

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC
TẬP TỐT NGHIỆP ĐẠI HỌC

***Đề tài: “Ứng dụng học máy giám sát SVM
phát hiện tấn công mạng”***

Người hướng dẫn : Phan Thanh Hy
Sinh viên thực hiện : Võ Đặng Quốc Huy
Mã số sinh viên : N20DCAT022
Lớp : D20CQAT01-N
Khóa : 2020-2025
Ngành : An toàn thông tin
Hệ : Đại học chính quy

TP.HCM, tháng 08 năm 2024

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC

***Đề tài: “Ứng dụng học máy giám sát SVM
phát hiện tấn công mạng”***

Người hướng dẫn : Phan Thanh Hy

Sinh viên thực hiện : Võ Đặng Quốc Huy

Mã số sinh viên : N20DCAT022

Lớp : D20CQAT01-N

Khóa : 2020-2025

Ngành : An toàn thông tin

Hệ : Đại học chính quy

TP.HCM, tháng 08 năm 2024

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Phan Thanh Hy, người đã tận tình hướng dẫn em trong suốt quá trình thực tập tốt nghiệp vừa qua. Sự nhiệt tình, tận tụy và những kiến thức quý báu mà thầy đã truyền đạt đã giúp em trưởng thành hơn rất nhiều về cả chuyên môn lẫn tư duy làm việc.

Những bài học và kinh nghiệm mà thầy chia sẻ không chỉ giúp em hoàn thành tốt công việc thực tập, mà còn mở ra cho em những góc nhìn mới về ngành nghề mà em đang theo đuổi. Sự hỗ trợ và khích lệ của thầy trong những lúc khó khăn đã tiếp thêm động lực để em không ngừng cố gắng và hoàn thiện bản thân.

Em thật sự biết ơn thầy vì đã dành thời gian quý báu của mình để hướng dẫn, chỉ dạy và định hướng cho em. Em sẽ luôn trân trọng những gì đã học được từ thầy và áp dụng nó vào con đường sự nghiệp sau này.

Một lần nữa, em xin cảm ơn thầy Hy và chúc thầy luôn mạnh khỏe, hạnh phúc và tiếp tục thành công trong sự nghiệp giảng dạy.

Thành phố Hồ Chí Minh, tháng 8 năm 2024

Sinh viên thực hiện đề tài

Võ Đặng Quốc Huy

MỤC LỤC

LỜI CẢM ƠN.....	i
MỤC LỤC.....	ii
DANH MỤC CÁC BẢNG.....	iv
DANH MỤC CÁC SƠ ĐỒ, HÌNH.....	v
KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT.....	vii
MỞ ĐẦU.....	ix
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI.....	1
1.1. Mục đích nghiên cứu.....	1
1.2. Đối tượng nghiên cứu.....	1
1.3. Phương pháp nghiên cứu.....	1
1.3.1. Phương pháp luận.....	1
1.3.2. Phương pháp đánh giá dựa trên cơ sở toán học.....	2
CHƯƠNG 2: TỔNG QUAN LÝ THUYẾT.....	3
2.1. Tổng quan về tấn công mạng.....	3
2.1.1. Khái niệm về tấn công mạng.....	3
2.1.2. Các giai đoạn tấn công mạng.....	3
2.1.3. Các kỹ thuật tấn công phổ biến.....	5
2.1.4. Lý thuyết hoạt động của malware.....	7
2.1.5. Lý thuyết hoạt động của malware evasion.....	8
2.1.6. Các phương pháp phát hiện tấn công mạng.....	12
2.2. Hệ thống giám sát phát hiện xâm nhập mạng IDS (Intrusion Detection System).....	13
2.2.1. Khái niệm IDS.....	13
2.2.2. Các loại IDS.....	13
2.2.3. Phương pháp phát hiện của IDS.....	14
2.2.4. Nhược điểm của hệ thống giám sát IDS truyền thống.....	15
2.3. Tổng quan về học máy có giám sát.....	15
2.3.1. Khái niệm học máy có giám sát.....	15
2.3.2. Thuật toán Hồi quy tuyến tính (Linear Regression).....	16
2.3.3. Thuật toán Hồi quy logistic (Logistic Regression).....	16
2.3.4. Thuật toán Cây quyết định (Decision Trees).....	17
2.3.5. Thuật toán Naive Bayes.....	17

2.3.6. Thuật toán Support Vector Machine (SVM).....	18
2.4. Các công trình liên quan.....	25
CHƯƠNG 3: XÂY DỰNG MÔ HÌNH HỌC MÁY SVM PHÁT HIỆN TẤN CÔNG	
DOS	28
3.1. Giới thiệu bộ dữ liệu.....	28
3.2. Xử lý dữ liệu.....	32
3.2.1. Phân tích dữ liệu	33
3.2.2. Tiền xử lý dữ liệu.....	39
3.3. Giai đoạn huấn luyện	48
3.4. Giai đoạn kiểm tra.....	49
3.4.1. Confusion Matrix.....	49
3.4.2. Accuracy.....	50
3.4.3. Precision	51
3.4.4. Recall.....	51
3.4.5. F1-Score	51
3.4.6. Hinge Loss.....	51
3.4.7. Mô phỏng tấn công DoS	53
3.5. Kết quả thực nghiệm.....	54
3.5.1. Kiểm tra trên bộ dữ liệu	54
3.5.2. Kiểm tra trên dữ liệu log mới.....	56
CHƯƠNG 4: KẾT LUẬN VÀ ĐÁNH GIÁ.....	62
4.1. Kết quả đạt được.....	62
4.1.1. Về mặt lý thuyết	62
4.1.2. Về mặt thực tiễn.....	62
4.2. Đánh giá	62
4.3. Hướng phát triển.....	62
DANH MỤC TÀI LIỆU THAM KHẢO.....	64

DANH MỤC CÁC BẢNG

Bảng 3.1: Các kịch bản tấn công của bộ dữ liệu.....	30
Bảng 3.2: Lựa chọn đặc trưng.....	31
Bảng 3.3: Chi tiết các tập dữ liệu định dạng csv	32
Bảng 3.4: Các siêu tham số cho Hyperparameter Tuning	48
Bảng 3.5: Các chỉ số hiệu suất trên tập huấn luyện và tập kiểm tra	56

DANH MỤC CÁC SƠ ĐỒ, HÌNH

Hình 2.1: Kỹ thuật né tránh dựa trên hành vi – Rỗng hóa tiến trình [2].....	11
Hình 2.2: Mô hình tập dữ liệu một chiều.....	19
Hình 2.3: Mô hình tập dữ liệu một chiều có dữ liệu gần điểm ngưỡng.....	19
Hình 2.4: Mô hình tập dữ liệu một chiều áp dụng Maximum Margin Classifier để phân loại.....	20
Hình 2.5: Mô hình tập dữ liệu một chiều về nhược điểm của Maximum Margin Classifier.....	20
Hình 2.6: Mô hình tập dữ liệu một chiều áp dụng SVC để phân loại.....	21
Hình 2.7: Sự cân bằng giữa độ lệch (bias) và phương sai (variance)	21
Hình 2.8: Mô hình tập dữ liệu hai chiều sử dụng SVC để phân loại	22
Hình 2.9: Mô hình tập dữ liệu một chiều không tuyến tính.....	22
Hình 2.10: Mô hình tập dữ liệu hai chiều áp dụng Kernel Trick để tìm hyperplane.....	23
Hình 2.11: Mô hình tập dữ liệu hai chiều biểu diễn tính toán dot product.....	24
Hình 2.12: Mô hình tập dữ liệu hai chiều áp dụng thuật toán SVR.....	25
Hình 3.1: Sơ đồ tổng quát quá trình xây dựng mô hình SVM.....	28
Hình 3.2: Mô hình cấu trúc mạng [9]	29
Hình 3.3: Code hiển thị số lượng hàng (mẫu) và cột (đặc trưng)	34
Hình 3.4: Biểu đồ cột ngang tỉ lệ phần trăm của mỗi nhãn	36
Hình 3.5: Biểu đồ tròn tỉ lệ phần trăm các mẫu bình thường và mẫu tấn công	38
Hình 3.6: Biểu đồ cột tỉ lệ phần trăm các loại giao thức.....	39
Hình 3.7: Số lượng mẫu khi chưa cân bằng dữ liệu.....	42
Hình 3.8: Số lượng mẫu sau khi cân bằng dữ liệu	43
Hình 3.9: Biểu đồ cột phần trăm phương sai giải thích được	45
Hình 3.10: Kết quả huấn luyện mô hình SVC	49
Hình 3.11: Code và kết quả chỉ số ma trận nhầm lẫn trên tập kiểm tra.....	50
Hình 3.12: Code và kết quả các chỉ số đánh giá trên tập huấn luyện	52
Hình 3.13: Code và kết quả các chỉ số đánh giá trên tập kiểm tra.....	53
Hình 3.14: Mô hình mô phỏng tấn công DoS bằng công cụ Golden Eye.....	53
Hình 3.15: Biểu đồ ranh giới quyết định của SVM với Kernel RBF (không gian PCA)	54
Hình 3.16: Ma trận nhầm lẫn trên tập kiểm tra.....	55
Hình 3.17: Thực hiện tấn công DoS vào web server Ubuntu bằng GoldenEye (1).....	56
Hình 3.18: Thực hiện tấn công DoS vào web server Ubuntu bằng GoldenEye (2).....	57
Hình 3.19: Bắt gói tin dữ liệu tấn công DoS bằng Wireshark	57
Hình 3.20: Chuyển đổi tập tin pcap sang định dạng csv.....	58

Hình 3.21: Nội dung dữ liệu log trong định dạng csv	58
Hình 3.22: Import các thư viện cần thiết cho công việc dự đoán mẫu dữ liệu	59
Hình 3.23: Kết quả dataframe trích từ dữ liệu log mô phỏng tấn công DoS (1)	60
Hình 3.24: Kết quả dataframe trích từ dữ liệu log mô phỏng tấn công DoS (2)	60
Hình 3.25: Tải các mô hình chuẩn hóa dữ liệu và PCA bằng joblib	61
Hình 3.26: Tải mô hình SVM bằng joblib và kết quả dự đoán mẫu dữ liệu.....	61

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

Viết tắt	Tiếng anh	Tiếng việt
API	Application Programming Interface	Giao diện lập trình ứng dụng
AWS	Amazon Web Services	Dịch vụ Web của Amazon
DdoS	Distributed denial-of-service	Tấn công từ chối dịch vụ phân tán
FTP	File Transfer Protocol	Giao thức Chuyển tập tin
DoS	Denial of Service	Tấn công từ chối dịch vụ
HIDS	Host-based Intrusion Detection System	Hệ thống phát hiện xâm nhập dựa trên máy chủ
HTTP	Hypertext Transfer Protocol	Giao thức Truyền tải Siêu văn bản
IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập
IOC	Indicator of Compromise	Dấu hiệu của sự xâm nhập
IP	Internet Protocol	Giao thức internet
LAN	Local Area Network	Mạng cục bộ
MITM	Man-In-The-Middle	Tấn công trung gian
NaN	Not a Number	Không phải là số
NIDS	Network-based Intrusion Detection System	Hệ thống phát hiện xâm nhập dựa trên mạng
PCA	Principal Component Analysis	Phân tích thành phần chính
SMOTE	Synthetic Minority Over-sampling Technique	Kỹ thuật Tạo mẫu thiểu số Tổng hợp
SSH	Secure Shell	Giao thức Vô bảo mật

