.

Hệ thống học máy thường sử dụng các thuật toán đơn lẻ (Single Models) hoặc kết hợp nhiều thuật toán khác nhau (Ensemble Models) để tăng cường hiệu quả.



Hình 2. Các mô hình học máy

Hình 2 cung cấp một cái nhìn tổng quan về các thuật toán học máy, chia chúng thành hai nhóm chính: Single Models và Ensemble Models.

- Single Models là các mô hình đơn lẻ, như k-Nearest Neighbors (kNN), Decision Tree, hay Linear Regression. Những thuật toán này phù hợp cho các bài toán đơn giản, nhưng hiệu suất có thể bị hạn chế khi xử lý dữ liệu phức tạp.
- Ensemble Models là sự kết hợp của nhiều mô hình đơn lẻ nhằm cải thiện độ chính xác và hiệu quả dự đoán. Các kỹ thuật kết hợp được phân thành hai nhóm lớn: kỹ thuật kết hợp cơ bản (Basic Ensemble Techniques) và kỹ thuật kết hợp nâng cao (Advanced Ensemble Techniques).
    - Basic Ensemble Techniques bao gồm các phương pháp như Max Voting, Averaging, và Weighted Average. Những phương pháp này sử dụng kết quả của nhiều mô hình để đưa ra dự đoán cuối cùng bằng cách tổng hợp một cách đơn giản.
    - Advanced Ensemble Techniques áp dụng các phương pháp kết hợp phức tạp hơn, như Bagging (Bootstrap Aggregating) và Boosting.

Trong Boosting, nhiều thuật toán mạnh mẽ đã được phát triển, bao gồm AdaBoost, Gradient Boosting Machine (GBM), LightGBM, CatBoost, và XGBoost. Trong đó, XGBoost nổi bật với hiệu suất cao, khả năng tối ưu hóa tốc độ tính toán và xử lý hiệu quả dữ liệu lớn, trở thành một trong những lựa chọn hàng đầu trong các cuộc thi học máy.

1.3 Thuật toán học máy XGBoost

XGBoost (eXtreme Gradient Boosting) là một thuật toán học máy thuộc nhóm Boosting, được thiết kế để tối ưu hóa hiệu suất dự đoán và tốc độ xử lý. Đây là một cải tiến của thuật toán Gradient Boosting Machine (GBM), tập trung vào việc khắc phục những hạn chế của các phiên bản Boosting truyền thống, đặc biệt là về hiệu quả tính toán và khả năng xử lý dữ liệu lớn, được các nhà khoa học dữ liệu sử dụng rộng rãi để đạt được kết quả tiên tiến nhất trong nhiều thách thức về máy học [1]

XGBoost dựa trên ý tưởng chính của Boosting: kết hợp nhiều mô hình yếu (weak learners) để tạo thành một mô hình mạnh (strong learner). Các mô hình yếu này thường là các cây quyết định nông (shallow decision trees). Quá trình huấn luyện XGBoost diễn ra theo từng vòng lặp (iteration), trong đó mỗi vòng xây dựng một cây mới để giảm thiểu lỗi (residual errors) của mô hình trước đó.

Công thức tối ưu hóa trong XGBoost không chỉ dựa trên giảm lỗi thông thường mà còn sử dụng thuật toán Taylor Series bậc hai để tính toán gradient và hessian, giúp tối ưu

hóa hiệu quả hơn. Điều này giúp XGBoost đạt được hiệu suất cao và tốc độ nhanh hơn nhiều so với các thuật toán Boosting truyền thống.

Ưu điểm của XGBoost:

- Tốc độ nhanh: XGBoost được tối ưu hóa cao cho cả bộ nhớ và tính toán, có thể sử dụng đa luồng (multithreading) để xử lý dữ liệu lớn.
- Khả năng chống overfitting: XGBoost áp dụng nhiều kỹ thuật để giảm overfitting, như regularization (L1, L2) và pruning.
- Hỗ trợ dữ liệu thiếu: XGBoost có khả năng tự động xử lý dữ liệu thiếu (missing values), một đặc điểm nổi bật so với các thuật toán khác.
- Tích hợp tính năng quan trọng: XGBoost tự động tính toán mức độ quan trọng của các đặc trưng (feature importance), giúp phân tích và giải thích kết quả dễ dàng hơn.
- Khả năng xử lý dữ liệu lớn: Nhờ cách tối ưu hóa về mặt tính toán, XGBoost có thể xử lý hiệu quả các bộ dữ liệu lớn mà không gặp phải vấn đề về hiệu năng.

XGBoost được sử dụng rộng rãi trong các bài toán học có giám sát (supervised learning), bao gồm:

- Phân loại (Classification): Dự đoán rủi ro tín dụng, phát hiện gian lận, chẩn đoán bệnh.
- Hồi quy (Regression): Dự đoán giá bất động sản, doanh thu, hoặc các chuỗi thời gian.

Nhờ tốc độ cao và khả năng tối ưu hóa, XGBoost thường xuất hiện trong các cuộc thi học máy, như Kaggle, và trong các hệ thống sản xuất lớn.

CHƯƠNG II: PHÁT HIỆN BẤT THƯỜNG TRONG MẠNG NỘI BỘ BẰNG THUẬT TOÁN HỌC MÁY XGBOOST

2.1 Phân tích gói tin trong mạng nội bộ

Phân tích gói tin trong mạng nội bộ là một trong những phương pháp nền tảng để hiểu rõ các hoạt động diễn ra trong hệ thống mạng. Mỗi gói tin mang theo các thông tin quan trọng, bao gồm địa chỉ nguồn, địa chỉ đích, giao thức, và dữ liệu truyền tải. Những thông tin này cung cấp bức tranh tổng quan về lưu lượng mạng và là cơ sở để nhận diện các hoạt động bất thường hoặc tấn công.

Trong mạng nội bộ, các gói tin thường được truyền tải theo cấu trúc chuẩn, bao gồm ba phần chính: phần tiêu đề (header), phần dữ liệu (payload), và phần kiểm tra lỗi (trailer). Phần tiêu đề mang các thông tin như giao thức (TCP, UDP, ICMP), địa chỉ IP, số cổng và cờ trạng thái, giúp xác định nguồn gốc và mục đích của gói tin. Chính phần này là nơi tập trung nhiều thông tin có giá trị để phát hiện bất thường.

Phân tích gói tin có vai trò quan trọng trong việc phát hiện các hoạt động bất thường như tấn công từ chối dịch vụ (DDoS), quét cổng (port scanning) và truy cập trái phép. Ví dụ, trong trường hợp tấn công DDoS, lưu lượng mạng thường đột ngột tăng cao, với hàng nghìn yêu cầu đến từ một hoặc nhiều địa chỉ IP đáng ngờ. Tương tự, hành vi quét cổng có thể được nhận diện thông qua việc gửi nhiều gói tin đến các cổng khác nhau trong thời gian ngắn.

Ngoài ra, các công cụ phân tích như Wireshark, Tcpdump, hoặc Snort được sử dụng để thu thập và phân tích dữ liệu gói tin. Dữ liệu này có thể được xuất dưới dạng tập tin PCAP, làm cơ sở cho các mô hình học máy. Từ việc phân tích những đặc trưng trong gói tin, chúng ta có thể xây dựng các mô hình mạnh mẽ để phát hiện bất thường, chẳng hạn như sử dụng XGBoost để phân loại hành vi mạng.

2.2 Phân tích bộ dữ liệu NSL-KDD [2]

Bộ dữ liệu NSL-KDD là một trong những tập dữ liệu quan trọng được sử dụng để nghiên cứu và huấn luyện các hệ thống phát hiện xâm nhập (IDS - Intrusion Detection Systems). Được phát triển nhằm khắc phục những hạn chế của bộ dữ liệu KDD'99, NSL-

KDD cung cấp một tập hợp dữ liệu có cấu trúc cân đối và ít trùng lặp hơn, giúp tăng hiệu quả của các mô hình học máy khi được huấn luyện.

Tập dữ liệu NSL-KDD bao gồm hai phần chính: tập huấn luyện và tập kiểm tra. Mỗi mẫu trong tập dữ liệu đại diện cho một kết nối mạng, được mô tả thông qua 41 đặc trưng, bao gồm các đặc trưng cơ bản (như thời gian kết nối, giao thức, cổng đích), các đặc trưng dựa trên nội dung (chẳng hạn số lần đăng nhập sai) và các đặc trưng thống kê dựa trên thời gian (như tần suất kết nối trong một khoảng thời gian nhất định).