SIEM	Security Information and Event Management	Quản lý thông tin và sự kiện bảo mật
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
SVC	Support Vector Classification / Support Vector Classifier	Phân loại dựa trên Vector hỗ trợ
SVM	Support Vector Machine	Máy Vector hỗ trợ
SVR	Support Vector Regression / Support Vector Regressor	Hồi quy dựa trên Vector hỗ trợ
TCP	Transmission Control Protocol	Giao thức Điều khiển Truyền dẫn
UDP	User Datagram Protocol	Giao thức Gói tin Người dùng
VM	Virtual Machine	Máy ảo
WAN	Wide Area Network	Mạng diện rộng

MỞ ĐẦU

Trong bối cảnh thế giới số hóa hiện nay, việc ngăn chặn hoạt động tấn công mạng ngày càng trở nên cấp thiết hơn bao giờ hết. Với sự gia tăng không ngừng của các cuộc tấn công mạng như ransomware, phishing, và DoS/DDoS, an ninh mạng không chỉ là vấn đề kỹ thuật mà còn là một yếu tố sống còn đối với các tổ chức và cá nhân. Những cuộc tấn công này có thể gây ra thiệt hại nghiêm trọng về tài chính, mất mát dữ liệu quan trọng, và uy tín của tổ chức bị suy giảm. Hơn nữa, tấn công mạng còn đe dọa đến an ninh quốc gia khi các cơ sở hạ tầng quan trọng như điện lực, y tế và giao thông bị tấn công. Do đó, việc đầu tư vào các biện pháp bảo mật, nâng cao nhận thức về an ninh mạng và hợp tác quốc tế trong việc chia sẻ thông tin và kỹ thuật phòng chống tấn công mạng là vô cùng cần thiết để bảo vệ sự ổn định và an toàn của xã hội hiện đại.

An ninh mạng không chỉ đơn giản là việc ngăn chặn các cuộc tấn công mà còn bao gồm việc xây dựng một hệ thống phòng thủ vững chắc có khả năng phản ứng nhanh chóng và hiệu quả khi có sự cố xảy ra. Các biện pháp này cần phải được cập nhật liên tục để theo kịp với những phương thức tấn công ngày càng tinh vi của tin tặc. Việc đào tạo và nâng cao nhận thức về an ninh mạng cho người dùng cũng đóng một vai trò quan trọng, bởi vì một phần lớn các cuộc tấn công bắt nguồn từ sự thiếu hiểu biết và cẩn trọng của người dùng.

Trong những năm gần đây, sự phát triển mạnh mẽ của trí tuệ nhân tạo, đặc biệt là các kỹ thuật học máy, đã mở ra nhiều giải pháp tiềm năng cho việc phát hiện tấn công mạng với độ chính xác và hiệu quả cao hơn các phương pháp truyền thống. Học máy có thể phân tích lượng dữ liệu lớn một cách nhanh chóng và phát hiện ra những bất thường mà các phương pháp truyền thống có thể bỏ sót. Một trong những thuật toán học máy được đánh giá cao trong việc phát hiện tấn công mạng là học giám sát SVM. SVM là một thuật toán học máy mạnh mẽ, có khả năng phân loại dữ liệu hiệu quả ngay cả trong không gian nhiều chiều. Thuật toán này không chỉ có khả năng xử lý các dữ liệu phức tạp mà còn có thể tạo ra các mô hình dự đoán chính xác cao, giúp phát hiện sớm các cuộc tấn công mạng. SVM có thể học từ các mẫu dữ liệu đã được gắn nhãn để phân loại chính xác các sự kiện có khả năng là tấn công mạng và giảm thiểu các cảnh báo sai, từ đó nâng cao hiệu quả của hệ thống bảo mật.

Vì vậy, việc ứng dụng học máy có giám sát sử dụng thuật toán SVM để phát hiện tấn công mạng là một trong những giải pháp hiệu quả, có tiềm năng trong công cuộc đối phó với tình trạng tấn công mạng ngày càng tinh vi hơn.

Bài báo cáo thực tập tốt nghiệp với đề tài “Ứng dụng học máy giám sát SVM phát hiện tấn công mạng” sẽ bao gồm các chương sau:

Chương 1: Tổng quan đề tài

Chương 2: Tổng quan lý thuyết

Chương 3: Xây dựng mô hình học máy SVM phát hiện tấn công DoS

Chương 4: Kết luận và đánh giá

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Mục đích nghiên cứu

Mục đích chính của đề tài là phát triển một mô hình học máy phát hiện tấn công mạng hiệu quả sử dụng thuật toán học máy Support Vector Machine, nhằm nâng cao hiệu quả trong việc nhận diện các mối đe dọa mạng, từ đó bảo vệ dữ liệu và tài nguyên của tổ chức.

Với mục tiêu chính trên, đề tài sẽ có những mục tiêu cụ thể sau:

- Nghiên cứu cơ sở lý thuyết về tấn công mạng, malware.
- Nghiên cứu về thuật toán học máy nói chung và thuật toán Support Vector Machine nói riêng, các ưu điểm và nhược điểm của thuật toán.
- Nghiên cứu và thu thập tập dữ liệu liên quan đến tấn công mạng.
- Dựa vào tập dữ liệu, tiền xử lý, huấn luyện và tối ưu hóa và kiểm thử mô hình học máy SVM.

1.2. Đối tượng nghiên cứu

Đối tượng nghiên cứu là ứng dụng phát hiện tấn công mạng bằng học máy với thuật toán Support Vector Machine, cụ thể bao gồm:

- Tìm hiểu tổng quan về tấn công mạng, các giai đoạn tấn công, các loại tấn công và phương pháp phát hiện tấn công mạng.
- Tìm hiểu lý thuyết và nguyên lý hoạt động của malware và malware evasion.
- Tìm hiểu về hệ thống giám sát an ninh mạng, dữ liệu mạng và log.
- Nghiên cứu tổng quan về các thuật toán, và trọng tâm hơn về Support Vector Machine trong học máy.
- Nghiên cứu và đánh giá thuật toán học máy SVM.
- Đối tượng nghiên cứu chính là phát hiện tấn công mạng thông qua học máy sử dụng phương pháp SVM.

1.3. Phương pháp nghiên cứu

1.3.1. Phương pháp luận

Phương pháp luận là nền tảng của nghiên cứu khoa học, giúp định hướng và xác định cách tiếp cận phù hợp để giải quyết vấn đề nghiên cứu.

Đối với đề tài này, phương pháp luận bao gồm nghiên cứu tài liệu và phân tích các tài liệu, bài báo khoa học, và sách liên quan đến tấn công mạng, thuật toán học máy, và SVM.

Nghiên cứu lý thuyết tìm hiểu và phân tích các khái niệm, cơ sở lý thuyết về học máy, các loại tấn công mạng, và các phương pháp phát hiện tấn công.

Đồng thời, xây dựng mô hình dựa trên các kiến thức lý thuyết và dữ liệu thực tế, phát triển mô hình phát hiện tấn công mạng sử dụng thuật toán SVM.

1.3.2. Phương pháp đánh giá dựa trên cơ sở toán học

Phương pháp này tập trung vào việc sử dụng các công cụ và kỹ thuật toán học để đánh giá độ hiệu quả của mô hình SVM trong phát hiện tấn công mạng.

Phân tích và xử lý dữ liệu sử dụng các kỹ thuật tiền xử lý dữ liệu như chuẩn hóa, giảm chiều dữ liệu, và chia tách dữ liệu thành tập huấn luyện và tập kiểm tra.

Huấn luyện mô hình sử dụng thuật toán SVM để huấn luyện mô hình dựa trên tập dữ liệu đã được tiền xử lý, các tham số của mô hình sẽ được tối ưu hóa để đạt được hiệu quả tốt nhất.

Đánh giá hiệu quả mô hình sử dụng các chỉ số đánh giá như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (specificity), và F1-score để đánh giá hiệu quả của mô hình. Phân tích các chỉ số này giúp hiểu rõ hơn về hiệu suất của mô hình trong việc phát hiện tấn công mạng.

CHƯƠNG 2: TỔNG QUAN LÝ THUYẾT

2.1. Tổng quan về tấn công mạng

2.1.1. Khái niệm về tấn công mạng

"Tấn công mạng" (cyberattack) là hành vi có chủ đích và có tổ chức nhằm vào hệ thống thông tin, mạng máy tính, hoặc dịch vụ trực tuyến của một tổ chức hoặc cá nhân. Mục đích của các tấn công này có thể bao gồm đánh cắp thông tin quan trọng, phá hoại hoạt động bình thường của hệ thống, lây nhiễm các phần mềm độc hại, hoặc thậm chí kiểm soát từ xa các tài nguyên mạng.

Tấn công mạng thường được thực hiện bởi các hacker hoặc nhóm tội phạm mạng có kỹ năng về công nghệ thông tin. Các hình thức tấn công có thể khác nhau từ việc khai thác các lỗ hổng bảo mật trong phần mềm, tấn công vào cơ sở hạ tầng mạng (ví dụ như mạng LAN, WAN), đến việc tìm cách xâm nhập vào hệ thống bằng cách sử dụng các kỹ thuật như lừa đảo (phishing), thay đổi dữ liệu truyền tải (man-in-the-middle), hoặc thậm chí lây nhiễm các phần mềm độc hại.

Hậu quả của tấn công mạng có thể rất nghiêm trọng, bao gồm mất mát tài chính do phải chi trả tiền chuộc (nếu bị tấn công ransomware), giảm danh tiếng do dữ liệu bị lộ ra ngoài, mất quyền kiểm soát của hệ thống, và thậm chí là ảnh hưởng đến an ninh quốc gia nếu hệ thống quan trọng bị tấn công và kiểm soát.

Do đó, việc bảo vệ hệ thống mạng và phòng ngừa tấn công mạng là một trong những ưu tiên hàng đầu của các tổ chức và cơ quan chính phủ hiện nay.

2.1.2. Các giai đoạn tấn công mạng

a. *Trình sát (Reconnaissance)*

Trình sát là giai đoạn quan trọng trong quá trình chuẩn bị của kẻ tấn công mạng, trong đó họ tập trung vào việc thu thập thông tin về mục tiêu trước khi thực hiện cuộc tấn công. Mục đích chính của trình sát là có được một lượng thông tin lớn về mục tiêu để lên kế hoạch và triển khai các cuộc tấn công một cách hiệu quả và dễ dàng hơn. Phạm vi của trình sát có thể bao gồm các khách hàng, nhân viên, hoạt động, mạng và hệ thống của mục tiêu...

Có hai loại trình sát chính:

- **Trình sát thụ động:** Kẻ tấn công thu thập thông tin mà không cần tương tác trực tiếp với mục tiêu. Ví dụ như tìm kiếm thông tin từ các hồ sơ công khai hoặc các bản tin đã phát hành trên internet.
- **Trình sát chủ động:** Đây là việc tương tác trực tiếp với mục tiêu thông qua nhiều phương tiện khác nhau. Ví dụ như thực hiện các cuộc gọi để lấy thông tin kỹ thuật hoặc cá nhân từ mục tiêu.

b. *Quét (Scanning)*

Sau khi đã thu thập được thông tin từ giai đoạn trình sát, kẻ tấn công tiếp tục với giai đoạn quét mạng lưới thông tin của mục tiêu. Quá trình này có thể bao gồm các hoạt động sau:

1. Sử dụng các công cụ quét mạng: Kẻ tấn công có thể sử dụng các công cụ như trình quét cổng (port scanner) để kiểm tra các cổng mạng mà hệ thống mục tiêu đang mở và sẵn sàng cho kết nối.

2. Lập bản đồ mạng (network mapping): Đây là quá trình xây dựng sơ đồ chi tiết về cấu trúc mạng của mục tiêu, bao gồm các thiết bị mạng và cách chúng kết nối với nhau.

3. Quét bao quát: Kẻ tấn công cố gắng xác định các máy chủ, thiết bị mạng và các nguồn tài nguyên quan trọng khác trong hệ thống của mục tiêu.

4. Quét lỗ hổng (vulnerability scanning): Đây là việc kiểm tra các lỗ hổng bảo mật trên hệ thống mục tiêu. Kẻ tấn công tìm kiếm các lỗ hổng này để khai thác và xâm nhập vào hệ thống.

Khi đã có được thông tin chi tiết như tên máy tính, địa chỉ IP và các tài khoản người dùng, lỗ hổng bảo mật của hệ thống, kẻ tấn công sẽ sử dụng những thông tin này để bắt đầu triển khai các hành động tấn công mạng.

c. Truy cập (*Gaining Access*)

Truy cập là một giai đoạn then chốt trong quá trình tấn công mạng, khi kẻ tấn công cố gắng xâm nhập và có được quyền truy cập vào hệ thống điều hành hoặc các ứng dụng mạng của mục tiêu. Quá trình này diễn ra ở các cấp độ khác nhau:

1. Cấp hệ điều hành: Kẻ tấn công nhằm lấy quyền kiểm soát trên hệ điều hành của máy tính mục tiêu. Điều này cho phép họ thực hiện các hành động như bẻ khóa mật khẩu người dùng, cài đặt phần mềm gián điệp, hoặc thực hiện các lệnh điều khiển từ xa.

2. Cấp ứng dụng: Ngoài việc xâm nhập vào hệ điều hành, kẻ tấn công cũng cố gắng xâm nhập vào các ứng dụng mạng đang chạy trên hệ thống. Điều này có thể dẫn đến khai thác các lỗ hổng bảo mật trong các ứng dụng để đánh cắp dữ liệu hoặc thực hiện các cuộc tấn công khác.

3. Cấp mạng: Kẻ tấn công có thể tìm cách xâm nhập vào cơ sở hạ tầng mạng của mục tiêu, bao gồm cả việc chiếm đoạt kiểm soát các thiết bị mạng như router, switch để thực hiện các cuộc tấn công từ đó, hoặc thực hiện các cuộc tấn công từ xa như tấn công từ chối dịch vụ (DoS).

Khi đã có quyền truy cập, kẻ tấn công có thể nâng cấp quyền để thao túng toàn bộ hệ thống mục tiêu và khai thác thông tin như danh sách người dùng và mật khẩu để tiếp tục các hoạt động xâm nhập và tấn công. Các hành động này thường được thực hiện từ xa và được lên kế hoạch cẩn thận để tránh bị phát hiện sớm.

d. Duy trì truy cập (*Maintaining Access*)

Duy trì truy cập là giai đoạn kẻ tấn công mạng cố gắng duy trì quyền kiểm soát trên hệ thống mà họ đã xâm nhập.

Khi đã có quyền kiểm soát, kẻ tấn công sử dụng hệ thống này để triển khai các cuộc tấn công khác. Điều này có thể bao gồm cài đặt malware, tiến hành phishing hoặc tấn công mạng khác nhằm vào các mục tiêu mới.

Để duy trì quyền truy cập vào hệ thống mà không bị phát hiện và tránh bị đối thủ chiếm đoạt lại quyền sở hữu hệ thống, kẻ tấn công có thể triển khai các malware như

backdoors ẩn, rootkits để che giấu và tránh bị phát hiện, hoặc cài đặt trojan để duy trì quyền điều khiển từ xa. Nhờ vào các công cụ này, họ có thể quay lại và tiếp tục xâm nhập mà không gặp khó khăn.

Ngoài ra, khi đã có quyền sở hữu, kẻ tấn công có thể thực hiện mọi hành động với dữ liệu trên hệ thống, bao gồm đọc, sửa đổi hoặc xóa dữ liệu, tùy theo mục đích tấn công của họ.

Quá trình duy trì truy cập giúp kẻ tấn công duy trì và tiếp tục tấn công một cách hiệu quả, đồng thời cũng giúp họ giữ được quyền kiểm soát và bảo vệ quyền sở hữu tránh bị mục tiêu giành lại quyền sở hữu hệ thống.

e. Xóa dấu vết (Covering Tracks)

Kẻ tấn công thực hiện việc xóa các bản ghi trên máy chủ, hệ thống và các ứng dụng nhằm mục đích chủ yếu là che giấu hành vi tấn công và tiếp tục duy trì quyền truy cập vào hệ thống mà không bị nghi ngờ. Hành động này giúp kẻ tấn công tránh bị phát hiện sớm và làm khó cho các nhân viên an ninh mạng trong việc điều tra và phòng ngừa.

Bằng cách xóa các dấu vết và các bằng chứng liên quan đến bản thân, kẻ tấn công có thể tiếp tục tiến hành các hoạt động xâm nhập mạng và tấn công mà không gặp phải sự chú ý đáng kể từ phía các hệ thống giám sát an ninh. Điều này cho phép họ duy trì quyền kiểm soát và tiếp tục thao túng hệ thống một cách hiệu quả, từ đó có thể gây ra các tổn hại hoặc đánh cắp thông tin quan trọng mà không bị phát hiện.

2.1.3. Các kỹ thuật tấn công phổ biến

a. Tấn công từ chối dịch vụ DoS và DDoS

Tấn công từ chối dịch vụ (DoS) là một loại tấn công mạng nhằm mục đích làm gián đoạn hoặc ngừng hoạt động của một hệ thống, dịch vụ hoặc mạng, khiến cho người dùng không thể truy cập hoặc sử dụng dịch vụ đó. Tấn công DoS thường được thực hiện bằng cách gửi một lượng lớn yêu cầu hoặc dữ liệu đến hệ thống mục tiêu, khiến cho hệ thống bị quá tải và ngừng hoạt động.

Các phương pháp tấn công DoS phổ biến bao gồm:

1. Flooding Attacks: Gửi một lượng lớn dữ liệu hoặc yêu cầu để làm ngập hệ thống mục tiêu.
2. Buffer Overflow Attacks: Gửi dữ liệu vượt quá giới hạn của bộ nhớ đệm, gây lỗi hoặc ngừng hoạt động của hệ thống.
3. Resource Exhaustion Attacks: Sử dụng hết tài nguyên hệ thống như CPU, bộ nhớ hoặc băng thông.

Tấn công từ chối dịch vụ phân tán (DDoS) là một phiên bản nâng cao và nguy hiểm hơn của tấn công DoS. Trong tấn công DDoS, kẻ tấn công sử dụng nhiều thiết bị hoặc máy tính bị chiếm quyền điều khiển (thường được gọi là botnet) để gửi lượng lớn lưu lượng truy cập đến hệ thống mục tiêu từ nhiều nguồn khác nhau. Điều này làm cho việc phòng chống và ngăn chặn tấn công trở nên khó khăn hơn.

Các phương pháp tấn công DDoS phổ biến bao gồm:

1. Volume-Based Attacks: Tấn công làm ngập bằng thông của mục tiêu bằng lượng lớn lưu lượng.
2. Protocol Attacks: Tấn công làm cạn kiệt các tài nguyên của hệ thống như bộ nhớ hoặc CPU.
3. Application Layer Attacks: Tấn công vào các ứng dụng hoặc dịch vụ cụ thể để làm chúng ngừng hoạt động.

b. Tấn công Phishing

Tấn công phishing là một hình thức tấn công mạng thuộc dạng social engineering, nhằm đánh cắp thông tin nhạy cảm của người dùng như tên đăng nhập, mật khẩu, thông tin thẻ tín dụng và dữ liệu cá nhân khác. Thay vì khai thác tài nguyên hệ thống, tấn công phishing khai thác yếu tố con người bằng cách giả mạo các đơn vị đáng tin cậy như ngân hàng, trang web giao dịch trực tuyến, ví điện tử hoặc công ty thẻ tín dụng để lừa người dùng cung cấp thông tin nhạy cảm.

Có nhiều kỹ thuật mà tin tặc sử dụng để thực hiện một vụ tấn công Phishing.

1. Email Phishing: kẻ tấn công gửi email giả mạo từ các nguồn đáng tin cậy như ngân hàng, dịch vụ trực tuyến hoặc công ty nổi tiếng. Trong email chứa các liên kết đến các trang web giả mạo trông giống như trang web thật, yêu cầu người dùng nhập thông tin nhạy cảm.
2. Spear Phishing: là một dạng tấn công phishing được nhắm mục tiêu cụ thể đến một cá nhân hoặc tổ chức. Kẻ tấn công nghiên cứu nạn nhân để thu thập thông tin nhằm tạo một email hoặc tin nhắn có vẻ đáng tin cậy hơn đối với mục tiêu.
3. Whale Phishing: là một hình thức spear phishing nhưng nhắm vào các cá nhân cấp cao trong một tổ chức, chẳng hạn như giám đốc điều hành hoặc lãnh đạo doanh nghiệp.
4. Clone Phishing: kẻ tấn công sao chép một email hợp pháp đã được gửi trước đó và thay đổi một số chi tiết để chuyển hướng người nhận đến trang web giả mạo. Email giả mạo này có thể yêu cầu người dùng cập nhật thông tin hoặc xác minh tài khoản.
5. Vishing (Voice Phishing): kẻ tấn công sử dụng cuộc gọi điện thoại để lừa nạn nhân cung cấp thông tin nhạy cảm. Các cuộc gọi thường được giả mạo làm nhân viên hỗ trợ khách hàng hoặc đại diện từ các tổ chức uy tín.

c. Tấn công nghe lén Man-in-the-middle (MITM)

Tấn công Man-in-the-middle liên quan đến hành động nghe lén của kẻ tấn công. Cụ thể kẻ tấn công bí mật ngăn chặn lưu lượng mạng giữa hai bên để lắng nghe hoặc thậm chí có thể thay đổi thông tin trước khi chuyển cho bên nhận thông tin. Mục đích của tấn công MITM là đánh cắp thông tin nhạy cảm như thông tin đăng nhập, dữ liệu tài chính, hoặc bất kỳ dữ liệu cá nhân nào khác, hoặc thao túng thông tin truyền tải giữa hai bên.

Các loại tấn công MITM phổ biến bao gồm:

1. ARP Spoofing: ARP (Address Resolution Protocol) ánh xạ địa chỉ IP thành địa chỉ MAC trong mạng LAN. Khi một thiết bị cần biết địa chỉ MAC của thiết bị khác dựa trên địa chỉ IP, nó gửi yêu cầu ARP. Trong tấn công ARP Spoofing, kẻ tấn công giả mạo làm thiết bị khác bằng cách trả lời yêu cầu ARP với địa chỉ MAC của mình. Điều

này cho phép kẻ tấn công chặn và theo dõi lưu lượng dữ liệu giữa hai thiết bị, từ đó đánh cắp thông tin quan trọng.

2. DNS Spoofing: là tấn công giả mạo DNS, trong đó kẻ tấn công cố gắng đưa thông tin DNS bị hỏng vào bộ nhớ cache của máy chủ. Điều này nhằm đánh lừa nạn nhân truy cập vào một máy chủ giả mạo khi họ truy cập một tên miền. Kết quả là, nạn nhân gửi thông tin nhạy cảm đến máy chủ của kẻ tấn công, tưởng rằng đang gửi đến một nguồn đáng tin cậy.