Các mẫu trong tập dữ liệu được gắn nhãn thành hai loại: hành vi bình thường và hành vi bất thường. Hành vi bất thường được chia thành bốn nhóm chính như hình 3:

- DoS (Denial of Service): Tấn công làm ngừng dịch vụ, như SYN Flood hoặc UDP Flood.
- R2L (Remote to Local): Xâm nhập từ xa để chiếm quyền truy cập cục bộ, như các cuộc tấn công brute force.
- U2R (User to Root): Tấn công nhằm chiếm quyền root từ người dùng bình thường, chẳng hạn buffer overflow.
- Probe (Quét mạng): Hành vi quét cổng hoặc tìm kiếm các lỗ hổng trong hệ thống.

Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

Hình 3. Các lớp tấn công trong tập dữ liệu NSL-KDD

Ngoài các class được đề cập ở trên, bộ dữ liệu NSL-KDD còn có một số trường phân loại như Protocol Type, Service và flag phân tích từ các gói tin. Không giống như các thuộc tính Protocol Type hoặc Service có giá trị tự giải thích cho ý nghĩa của nó (giá trị mô tả kết nối), thuộc tính Flags hơi khó hiểu. Flag mô tả trạng thái của kết nối và các cờ trong gói tin có được bật hay không. Mỗi giá trị của Flag đại diện cho một trạng thái của kết nối và ý nghĩa của mỗi giá trị.

Protocol Type (2)	Service (3)				Flag (4)
• icmp • tcp • udp	• other • link • netbios_ssn • smtp • netstat • ctf • ntp_u • harvest • efs • klogin • systat • exec • nntp • pop_3 • printer • vmnet • netbios_ns	• urh_i • ssh • http_8001 • iso_tsap • aol • sql_net • shell • supdup • auth • whois • discard • sunrpc • urp_i • Rje • ftp • daytime • domain_u • pm_dump	• time • hostnames • name • ecr_i • bgp • telnet • domain • ftp_data • nnsp • courier • finger • uucp_path • X11 • imap4 • mtp • login • tftp_u • kshell	• private • http_2784 • echo • http • ldap • tim_i • netbios_dgm • uucp • eco_i • Remote_job • IRC • http_443 • red_i • Z39_50 • Pop_2 • gopher • Csnet_ns	• OTH • S1 • S2 • RSTO • RSTRs • RSTOS0 • SF • SH • REJ • S0 • S3

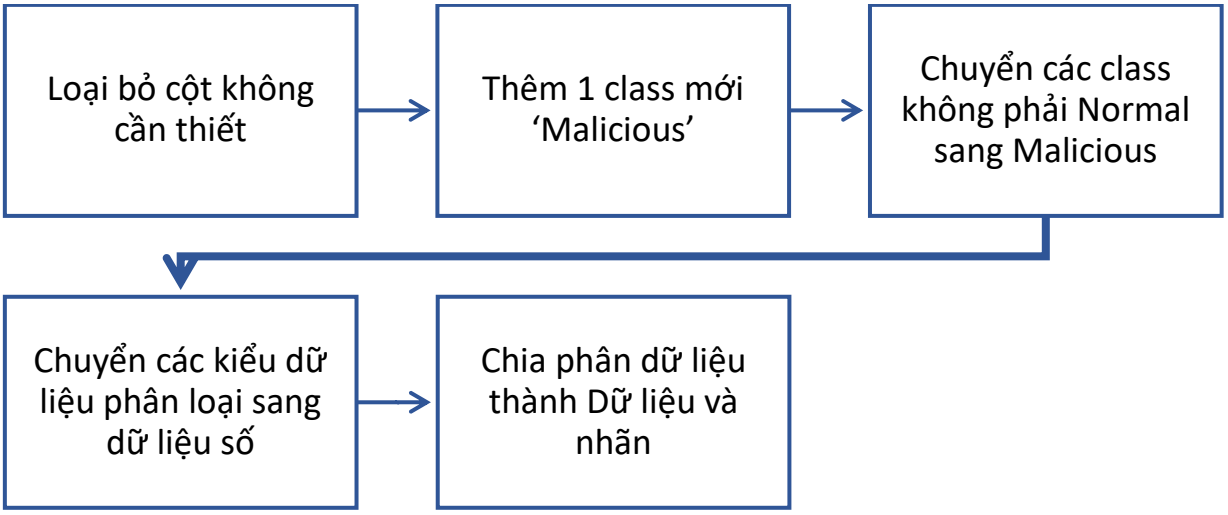
Hình 4. Giá trị các thuộc tính kiểu phân loại có thể có của NSL-KDD