3. Rogue Access Point: là hình thức tấn công kẻ tấn công thiết lập một điểm truy cập không dây giả mạo chẳng hạn như wifi mở không cần nhập mật khẩu. Các thiết bị với card mạng không dây thường tự động kết nối với điểm truy cập có tín hiệu mạnh nhất. Kẻ tấn công lợi dụng điều này để lừa các thiết bị gần đó kết nối với điểm truy cập của họ, cho phép kẻ tấn công kiểm soát và thao túng toàn bộ lưu lượng mạng của nạn nhân. Điều này đặc biệt nguy hiểm vì kẻ tấn công không cần phải ở trong mạng đáng tin cậy, chỉ cần ở gần nạn nhân.

d. Tấn công Botnet

Tấn công Botnet sử dụng một mạng lưới các thiết bị bị nhiễm phần mềm độc hại (gọi là "bot" hoặc "zombie") do kẻ tấn công điều khiển từ xa. Các thiết bị này có thể là máy tính, điện thoại thông minh, thiết bị IoT, hoặc bất kỳ thiết bị kết nối mạng nào khác. Hình thức tấn công này có nhiều chức năng khác nhau phục vụ cho mục đích của kẻ tấn công chẳng hạn như:

1. Tấn công từ chối dịch vụ phân tán DDoS: Sử dụng mạng botnet để gửi lượng lớn yêu cầu đến một máy chủ mục tiêu, làm quá tải và gây gián đoạn dịch vụ.
2. Gửi spam: Sử dụng các bot để gửi hàng loạt email spam hoặc lừa đảo.
3. Đánh cắp dữ liệu: Thu thập thông tin nhạy cảm như mật khẩu, thông tin thẻ tín dụng từ các thiết bị bị nhiễm botnet.
4. Phát tán phần mềm độc hại: Sử dụng botnet để phát tán phần mềm độc hại đến nhiều thiết bị khác.
5. Chiếm quyền điều khiển: Sử dụng các thiết bị bị nhiễm để thực hiện các hành vi tấn công khác, như tấn công mạng khác hoặc khai thác tiền điện tử.

e. Tấn công lỗ hổng Zero-day

Tấn công Zero-Day là một loại tấn công mạng lợi dụng lỗ hổng bảo mật chưa được phát hiện hoặc công bố rộng rãi. Những lỗ hổng này chưa có bản vá hoặc biện pháp khắc phục từ phía nhà phát triển phần mềm hoặc nhà cung cấp dịch vụ, khiến cho các hệ thống, ứng dụng và thiết bị trở nên dễ bị tấn công.

2.1.4. Lý thuyết hoạt động của malware

Trong giai đoạn truy cập và duy trì truy cập của quá trình tấn công mạng, kẻ tấn công sử dụng nhiều kỹ thuật và công cụ khác nhau để vượt qua các biện pháp bảo mật, giành quyền và duy trì quyền kiểm soát hệ thống. Malware thường được sử dụng trong các giai đoạn này để hỗ trợ việc truy cập, kiểm soát hệ thống và duy trì sự truy cập. Vậy malware là gì mà nó có thể thực hiện những hành vi độc hại này?

a. Khái niệm malware

Malware là bất cứ phần mềm độc hại nào được thiết kế với mục đích xấu liên quan đến thông tin nhạy cảm, thiết bị hoặc hệ thống của người dùng. Các chương trình này có thể thực hiện nhiều chức năng bao gồm ăn cắp, mã hóa hoặc xóa dữ liệu nhạy cảm, thay đổi hoặc chiếm đoạt các chức năng tính toán quan trọng và giám sát hoạt động thiết bị của người dùng mà không được sự cho phép của họ.

b. Các loại malware

Có nhiều loại phần mềm độc hại khác nhau chứa các đặc điểm và đặc tính riêng. Sau đây là các loại malware phổ biến:

- **Virus:** có khả năng tự sao chép và lây lan bằng cách gắn vào các tệp tin hoặc chương trình khác. Khi tệp tin nhiễm virus được chạy, virus tạo ra bản sao của mình và lây lan sang các file khác, đồng thời có thể thực hiện các hành động độc hại như xóa hoặc chỉnh sửa tệp tin và làm hỏng hệ thống.
- **Worm:** có khả năng tự sao chép và lây lan qua mạng mà không cần sự tương tác của người dùng. Worm thường khai thác các lỗ hổng bảo mật để truy cập vào hệ thống và lan truyền sang các thiết bị khác. Nó có thể xóa hoặc chỉnh sửa tệp tin và thêm phần mềm độc hại, như cài đặt backdoor để kẻ tấn công xâm nhập. Đôi khi, mục đích của worm đơn giản chỉ là tự sao chép nhiều lần để gây lãng phí tài nguyên hệ thống, băng thông, hoặc dung lượng ổ cứng.
- **Spyware:** thu thập thông tin cá nhân hoặc tổ chức mà không có sự cho phép. Nó có thể được cài đặt trên máy tính, điện thoại thông minh, hoặc máy tính bảng để theo dõi hoạt động trực tuyến, đánh cắp dữ liệu cá nhân, theo dõi vị trí, và gửi tin nhắn trái phép. Spyware thường sử dụng các kỹ thuật như keylogging, ghi màn hình, và thu thập dữ liệu, làm cho nó khó phát hiện và loại bỏ mà không có công cụ chuyên dụng.
- **Ransomware:** mã hóa các tệp tin trên thiết bị bị nhiễm và yêu cầu thanh toán để lấy lại khóa giải mã. Kẻ tấn công thường sử dụng email phishing, trang web độc hại, hoặc cập nhật phần mềm giả mạo để lừa người dùng cài đặt ransomware. Một khi cài đặt xong, ransomware rất khó phát hiện và loại bỏ, và có thể đe dọa công khai hoặc bán thông tin bị đánh cắp nếu nạn nhân không trả tiền chuộc.
- **Rootkit:** cung cấp quyền truy cập và kiểm soát cấp cao nhất trên hệ thống mà không bị phát hiện. Tin tặc sử dụng rootkit để duy trì quyền truy cập trái phép, ẩn các tệp tin, tiến trình, và registry key, giúp hoạt động ngầm mà không gây nghi ngờ. Điều này cho phép kẻ tấn công kiểm soát toàn bộ hệ thống, thực hiện các lệnh, giám sát hoạt động của người dùng, đánh cắp thông tin, và cài đặt thêm phần mềm độc hại.

2.1.5. Lý thuyết hoạt động của malware evasion

Tin tặc thường sử dụng các kỹ thuật né tránh tinh vi và kết hợp chúng để vượt qua các biện pháp bảo mật thông thường đồng thời che giấu sự hiện diện của mình sau khi xâm nhập thành công. Dựa trên bài viết của trang web Cyfirma [1], các chiến lược né tránh cần phải kể đến là Kỹ thuật né tránh dựa trên chữ ký (Signature-based Evasion Techniques), Kỹ thuật né tránh dựa trên hành vi (Behaviour-based Evasion Techniques), Kỹ thuật chống phân tích (Anti-analysis Techniques), Kỹ thuật tiêm quy trình (Process Injection Techniques) và Kỹ thuật phần mềm độc hại không tệp tin (Fileless Malware Techniques).

a. Kỹ thuật né tránh dựa trên chữ ký (Signature-based Techniques)

Chữ ký tập tin là chuỗi ký tự đặc biệt được tạo dựa vào thông tin file như tên, loại file, nội dung file. Các phương pháp phát hiện mã độc truyền thống khi phát hiện được phần mềm độc hại sẽ lưu chữ ký hoặc khuôn mẫu của chúng để lần sau phát hiện nhanh chóng và hiệu quả hơn. Tuy nhiên kẻ tạo mã độc có thể giải quyết vấn đề đó bằng cách tinh chỉnh các đặc điểm của mã độc từ đó thay đổi hoặc che giấu chữ ký.

i. Mã độc Đa hình (Polymorphic Malware) và mã độc Siêu hình (Metamorphic Malware)

Mã độc đa hình và mã độc siêu hình đều là những loại phần mềm độc hại có khả năng né tránh các biện pháp bảo mật truyền thống bằng cách thay đổi hình dạng của chúng. Tuy nhiên cách thức thực hiện việc này lại có sự khác biệt đáng kể.

Mã độc đa hình (Polymorphic malware) mặc dù thay đổi liên tục sau mỗi lần lặp nhưng giữ lại phần mã chính để thực hiện chức năng độc hại. Phần mã còn lại liên tục biến đổi để tránh sự phát hiện của chương trình antivirus dựa trên chữ ký.

Mã độc siêu hình (Metamorphic malware) tinh vi và phức tạp hơn nhiều. Trong khi mã độc đa hình vẫn có thể bị phát hiện do mã chính không thay đổi, mã siêu hình tái cấu trúc toàn bộ mã của chúng sau mỗi lần lặp lại. Nó thêm vào các yếu tố như chèn mã giả, mã rác và sắp xếp lại các chức năng, làm cho mã trông khác biệt so với các thể hệ trước, gây khó khăn hơn trong việc phát hiện phần mềm độc hại.

ii. Mã hóa tập tin (File Encryption)

Công việc mã hóa dữ liệu thường được sử dụng để tránh bị lộ hay bị đánh cắp dữ liệu bởi các tin tặc. Ngược lại, tội phạm mạng có thể dùng việc mã hóa để mã hóa mã độc hoặc các thành phần của nó nhằm che giấu mục đích thực sự và né tránh sự phát hiện của phần mềm bảo mật. Kỹ thuật này nhằm ngăn chặn việc bị phân tích dễ dàng. Khi mã độc được mã hóa, chữ ký của nó bị ẩn bên trong nội dung đã mã hóa, khiến phương pháp phát hiện dựa trên chữ ký trở nên kém hiệu quả.

iii. Công cụ nén (Packer) và mã hóa code (Crypter)

Packer và Crypter là hai công cụ hỗ trợ trong mã độc để tránh bị phát hiện dựa trên chữ ký. Packers nén và mã hóa mã độc, tạo ra tệp thực thi nén mới yêu cầu quy trình cụ thể để giải nén. Phần mềm độc hại được nén sẽ tự động giải nén khi thực thi, thêm một lớp bảo mật để tránh bị phát hiện bởi các phần mềm antivirus truyền thống. Crypter tập trung vào mã hóa mã độc, cung cấp thêm một lớp bảo vệ tránh bị nhận diện. Mặc dù hai kỹ thuật này không khó để vượt qua bằng cách sử dụng công cụ unpack và dịch ngược, phần mềm độc hại sử dụng chúng thường tạo ra các phiên bản động và thay đổi liên tục, khiến hệ thống bảo mật khó phát hiện và ngăn chặn hơn.

iv. Làm rối mã (Code Obfuscation)

Kỹ thuật làm rối mã là việc chỉnh sửa, thay đổi cấu trúc, logic, và cách trình bày của mã sao cho tệp thực thi mã độc không thể hiểu, diễn dịch hay thực thi bởi một bên thứ ba nào đó. Hành động này không làm ảnh hưởng hay hư hại đến mã độc và vẫn có thể thực thi được các chức năng độc hại của chúng. Kỹ thuật này không chỉ ngăn chặn việc phát hiện dựa trên chữ ký mà thậm chí còn có thể chống dịch ngược và phân tích bởi sự phức tạp và rối rắm của mã độc.

*b. Kỹ thuật né tránh dựa trên hành vi (Behaviour-based Techniques)**i. Phát hiện môi trường Sandbox (Sandbox Detection)*

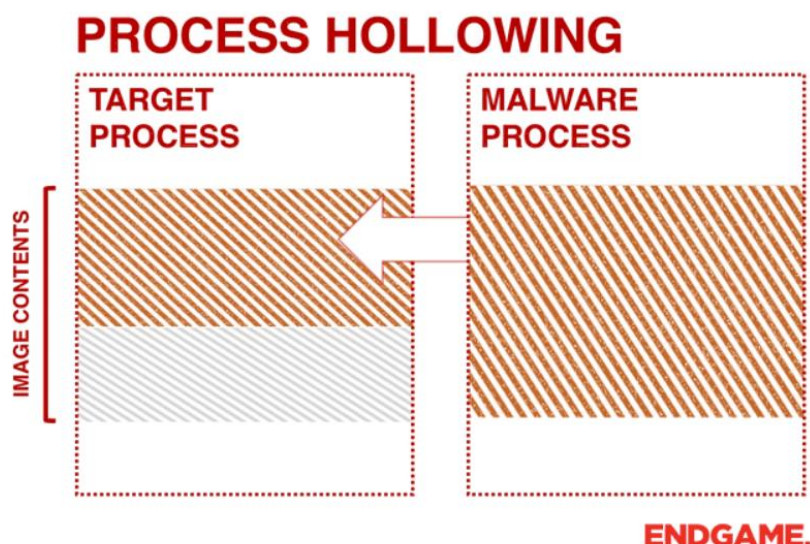
Thực thi mã độc trong môi trường kiểm soát là một cách phân tích hành vi để phát hiện phần mềm độc hại. Tuy nhiên, kẻ tạo mã độc có thể sử dụng kỹ thuật né tránh để phát hiện liệu phần mềm có đang chạy trong môi trường kiểm soát như máy ảo hoặc sandbox hay không. Nếu phát hiện, mã độc có thể tùy chỉnh hành vi, như không thực thi hoặc thực hiện các hành động vô hại, để tránh bị phát hiện. Phần mềm độc hại có khả năng nhận diện các đặc điểm của môi trường kiểm soát, như đường dẫn tệp cụ thể, mục đăng ký, hoặc cấu hình mạng. Khi phát hiện đang bị kiểm soát, mã độc có thể trì hoãn hành động độc hại, duy trì trạng thái ngủ đông, hoặc giả mạo các hành động vô hại cho đến khi chạy trong môi trường thực tế. Điều này làm chậm quá trình phát hiện và giúp mã độc tránh bị phát hiện, tăng khả năng gây hại khi thực thi trong môi trường người dùng thực tế. Một số mã độc còn tự cập nhật và cải tiến khả năng né tránh dựa trên các phân tích trước đó, làm cho việc phát hiện và ngăn chặn trở nên khó khăn hơn.

ii. Kiểm tra môi trường (Environment Checking)

Một kỹ thuật nâng cao hơn trong việc né tránh là mã độc kiểm tra môi trường xung quanh để đánh giá bối cảnh mà nó đang chạy, từ đó điều chỉnh hành vi cho phù hợp. Mã độc có thể giám sát các yếu tố như cấu hình hệ thống, cài đặt phần mềm, thiết lập mạng, và công cụ bảo mật để xác định liệu nó có đang chạy trong môi trường ảo hóa hay không. Ngoài ra, mã độc cũng xem xét các phần mềm đã được cài đặt, đặc biệt là các công cụ bảo mật và chương trình phân tích. Nó kiểm tra cấu hình mạng để phát hiện các thiết lập bất thường có thể liên quan đến môi trường kiểm tra hoặc phân tích, và nhận diện các công cụ bảo mật hoặc dấu hiệu của hoạt động phân tích như các đường dẫn tệp cụ thể hoặc các mục đăng ký. Dựa trên những đánh giá này, giống như kỹ thuật né tránh môi trường sandbox, mã độc có thể thay đổi hành vi để trở nên ít đáng ngờ hơn, tạm hoãn lại hành động độc hại cùng với khả năng học hỏi từ các lần phân tích trước để cải thiện và tinh chỉnh các kỹ thuật né tránh, làm cho việc phát hiện và ngăn chặn trở nên khó khăn hơn.

iii. Rỗng hóa tiến trình (Process Hollowing)

Rỗng hóa tiến trình là một kỹ thuật né tránh dựa trên hành vi trong đó một tiến trình hợp pháp bị mã độc thao túng để thay thế mã hợp pháp của nó bằng mã độc hại. Đây cũng là một trong những kỹ thuật tiêm tiến trình (Process Injection Techniques). Phần mềm độc hại sử dụng kỹ thuật này thường tạo ra một quá trình ở trạng thái bị treo, sau đó xóa bộ nhớ của tiến trình này tạo ra một tiến trình rỗng từ đó mã độc sẽ thay thế phần bị xóa. Kỹ thuật này được sử dụng để thực thi mã độc trong một tiến trình có vẻ bình thường, khiến việc phát hiện trở nên khó khăn hơn.



Hình 2.1: Kỹ thuật né tránh dựa trên hành vi – Rỗng hóa tiến trình [2]

c. Kỹ thuật chống phân tích (Anti-analysis Techniques)

i. Làm rối mã (Code Obfuscation)

Như đã đề cập ở kỹ thuật né tránh dựa trên chữ ký, kỹ thuật làm rối mã khiến cho mã độc không thể phân tích hoặc cản trở quá trình dịch ngược, gây khó khăn cho các nhà phân tích, các nhà nghiên cứu bảo mật trong công việc tìm hiểu và giải mã các chức năng và mục đích thực sự của phần mềm độc hại.

ii. Kỹ thuật chống gỡ lỗi (Anti-debugging Techniques)

Các tác giả malware sử dụng các kỹ thuật chống gỡ lỗi để phát hiện và ngăn chặn các nỗ lực phân tích từ các công cụ debug. Khi phát hiện mình đang chạy trong môi trường debug, mã độc có thể điều chỉnh hành vi hoặc gây gián đoạn trong quá trình thực thi. Các kỹ thuật chống debug có thể bao gồm kiểm tra các biến môi trường, registry, hay giám sát các hành vi của hệ thống để nhận diện các dấu hiệu của việc gỡ lỗi như các lời gọi API đặc specific hay lệnh ngắt gỡ lỗi (breakpoints).

iii. Phát hiện máy ảo (Anti-VM Detection)

Tương tự như kỹ thuật phát hiện môi trường sandbox, mã độc sẽ cố gắng phát hiện xem liệu nó có đang chạy trong một môi trường máy ảo hay không. Nếu phát hiện ra máy ảo, phần mềm độc hại có thể thay đổi hành vi hoặc đơn giản là từ chối hoạt động, gây khó khăn đáng kể cho các nhà phân tích.

d. Kỹ thuật tiêm tiến trình (Process Injection Techniques)

i. Tiêm luồng (Thread Injection)

Thread Injection là một kỹ thuật tấn công trong đó mã độc tạo ra một luồng mới bên trong một tiến trình hợp lệ và tiêm payload vào luồng đó. Nhờ hoạt động trong ngữ cảnh của một tiến trình đáng tin cậy, mã độc có thể tránh bị phát hiện bởi các cơ chế giám sát hoạt động ở mức tiến trình. Bằng cách này, mã độc lợi dụng tài nguyên và quyền hạn của tiến trình hợp lệ để thực hiện các hành động độc hại. Kỹ thuật này làm cho luồng được tiêm mô phỏng hành vi hợp lệ, khiến các hệ thống bảo mật gặp khó khăn trong

việc phân biệt luồng độc hại với các hoạt động chính thống của tiến trình chủ, và do đó dễ dàng né tránh các phương pháp phát hiện dựa trên chữ ký và hành vi.

ii. Kỹ thuật cấp phát/ghi bộ nhớ (Memory Allocation/Write Technique)

Memory Allocation/Write Techniques là những phương pháp mà malware sử dụng để cấp phát bộ nhớ động và viết payload của chúng vào không gian địa chỉ của một tiến trình hợp lệ. Kỹ thuật này giúp malware thực thi mã độc mà không bị phát hiện, vì nó hoạt động trong ngữ cảnh của tiến trình đáng tin cậy. Về mặt lập trình, cấp phát bộ nhớ sử dụng các hàm như malloc, calloc, hoặc new giúp ứng dụng quản lý bộ nhớ một cách linh hoạt. Nhưng đối với các kẻ tấn công, họ lợi dụng các kỹ thuật như VirtualAllocEx và WriteProcessMemory trên Windows để tiêm và thực thi mã độc một cách tinh vi. [2]

e. Kỹ thuật mã độc không dấu vết (Fileless Malware Techniques)

i. Thực thi trong bộ nhớ (Memory-based Execution)

Memory-based Execution là một kỹ thuật không dấu vết trong malware, trong đó mã độc được thực thi trực tiếp trong bộ nhớ của hệ thống, mà không cần lưu trữ mã độc vào đĩa cứng. Kỹ thuật này tận dụng các lỗ hổng hoặc các công cụ hệ thống hợp pháp để tiêm và thực thi payload độc hại trong bộ nhớ, vượt qua các phương pháp phát hiện dựa trên tập tin. Kỹ thuật này tập trung vào khai thác các yếu điểm của hệ thống và hoạt động trong bộ nhớ, giúp malware hoạt động một cách âm thầm và tránh các dấu vết pháp lý truyền thống.

ii. Sử dụng công cụ có sẵn (Living-off-the-land Techniques - LOLbins)

Kỹ thuật Living-off-the-land, hay LOLbins, là phương pháp mà tin tặc sử dụng các công cụ và tiện ích có sẵn trong hệ thống để thực hiện các hành động độc hại. Thay vì đưa mã độc từ bên ngoài vào hệ thống, tin tặc tận dụng các chương trình đáng tin cậy đã được cài đặt sẵn trong hệ điều hành như Powershell, WMI, hoặc các lệnh tiện ích khác, mã độc có thể né tránh các hệ thống bảo mật với cơ chế không tập tin của nó.

iii. Tấn công dựa trên registry (Registry-based Attacks)

Tấn công dựa trên registry là kỹ thuật mà tin tặc lợi dụng Windows Registry để thực hiện các hành vi độc hại. Windows Registry là một cơ sở dữ liệu hệ thống lưu trữ các thiết lập cấu hình và tùy chọn cho hệ điều hành Windows và các ứng dụng. Bằng cách thao tác với registry, tin tặc có thể thay đổi hành vi của hệ thống và các chương trình theo ý muốn của chúng.

iv. Macro tài liệu (Document Macros)

Macro tài liệu là các đoạn mã thực thi tự động trong các tài liệu Microsoft Office như Word, Excel, và PowerPoint, được thiết kế để tự động hóa các tác vụ lặp lại. Tuy nhiên, kẻ tấn công có thể lợi dụng macros để phát tán mã độc, đánh cắp thông tin, hoặc thực hiện các hành vi độc hại khác. Khi người dùng mở tài liệu chứa macros độc hại, mã sẽ được kích hoạt, tải và thực thi payload độc hại trực tiếp trong bộ nhớ, tránh được các phương pháp phát hiện truyền thống. Kỹ thuật này thuộc loại fileless malware, lợi dụng sự tin tưởng của người dùng vào các định dạng tài liệu phổ biến để thực hiện hành vi độc hại mà không cần các tệp thực thi độc lập.