Một trong những ưu điểm nổi bật của NSL-KDD là việc loại bỏ các mẫu trùng lặp và không cần thiết, điều này giúp giảm thiểu hiện tượng "overfitting" khi huấn luyện mô hình. Bên cạnh đó, tập kiểm tra của NSL-KDD được thiết kế để chứa các mẫu chưa từng xuất hiện trong tập huấn luyện, giúp kiểm tra khả năng tổng quát hóa của mô hình.

Trong nghiên cứu này, bộ dữ liệu NSL-KDD được sử dụng làm cơ sở để huấn luyện mô hình XGBoost. Các đặc trưng từ tập dữ liệu được chuẩn hóa và mã hóa trước khi đưa vào mô hình. XGBoost tận dụng khả năng học mạnh mẽ của thuật toán boosting để phân biệt giữa hành vi bình thường và bất thường.

CHƯƠNG III. THỰC NGHIỆM PHÁT HIỆN GÓI TIN BẤT THƯỜNG BẰNG THUẬT TOÁN HỌC MÁY XGBOOST

3.1 Tiền xử lý bộ dữ liệu NSL-KDD



Hình 5. Quy trình tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một bước quan trọng trong việc chuẩn bị dữ liệu cho các mô hình học máy, đặc biệt là với các thuật toán như XGBoost. Để đưa dữ liệu vào mô hình XGBoost một cách hiệu quả, các bước tiền xử lý dưới đây sẽ giúp dữ liệu trở nên phù hợp với yêu cầu của mô hình.

a. Loại bỏ cột không cần thiết

Dữ liệu NSL-KDD là một tập dữ liệu lớn thường được sử dụng trong các bài toán phát hiện xâm nhập mạng. Tuy nhiên, không phải tất cả các cột trong dữ liệu đều có giá trị đối với mô hình học máy. Các cột không có thông tin quan trọng, chẳng hạn như cột "classnum" có thể gây nhiễu và làm giảm hiệu quả của mô hình. Cột classnum, nếu là một biểu diễn mã hóa của lớp mục tiêu, cũng có thể được loại bỏ khi đã chuyển các nhãn sang dạng cần thiết cho mô hình.

b. Thêm một class mới 'Malicious'

Dữ liệu NSL-KDD có các lớp nhãn như Normal và các loại xâm nhập mạng khác. Để đơn giản hóa bài toán phân loại và tập trung vào việc phát hiện hành vi xâm nhập, ta sẽ thêm một lớp mới mang tên Malicious để đại diện cho tất cả các hành vi không phải là

Normal. Điều này giúp mô hình dễ dàng phân biệt giữa hành vi bình thường và hành vi xâm nhập mà không phải phân biệt quá nhiều lớp khác nhau.

c. Chuyển các class không phải Normal sang Malicious

Sau khi đã tạo lớp mới Malicious, ta cần chuyển tất cả các class không phải Normal thành lớp Malicious. Đây là một bước quan trọng, giúp giảm số lượng lớp trong bài toán phân loại và làm cho vấn đề trở nên dễ dàng hơn. Tất cả các mẫu có nhãn khác ngoài Normal sẽ được gán nhãn Malicious, từ đó các mẫu Normal và Malicious sẽ tạo thành hai lớp cơ bản cho bài toán phân loại.