2.1.6. Các phương pháp phát hiện tấn công mạng

Hiện nay, có nhiều phương pháp và công nghệ để phát hiện tấn công mạng. Các phương pháp phổ biến cần kể đến là:

- Phát hiện chữ ký: dựa vào việc so sánh các dấu hiệu hoặc mẫu đã biết của các cuộc tấn công với dữ liệu mạng hiện tại. Các hệ thống này sử dụng cơ sở dữ liệu chữ ký để nhận diện các cuộc tấn công đã biết trước.
- Phát hiện hành vi: theo dõi và phân tích hành vi và hoạt động của người dùng và thiết bị trong mạng để tìm kiếm các hành vi không bình thường hoặc không phù hợp với mô hình hoạt động bình thường của hệ thống.
- Phát hiện dữ liệu bất thường: nhận diện các biểu hiện không bình thường trong dữ liệu mạng bằng cách phân tích các mẫu dữ liệu để tìm ra các hành vi bất thường so với tiêu chuẩn thống kê, sử dụng các quy tắc hoặc ngưỡng nhất định để xác định hoạt động bất thường, hoặc sử dụng các thuật toán học máy để phân tích dữ liệu lưu lượng mạng thực tế.
- Phân tích luồng dữ liệu: giám sát và phân tích các luồng dữ liệu mạng để nhận diện các mẫu hoạt động bất thường hoặc các tấn công
- Phân tích giao thức: phân tích các giao thức mạng để nhận diện các hoạt động không phù hợp hoặc các lỗ hổng giao thức có thể bị khai thác để tấn công.
- Phân tích dữ liệu và đánh giá rủi ro: sử dụng các công cụ phân tích dữ liệu để đánh giá và dự đoán rủi ro bảo mật của hệ thống dựa trên dữ liệu từ nhiều nguồn khác nhau trong hệ thống mạng.

2.2. Hệ thống giám sát phát hiện xâm nhập mạng IDS (Intrusion Detection System)

Ngày nay, an toàn mạng đang ngày càng được phát triển, đồng thời các tội phạm mạng cũng tìm cách đối phó với các biện pháp bảo mật. Các phương pháp phát hiện trước đây dần trở nên lạc hậu và không còn hiệu quả, và người dùng bình thường không có đủ kiến thức và kỹ năng chuyên sâu để bảo vệ hệ thống một cách thủ công. Vì vậy, các công cụ giám sát hệ thống tự động IDS trở thành giải pháp hiệu quả để phát hiện các cuộc tấn công mạng ngày càng tinh vi hơn.

2.2.1. Khái niệm IDS

IDS là một thiết bị hoặc một phần mềm ứng dụng giám sát lưu lượng mạng để phát hiện các hoạt động bất thường hoặc độc hại, hoặc các vi phạm chính sách [3], từ đó gửi cảnh báo phát hiện. Bất cứ hành vi độc hại hoặc vi phạm chính sách nào xảy ra, sẽ được gửi cho người quản trị hoặc được lưu trữ để cung cấp dữ liệu bảo mật cho hệ thống quản lý thông tin và sự kiện bảo mật SIEM [4]. SIEM là một hệ thống được sử dụng để tổng hợp và phân tích các thông tin và sự kiện bảo mật từ các nguồn khác nhau trong mạng hoặc hệ thống, giúp nhận diện và phản ứng với các mối đe dọa bảo mật một cách hiệu quả và nhanh chóng. IDS thường là một phần của hệ thống SIEM đóng vai trò phát hiện và cảnh báo.

2.2.2. Các loại IDS

IDS được thiết kế để có thể hoạt động trên các hệ thống khác nhau. Vì vậy, IDS có hai loại:

- Host-based IDS (HIDS) là hệ thống phát hiện xâm nhập được cài đặt trực tiếp trên máy chủ hoặc thiết bị đầu cuối để giám sát và phân tích các hoạt động đáng ngờ hoặc bất thường từ cả bên ngoài lẫn bên trong.

Mặc dù HIDS có thể giám sát lưu lượng mạng và các quy trình đang chạy, nó chỉ hoạt động trên từng máy chủ cụ thể và không phát hiện được các cuộc dò quét mạng hay tấn công DoS một cách hiệu quả. HIDS cũng yêu cầu nhiều tài nguyên để hoạt động và không thể cung cấp tầm nhìn tổng thể vào môi trường mạng. Host-based IDS có khả năng theo dõi lưu lượng mạng vào và ra từ máy, quan sát các quá trình đang chạy và kiểm tra nhật ký hệ thống. Tuy nhiên, tầm nhìn của nó chỉ giới hạn trong máy chủ được cài đặt, làm giảm bớt ngữ cảnh có sẵn cho việc ra quyết định. [4]

- Network-based IDS (NIDS) được thiết kế để giám sát và phân tích lưu lượng của toàn bộ hệ thống mạng nhằm phát hiện các hoạt động đáng ngờ hoặc có khả năng gây hại. NIDS hoạt động bằng cách theo dõi luồng dữ liệu trong mạng và so sánh với các mẫu đã biết về các cuộc tấn công hoặc hành vi bất thường, từ đó đưa ra cảnh báo cho quản trị viên mạng hoặc tự động thực hiện các hành động phản ứng. Nó sử dụng các bộ dò và bộ cảm biến được cài đặt trên toàn mạng để theo dõi lưu lượng và tìm kiếm các mẫu hoặc dấu hiệu định trước.

Mặc dù NIDS có khả năng quản lý một số lượng lớn host và cài đặt đơn giản, nó có thể gặp phải báo động giả và không thể phân tích các gói tin đã được mã hóa. Ngoài ra, NIDS cần được cập nhật thường xuyên với các chữ ký mới nhất để đảm bảo tính an toàn và không thể xác định chắc chắn liệu một cuộc tấn công đã thành công hay chưa. Tuy có tầm nhìn rộng hơn về lưu lượng mạng, NIDS thiếu khả năng giám sát chi tiết bên trong các điểm cuối mà nó bảo vệ.

2.2.3. Phương pháp phát hiện của IDS

Hai phương pháp chính mà IDS sử dụng là phương pháp phát hiện dựa vào chữ ký và phát hiện dựa vào dữ liệu bất thường. Hai phương pháp này đã được đề cập ở phần 2.1.6. về các phương pháp phát hiện tấn công mạng.

- Phát hiện dựa trên chữ ký: IDS phát hiện mối đe dọa dựa vào danh sách dấu hiệu của sự xâm nhập (IOC). IOC là các dấu hiệu, bằng chứng hoặc thông tin cụ thể được sử dụng để phát hiện các hành vi, sự kiện hoặc hiện tượng liên quan đến các cuộc tấn công mạng. Các IOC có thể bao gồm các dấu hiệu khác nhau của hành vi độc hại, từ tập tin, địa chỉ IP, hash (chữ ký) của tập tin, đến các hành vi cụ thể trên mạng và hệ thống. [3] Nếu phần mềm độc hại hoặc nội dung độc hại khác được nhận dạng, chữ ký của chúng sẽ được thêm vào danh sách lưu trữ của IDS. Điều này cho phép IDS phát hiện mối đe dọa một cách chính xác vì tất cả các báo cáo đều được tạo ra dựa trên thông tin đã biết. Tuy nhiên, nhược điểm của phương pháp này là không thể nhận diện các lỗ hổng Zero-day.

- Phát hiện dựa trên dữ liệu bất thường: ngược lại, phương pháp này không dựa vào thông tin đã biết mà tập trung vào việc cảnh báo các hành vi đáng ngờ dựa vào tiêu chuẩn mô hình. Biện pháp này xây dựng một mô hình về hành vi được cho là bình thường của hệ thống. Các hành vi xảy ra trong tương lai sẽ được so sánh với mô hình này, từ đó bất kỳ sự bất thường nào, kể cả những mối đe dọa chưa được biết đến đều được đánh dấu là nguy cơ tiềm ẩn và tạo ra cảnh báo.

Tuy nhiên phương pháp dựa trên dữ liệu bất thường cũng có khả năng gây báo động sai, đòi hỏi nhiều thời gian và tài nguyên để xây dựng một hệ thống cân bằng giữa báo động sai và bỏ sót phát hiện. Mặt khác, phát hiện dựa trên sự bất thường có thể phát hiện lỗ hổng Zero-day, trong khi biện pháp phát hiện dựa vào chữ ký chỉ giới hạn trong các mối đe dọa đã biết. Vì vậy, cả hai phương pháp này thường được kết hợp với nhau để tối ưu hóa hiệu quả phát hiện trong các hệ thống bảo mật. [3]

2.2.4. Nhược điểm của hệ thống giám sát IDS truyền thống

Hệ thống giám sát IDS là công cụ quan trọng để phát hiện và cảnh báo các cuộc tấn công mạng. Tuy nhiên, IDS truyền thống có những nhược điểm. Thứ nhất, IDS truyền thống dựa vào chữ ký, làm cho chúng kém hiệu quả trong việc phát hiện các mối đe dọa mới hoặc biến thể chưa được biết đến, vì chúng dựa vào các quy tắc tĩnh và mẫu đã biết, có thể bỏ sót hành vi tinh vi. Thứ hai, với chức năng phát hiện dựa trên dữ liệu bất thường, IDS truyền thống có thể gặp tỷ lệ cảnh báo sai cao, đặc biệt trong môi trường mạng phức tạp. Vì vậy, người tạo mô hình tiêu chuẩn cho phương pháp này thường gặp khó khăn trong việc cân bằng giữa báo động sai và bỏ sót phát hiện.

Hệ thống IDS nếu được ứng dụng học máy sẽ khắc phục hiệu quả những nhược điểm của IDS truyền thống bằng cách học từ dữ liệu để phát hiện các mối đe dọa mới và các cuộc tấn công biến thể mà không cần dựa vào các chữ ký đã biết. Nhờ vào khả năng phân tích và nhận diện các mẫu hành vi phức tạp, hệ thống này có thể phát hiện các hành vi tinh vi từ kẻ tấn công, bất kể các kỹ thuật né tránh, và tự động thích ứng với các thay đổi trong môi trường mạng. Hơn nữa, sử dụng các thuật toán tối ưu hóa, IDS học máy có thể giảm thiểu tỷ lệ cảnh báo sai bằng cách cải thiện độ chính xác trong phân tích và phát hiện, đồng thời cân bằng tốt hơn giữa việc báo động sai và bỏ sót phát hiện. Điều này làm cho IDS ứng dụng học máy trở thành một công cụ mạnh mẽ và linh hoạt, đáp ứng tốt hơn các yêu cầu bảo mật trong môi trường mạng phức tạp và không ngừng thay đổi.

2.3. Tổng quan về học máy có giám sát

2.3.1. Khái niệm học máy có giám sát

Học máy là hệ thống học cách đưa ra dự đoán bằng cách nghiên cứu dữ liệu để khám phá các kết nối và mối tương quan, thay vì yêu cầu nhà lập trình viết thủ công các chỉ dẫn. Các hệ thống học máy thường đưa ra dự đoán chính xác hơn so với các chương trình truyền thống. Một mô hình học máy có giám sát có thể tìm ra các kết nối giữa dữ liệu mà con người khó nhận ra và dần dần cải thiện độ chính xác của các dự đoán.

Học máy có giám sát (Supervised Learning) là phương pháp phổ biến trong học máy, nơi mô hình học từ một tập dữ liệu đã được gán nhãn. Trong quá trình học, mô hình được cung cấp dữ liệu đầu vào và đầu ra mong muốn, với mục tiêu là học mối quan hệ giữa chúng để dự đoán chính xác nhãn của dữ liệu mới. Ví dụ, nó có thể dự đoán thời tiết hoặc phân loại thư điện tử thành thư rác (spam).