d. Chuyển các kiểu dữ liệu phân loại sang dữ liệu số

Để mô hình học máy có thể xử lý được dữ liệu, các biến phân loại (categorical variables) cần được chuyển đổi thành các dạng dữ liệu số. Điều này có thể thực hiện thông qua các kỹ thuật như mã hóa one-hot (one-hot encoding) hoặc mã hóa nhãn (label encoding). Ví dụ, nếu có một cột chứa thông tin về giao thức mạng như TCP, UDP, ICMP, chúng ta có thể sử dụng mã hóa one-hot để chuyển chúng thành các cột nhị phân, mỗi cột biểu diễn một giao thức riêng biệt. Việc chuyển đổi này giúp mô hình dễ dàng xử lý và học các mối quan hệ trong dữ liệu. Ở bài nghiên này sẽ dùng mã hóa nhãn.

e. Chia phân dữ liệu thành Dữ liệu và Nhãn

Cuối cùng, dữ liệu cần được chia thành hai phần: phần dữ liệu đầu vào (features) và nhãn (labels). Phần dữ liệu đầu vào chứa các đặc trưng của mẫu (chẳng hạn như các giá trị từ các cột còn lại sau khi loại bỏ các cột không cần thiết), trong khi nhãn là lớp mục tiêu mà mô hình cần dự đoán (trong trường hợp này là Normal và Malicious). Việc tách biệt rõ ràng dữ liệu và nhãn giúp mô hình dễ dàng học và tối ưu hóa.

3.2 Kiến trúc mô hình học máy - XGBoost

XGBoost (Extreme Gradient Boosting) là một trong những thuật toán học máy mạnh mẽ và phổ biến hiện nay, đặc biệt trong các bài toán phân loại và hồi quy. Mô hình này sử dụng phương pháp Gradient Boosting với các cây quyết định (Decision Trees) làm các mô hình yếu (weak learners). Cấu trúc của mô hình XGBoost bao gồm các tham số có thể điều chỉnh để tối ưu hóa hiệu suất và giảm thiểu overfitting. Sau đây, chúng ta sẽ phân tích các tham số cấu hình của mô hình XGBoost được sử dụng trong bài toán phát hiện xâm nhập mạng.

▪ XGBClassifier
▪ n_estimators= 100
▪ max_depth= 8
▪ booster= 'gbtree’
▪ subsample= 0.5
▪ colsample_bytree= 0.5
▪ importance_type= 'gain’
▪ objective='binary:logistic'
▪ eval_metric='logloss'

Bảng 1. Mô hình XGBoost và siêu tham số

Mỗi cây quyết định trong XGBoost là một mô hình yếu (weak learner) giúp cải thiện dần độ chính xác của mô hình. Với n_estimators=100, mô hình sẽ thực hiện 100 vòng boosting, nghĩa là tạo ra 100 cây quyết định. Việc lựa chọn số lượng cây phụ thuộc vào dữ liệu và yêu cầu của bài toán. Số cây quá ít có thể dẫn đến underfitting, trong khi số cây quá nhiều có thể gây overfitting. Tham số max_depth xác định độ sâu tối đa của mỗi cây quyết định. Một độ sâu cao có thể giúp mô hình học các mối quan hệ phức tạp trong dữ liệu, nhưng cũng dễ dẫn đến overfitting nếu cây học quá chi tiết vào dữ liệu huấn luyện. Trong trường hợp này, max_depth=8 sẽ giúp các cây quyết định có đủ độ phức tạp để phân biệt các lớp, đồng thời ngăn ngừa việc mô hình học quá chi tiết vào các mẫu đặc biệt trong dữ liệu huấn luyện. Tham số booster xác định loại mô hình yếu mà XGBoost sử dụng. Trong trường hợp này, chúng ta sử dụng gbtree, nghĩa là các cây quyết định sẽ được dùng làm mô hình yếu. Các mô hình khác như gblinear (dùng hồi quy tuyến tính) có thể được chọn, nhưng việc sử dụng cây quyết định thường mang lại kết quả tốt hơn trong các bài toán phân loại phức tạp như phát hiện xâm nhập mạng. Tham số subsample kiểm soát tỷ lệ mẫu ngẫu nhiên được lấy từ tập huấn luyện để xây dựng mỗi cây quyết định. Việc sử dụng tỷ lệ mẫu nhỏ hơn 1 (ở đây là 50%) giúp giảm thiểu overfitting bằng cách tạo ra các cây quyết định khác nhau, giúp mô hình tổng thể đa dạng hơn và ít bị ảnh hưởng bởi các mẫu cụ thể trong dữ liệu. Tham số này có thể được điều chỉnh để tìm ra giá trị tối ưu cho bài toán cụ thể. Tham số colsample_bytree kiểm soát tỷ lệ các đặc trưng được chọn ngẫu nhiên cho mỗi cây quyết định. Với giá trị 0.5, mô hình chỉ sử dụng 50% số đặc trưng có sẵn cho mỗi cây, điều này giúp giảm thiểu khả năng overfitting và làm cho mô hình trở nên tổng quát hơn. Điều này cũng giúp tăng tính đa dạng của các cây trong mô hình boosting. Tham số