Trong học máy có giám sát, có nhiều thuật toán được áp dụng cho việc học của hệ thống như Hồi quy tuyến tính (Linear Regression), Hồi quy logistic (Logistic Regression), Cây quyết định (Decision Trees), Naive Bayes và thuật toán SVM. Mỗi thuật toán có ưu điểm và hạn chế riêng, và sự lựa chọn giữa chúng phụ thuộc vào đặc điểm của bài toán cụ thể, loại dữ liệu có sẵn và mục tiêu cuối cùng của dự án.

2.3.2. Thuật toán Hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính là một thuật toán cơ bản trong học máy có giám sát, thường được sử dụng để dự đoán một giá trị số (đầu ra) dựa trên một hoặc nhiều biến độc lập (đầu vào). Mục tiêu của hồi quy tuyến tính là tìm ra một đường thẳng (trong trường hợp đơn biến) hoặc một mặt phẳng (trong trường hợp đa biến) phù hợp nhất với dữ liệu.

Phương trình tổng quát của hồi quy tuyến tính đơn giản có dạng:

$$y = \beta_0 + \beta_1 x$$

Trong đó:

- y là biến phụ thuộc,
- x là biến độc lập,
- β_0 là hệ số chặn, và β_1 là hệ số dốc.

Trong hồi quy tuyến tính đa biến, phương trình mở rộng thành:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Thuật toán này sử dụng phương pháp bình phương tối thiểu (least squares) để tìm ra giá trị tốt nhất cho β_0 và β_1 sao cho sai số giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất.

2.3.3. Thuật toán Hồi quy logistic (Logistic Regression)

Hồi quy logistic là một thuật toán học máy có giám sát, mặc dù có tên gọi là hồi quy nhưng thực chất lại được sử dụng chủ yếu cho các bài toán phân loại nhị phân (binary classification). Thay vì dự đoán một giá trị liên tục, hồi quy logistic dự đoán xác suất của một điểm dữ liệu thuộc về một trong hai lớp.

Phương trình tổng quát của hồi quy logistic có dạng:

$$\log(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Trong đó: p là xác suất của điểm dữ liệu thuộc về lớp dương, và x là biến độc lập. Xác suất p được tính bằng cách sử dụng hàm sigmoid:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Trong hồi quy logistic đa biến, phương trình mở rộng thành:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

Thuật toán này sử dụng phương pháp tối đa hóa hàm hợp lý (maximum likelihood estimation) để tìm ra giá trị tốt nhất cho các hệ số β_0 và β_1 .

2.3.4. Thuật toán Cây quyết định (Decision Trees)

Cây quyết định là một thuật toán học máy có giám sát được sử dụng cho cả bài toán phân loại và hồi quy. Thuật toán này sử dụng một cấu trúc dạng cây, trong đó mỗi nút trong cây đại diện cho một quyết định dựa trên giá trị của một biến đầu vào, và các nhánh của nút đó đại diện cho các kết quả của quyết định đó.

Cây quyết định bắt đầu từ gốc cây và chia tách dữ liệu thành các nhóm nhỏ hơn tại mỗi nút dựa trên các tiêu chí phân tách tốt nhất, như độ lợi thông tin (information gain) hoặc Gini impurity. Quá trình này được lặp lại đệ quy cho đến khi mỗi nút lá chứa các điểm dữ liệu của cùng một lớp (trong trường hợp phân loại) hoặc có giá trị đầu ra gần nhau (trong trường hợp hồi quy).

2.3.5. Thuật toán Naive Bayes

Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes với giả định rằng các đặc trưng của đầu vào là độc lập với nhau. Mặc dù giả định này thường không đúng trong thực tế, Naive Bayes vẫn hoạt động tốt cho nhiều bài toán thực tế.

Định lý Bayes phát biểu rằng:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Trong đó:

- A và B là các sự kiện, và $P(B) \neq 0$
- $P(B)$ là xác suất tiên nghiệm của A (prior probability), tức là xác suất của sự kiện trước khi có bằng chứng B .
- $P(B)$ là xác suất tiên nghiệm của B (evidence).
- $P(B/A)$ là xác suất có điều kiện của B khi biết A .

- $P(A/B)$ là xác suất hậu nghiệm của A (posterior probability), tức là xác suất của sự kiện sau khi đã có bằng chứng B . Về cơ bản, $P(A/B)$ là xác suất của sự kiện A khi biết sự kiện B đã xảy ra.

2.3.6. Thuật toán Support Vector Machine (SVM)

Support Vector Machine là một thuật toán học máy có giám sát được sử dụng cho cả các bài toán phân loại lẫn hồi quy. Vì vậy SVM được chia ra làm hai loại chính là SVC (Support Vector Classification) ứng dụng cho việc phân loại và SVR (Support Vector Regression) ứng dụng cho việc hồi quy.

Mục tiêu của SVM là tìm ra siêu phẳng (hyperplane) tối ưu để phân tách các lớp dữ liệu một cách rõ ràng trong không gian đặc trưng. Tùy vào loại bài toán mà hyperplane có những mục đích khác nhau.

Công thức tổng quát của SVM có dạng:

$$\min_{\omega, b, \xi} \left(\frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^n \xi_i \right)$$

Với các điều kiện:

$$y_i (\omega \cdot x_i + b) \geq 1 - \xi_i \text{ và } \xi_i \geq 0$$

Trong đó:

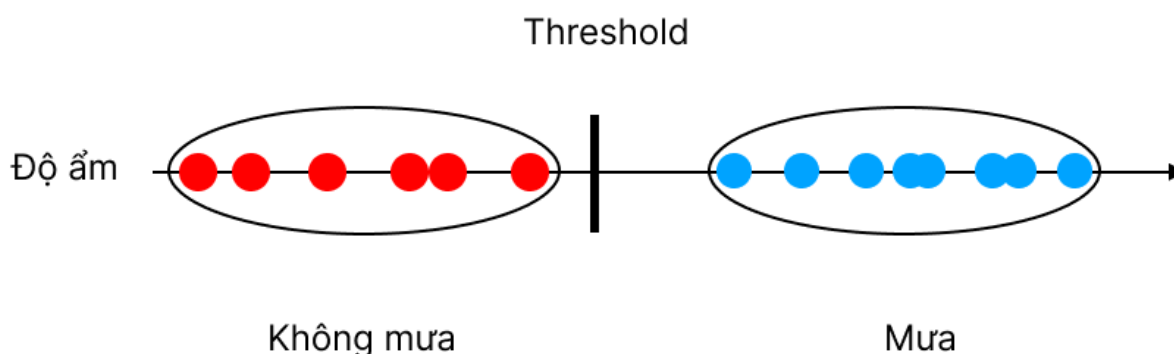
- ω là vector trọng số của hyperplane
- b là độ lệch (bias) của siêu phẳng
- ξ_i là các biến đổi (slack variables) cho phép mẫu dữ liệu i bị phân loại sai.
- C là tham số regularization điều khiển sự cân bằng giữa biên rộng (margin) và lỗi phân loại trên tập huấn luyện.

a. SVC (Support Vector Classifiers)

Ý tưởng:

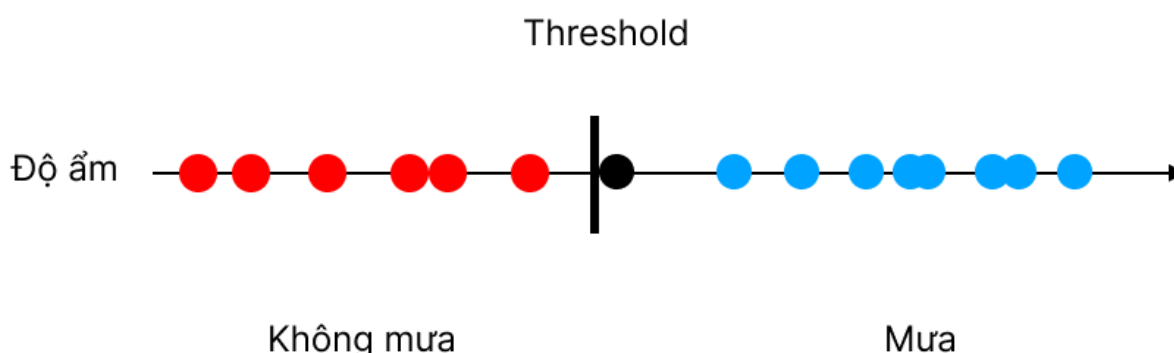
Giả sử ta có một tập dữ liệu, ta cần chia tập dữ liệu trên ra làm hai phần, một phần tích cực và một phần tiêu cực. Nhiệm vụ của các thuật toán phân loại nói chung là tạo ra một ranh giới gọi là một ngưỡng (Threshold) để phân chia kết quả.

Ví dụ: cho một tập dữ liệu một chiều về công việc dự báo trời mưa dựa vào đầu vào duy nhất là độ ẩm ở một thành phố.



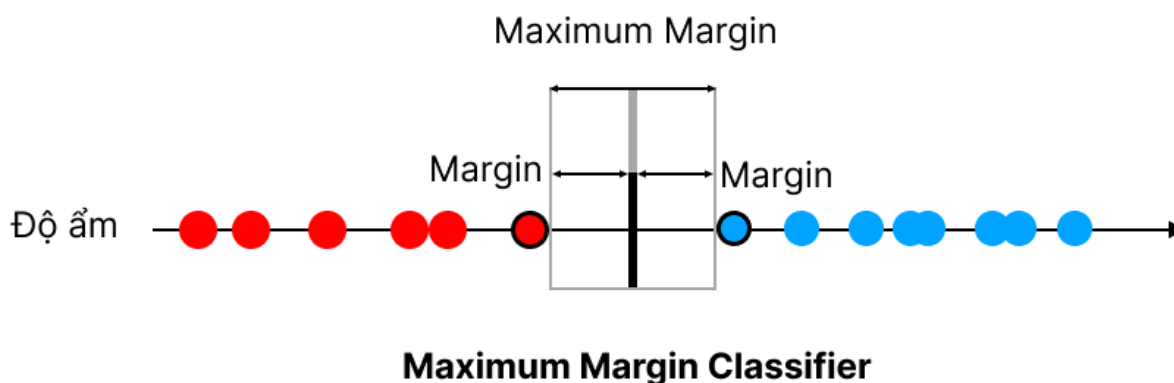
Hình 2.2: Mô hình tập dữ liệu một chiều

Dựa trên trục chiều dài độ ẩm, ta có thể phân dữ liệu thành hai nhóm bằng một ngưỡng bất kỳ ở giữa. Tuy nhiên dữ liệu thực tế phức tạp hơn nhiều và có thể xảy ra các trường hợp không công bằng như nếu điểm dữ liệu mới nằm gần điểm ngưỡng, mặc dù vị trí điểm đó nằm gần nhóm “không mưa” nên có xu hướng thuộc nhóm “không mưa” hơn nhưng vì theo sự phân tách của điểm ngưỡng, điểm dữ liệu đó được đưa vào nhóm “mưa”.