importance_type xác định phương pháp đánh giá độ quan trọng của các đặc trưng trong quá trình huấn luyện. Khi chọn gain, mô hình sẽ sử dụng mức độ cải thiện của hàm mất mát (loss function) để đánh giá sự đóng góp của mỗi đặc trưng. Điều này giúp xác định được các đặc trưng quan trọng nhất trong việc phân loại và cung cấp cái nhìn về các yếu tố quyết định trong mô hình. Tham số objective xác định mục tiêu của bài toán. Với binary:logistic, mô hình XGBoost sẽ thực hiện bài toán phân loại nhị phân, trong đó lớp Normal và lớp Malicious sẽ được phân loại. Mô hình sẽ sử dụng hàm logistic để tính toán xác suất thuộc về mỗi lớp, phù hợp với bài toán phát hiện xâm nhập mạng trong đó các hành vi xâm nhập (Malicious) được phân biệt với hành vi bình thường (Normal). Tham số eval_metric xác định chỉ số được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. logloss là một chỉ số phổ biến trong các bài toán phân loại nhị phân, đo lường mức độ sai lệch giữa xác suất dự đoán và giá trị thực tế. Một giá trị logloss thấp cho thấy mô hình dự đoán gần đúng với nhãn thực tế, giúp kiểm tra độ chính xác của mô hình trong từng vòng huấn luyện.

CHƯƠNG IV. KẾT QUẢ THỰC NGHIỆM, ĐÁNH GIÁ

Trong chương này, chúng ta sẽ trình bày kết quả thực nghiệm của mô hình XGBoost trong bài toán phát hiện xâm nhập mạng, đồng thời phân tích và đánh giá các chỉ số hiệu suất của mô hình. Kết quả thu được cho các chỉ số AUROC, Precision, Recall, và F1 Score lần lượt là:

- AUROC: 0.8255
- Accuracy : 0.8052
- Precision: 0.9693
- Recall: 0.6793
- F1 Score: 0.7988

AUROC là một chỉ số quan trọng để đánh giá hiệu suất của mô hình phân loại, đặc biệt trong các bài toán phân loại không cân bằng. Giá trị AUROC càng cao, mô hình càng hiệu quả trong việc phân biệt giữa các lớp, tức là giữa lớp Normal và lớp Malicious. Giá trị AUROC của 0.8255 cho thấy mô hình có khả năng phân loại khá tốt, nhưng vẫn có thể cải thiện để đạt hiệu quả cao hơn.

Accuracy là tỷ lệ phần trăm mẫu được phân loại đúng. Với giá trị Accuracy là 0.8052, mô hình đạt khoảng 80.5% đúng trong việc phân loại các mẫu. Tuy nhiên, trong trường hợp dữ liệu không cân bằng (với số lượng mẫu Normal nhiều hơn Malicious), Accuracy có thể không phản ánh chính xác hiệu suất của mô hình, vì mô hình có thể dễ dàng dự đoán Normal để đạt được tỷ lệ chính xác cao mà không thực sự nhận diện tốt các mẫu Malicious.

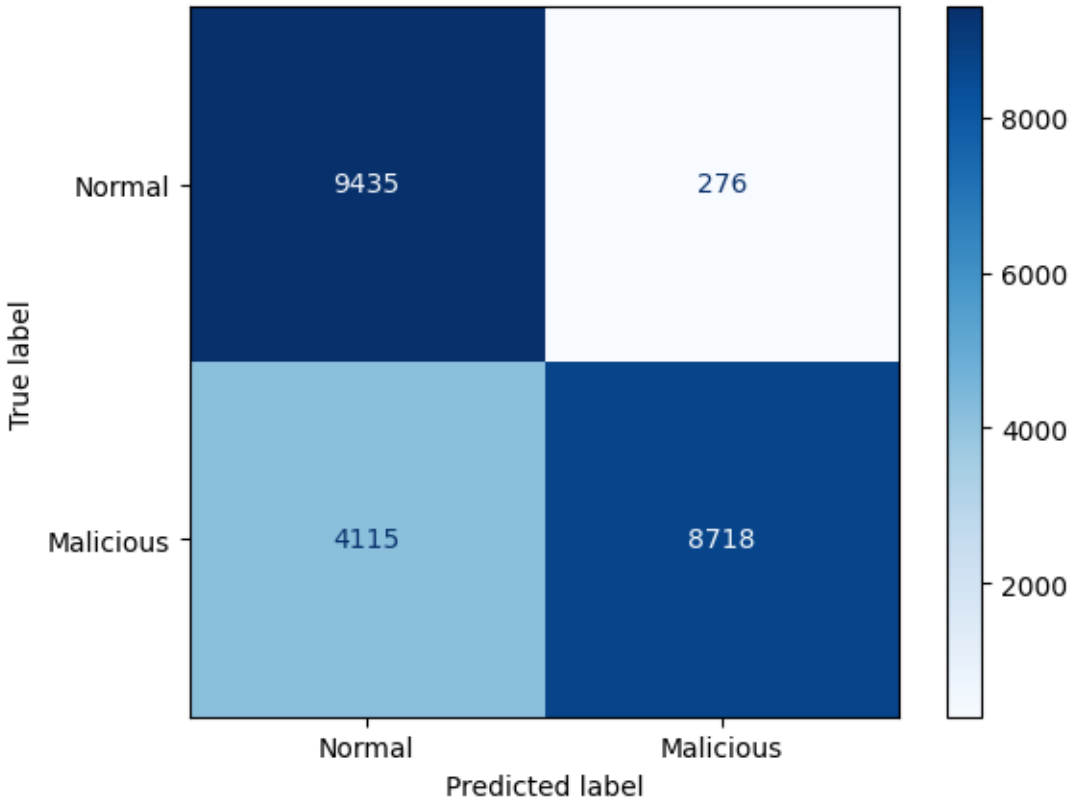
Precision đo lường tỷ lệ giữa số lượng mẫu được mô hình phân loại chính xác là Malicious so với tổng số mẫu được phân loại là Malicious. Với Precision cao (0.9693), mô hình rất ít khi phân loại nhầm các mẫu Normal thành Malicious, điều này cho thấy mô hình rất chính xác khi dự đoán hành vi xâm nhập mạng.

Recall đo lường tỷ lệ mẫu Malicious thực sự được phân loại đúng. Mặc dù Precision cao, nhưng Recall chỉ đạt 0.6793, cho thấy mô hình còn bỏ sót một số mẫu

Malicious, tức là có một tỷ lệ khá lớn các mẫu xâm nhập bị dự đoán nhầm là Normal. Điều này chỉ ra rằng mô hình chưa hoàn toàn hiệu quả trong việc nhận diện tất cả các mẫu hành vi xâm nhập.

F1 Score là trung bình điều hòa giữa Precision và Recall, là một chỉ số tổng quát giúp đánh giá hiệu suất mô hình. Với F1 Score là 0.7988, mô hình đạt hiệu suất khá tốt, nhưng có thể cải thiện thêm, đặc biệt là trong việc tăng Recall mà không làm giảm Precision quá nhiều.

Dựa vào các chỉ số trên, một trong những vấn đề chính mà mô hình gặp phải là overfitting. Mặc dù Precision cao, nhưng Recall không đạt yêu cầu, đặc biệt là trong việc nhận diện các mẫu Malicious. Điều này cho thấy mô hình có thể học quá kỹ các mẫu huấn luyện và trở nên quá đặc thù với dữ liệu huấn luyện, dẫn đến việc bỏ sót các mẫu mới trong dữ liệu kiểm tra hoặc thực tế như dữ liệu từ hình 6.



Hình 6. Ma trận nhầm lẫn của XGBoost sau khi huấn luyện

False Positives cao: Mô hình đang phân loại một số lượng các mẫu Normal thành Malicious, chứng tỏ mô hình đã học quá kỹ các đặc trưng trong dữ liệu huấn luyện, dẫn đến việc nhận diện nhầm mẫu. Đây là dấu hiệu rõ ràng của overfitting, khi mô hình quá chú ý đến các chi tiết và mẫu huấn luyện, thay vì học các quy luật chung.

False Negatives cao: Mô hình đang bỏ sót một lượng lớn mẫu Malicious, điều này cho thấy mô hình không tổng quát tốt và vẫn chưa nhận diện đầy đủ các hành vi xâm nhập mạng. Mặc dù Precision cao, Recall lại thấp, và mô hình có xu hướng ưu tiên phân loại chính xác hơn thay vì nhận diện đầy đủ mẫu Malicious. Đây cũng là dấu hiệu của overfitting, vì mô hình có thể đã học quá mức vào dữ liệu huấn luyện và không thể tổng quát cho dữ liệu mới.

Để giảm thiểu hiện tượng overfitting và cải thiện hiệu suất của mô hình, các phương pháp sau đây có thể được áp dụng:

Early Stopping là một kỹ thuật hữu ích trong việc ngừng huấn luyện mô hình khi hiệu suất trên tập kiểm tra không còn cải thiện sau một số vòng huấn luyện nhất định. Điều này giúp ngừng huấn luyện trước khi mô hình bắt đầu học quá kỹ các đặc trưng trong dữ liệu huấn luyện, từ đó giảm overfitting. Việc sử dụng Early Stopping giúp mô hình đạt được điểm cân bằng giữa việc học đúng các mẫu và không quá khớp với dữ liệu huấn luyện.

Việc điều chỉnh các siêu tham số của mô hình là rất quan trọng để tối ưu hóa hiệu suất và giảm overfitting. Một cách hiệu quả để tìm kiếm các giá trị siêu tham số tốt nhất là sử dụng RandomizedSearchCV thay vì tìm kiếm theo cách thủ công (Grid Search). RandomizedSearchCV giúp thử nghiệm một loạt các giá trị tham số trong không gian lớn hơn và tìm ra các tham số tối ưu mà không cần thử tất cả các kết hợp, tiết kiệm thời gian và tài nguyên.

Một lý do có thể dẫn đến overfitting trong mô hình là cách xử lý các dữ liệu phân loại (categorical data). Việc sử dụng Label Encoding cho các dữ liệu phân loại có thể khiến mô hình bị nhầm lẫn vì mô hình có thể hiểu các nhãn dưới dạng thứ tự (ordinal), trong khi thực tế các nhãn này không có thứ tự rõ ràng. Thay vào đó, sử dụng One-Hot Encoding cho các đặc trưng phân loại sẽ giúp mô hình học các mối quan hệ không thứ tự mà không gây hiểu nhầm. Việc này có thể giúp mô hình cải thiện khả năng phân loại và giảm overfitting bằng cách tạo ra các đặc trưng rõ ràng hơn.

Kết quả thực nghiệm cho thấy mô hình XGBoost đã có những bước tiến đáng kể trong việc phân loại các hành vi xâm nhập mạng. Tuy nhiên, vấn đề overfitting vẫn còn

tồn tại, đặc biệt là trong việc nhận diện các mẫu Malicious. Để cải thiện hiệu suất, các kỹ thuật như Early Stopping, RandomizedSearchCV để tối ưu siêu tham số và việc chuyển từ Label Encoding sang One-Hot Encoding sẽ giúp giảm overfitting và nâng cao độ chính xác của mô hình.

TÀI LIỆU THAM KHẢO

[1] T. Chen, "XGBoost: A Scalable Tree Boosting System," 2017.

[2] O. Yoachimik, "Cloudflare's Q3 DDoS report," 23 10 2024. [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-for-2024-q3/.

[3] M. H. Zaib, "NSL-KDD," 2019. [Online]. Available: https://www.kaggle.com/datasets/hassan06/nslkdd.