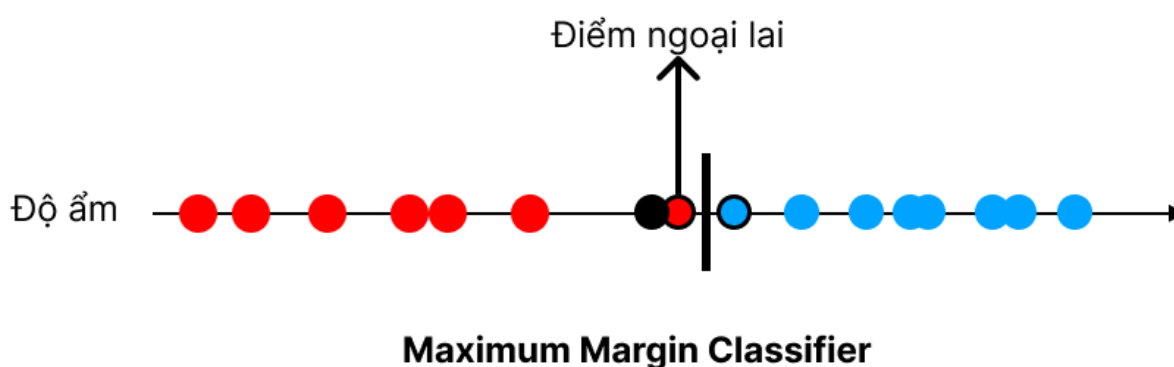
Hình 2.3: Mô hình tập dữ liệu một chiều có dữ liệu gần điểm ngưỡng

Để khắc phục điều này, ta cần đặt điểm ngưỡng ở giữa hai dữ liệu biên, là hai điểm dữ liệu gần nhất. Điểm chính giữa này được gọi là một hyperplane. Khoảng cách giữa điểm ngưỡng và một trong hai điểm dữ liệu biên gọi là margin. Và khoảng cách giữa hai điểm dữ liệu biên của hai nhóm có độ dài gấp đôi margin, được gọi là margin tối đa (maximum margin). Cách phân loại này được gọi là Maximum Margin Classifier.



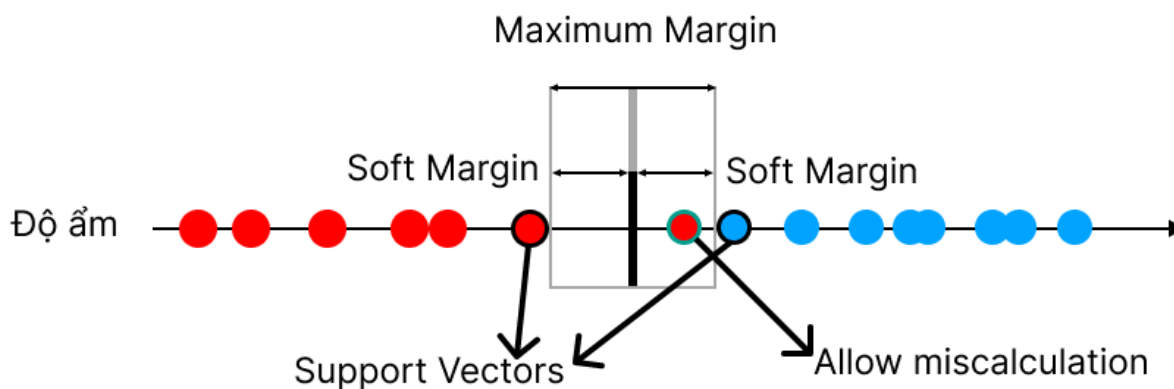
Hình 2.4: Mô hình tập dữ liệu một chiều áp dụng Maximum Margin Classifier để phân loại

Tuy nhiên trong thực tế, không ai sử dụng thuật toán phân loại này cả bởi độ đơn giản và khả năng xảy ra sai sót cao. Ví dụ như ta có một điểm dữ liệu “không mưa” nằm gần nhóm “mưa”, gọi là điểm ngoại lai. Theo thuật toán Maximum Margin Classifier, điểm ngưỡng sẽ nằm ở giữa điểm này và điểm “mưa” gần nhất. Điều này cũng dẫn đến lỗi thiên kiến (bias) khi có một điểm nằm gần nhóm “mưa” nhưng lại ở bên ngưỡng nhóm “không mưa”.



Hình 2.5: Mô hình tập dữ liệu một chiều về nhược điểm của Maximum Margin Classifier

Thuật toán phân loại SVC sẽ giải quyết vấn đề này bằng cách cho phép một số phân loại sai (Allow miscalculation) và xác định một lề khác phù hợp nhất gọi là soft margin. Các điểm dữ liệu nằm gần lề này được gọi là support vectors. SVC sẽ đánh giá các điểm dữ liệu, từ đó tìm ra các support vectors và xác định soft margin tối ưu nhất, cho phép chúng ta cân bằng được giữa độ thiên kiến và phương sai (Bias-variance tradeoff).

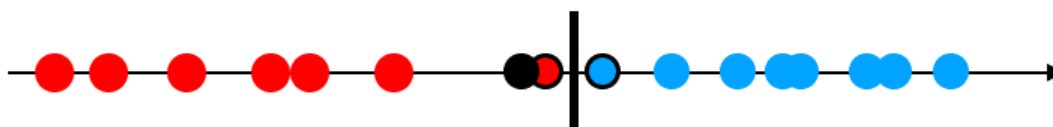


Support Vector Classifier

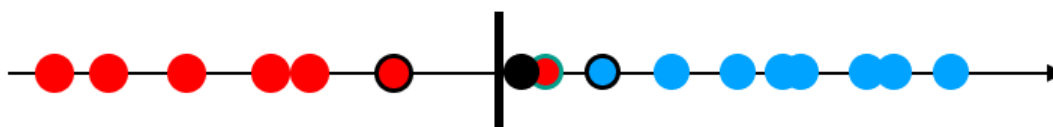
Hình 2.6: Mô hình tập dữ liệu một chiều áp dụng SVC để phân loại

Bias-variance trade off là một vấn đề quan trọng trong học máy trong việc cân bằng giữa hai lỗi mà mô hình có thể gặp phải là độ lệch (bias) và độ phương sai (variance). Như ví dụ ở trên nếu ta cho phép phân loại sai trong giai đoạn huấn luyện, độ lệch sẽ cao hơn nhưng khi thực hiện kiểm tra với dữ liệu mới thì độ phương sai sẽ thấp hơn, nghĩa là mô hình sẽ có xu hướng dự đoán tốt hơn trong giai đoạn kiểm tra. Ngược lại nếu độ lệch quá thấp, điều này sẽ dẫn đến tình trạng Overfitting khiến cho độ phương sai cao và dự đoán kém đi.

Bias thấp hơn, variance cao hơn:



Bias cao hơn, variance thấp hơn:

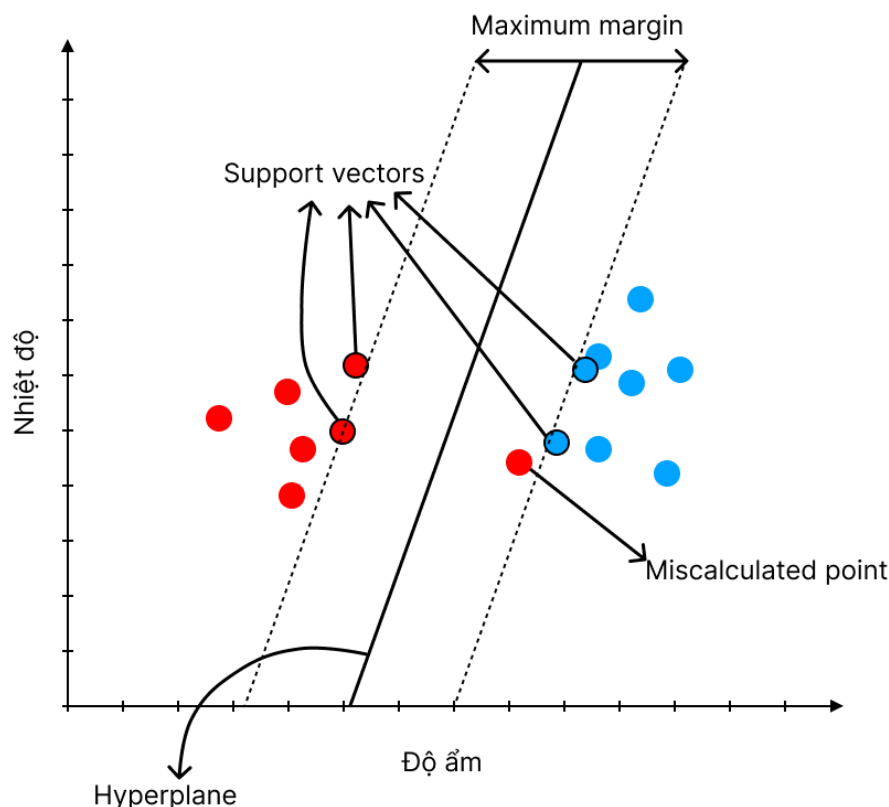


Hình 2.7: Sự cân bằng giữa độ lệch (bias) và phương sai (variance)

Với ví dụ trên, thuật toán cụ thể được sử dụng là Linear Support Vector Classifier làm việc tốt với các dữ liệu tuyến tính.

Ưu điểm của SVC tuyến tính:

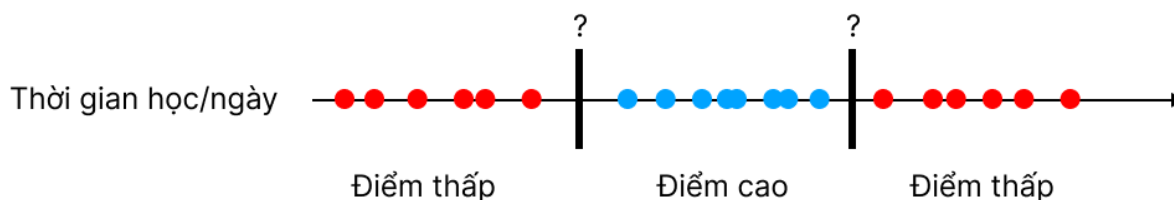
- Không chỉ áp dụng được cho mô hình không gian một chiều mà còn hiệu quả cho nhiều chiều không gian cao hơn. Ví dụ:



Hình 2.8: Mô hình tập dữ liệu hai chiều sử dụng SVC để phân loại

- Dễ dàng giải quyết được vấn đề overfitting bằng soft margin
- Hiệu quả với các bài toán phân loại nhị phân
- Dễ dàng tối ưu để tìm hyperplane phù hợp

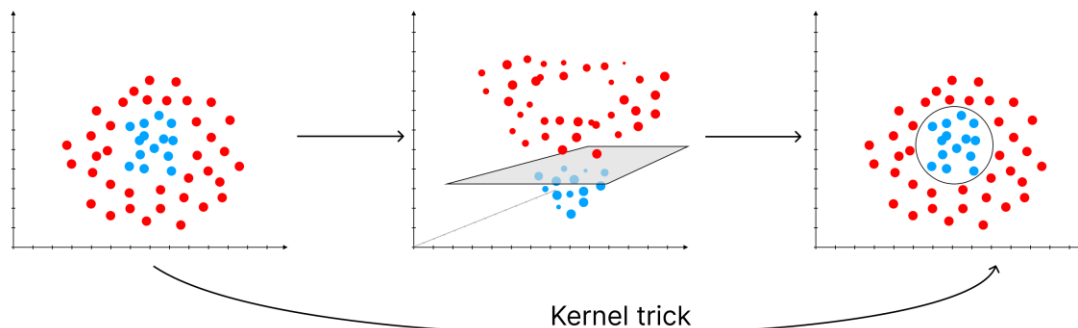
Nhược điểm của SVC tuyến tính: Không phù hợp với dữ liệu không tuyến tính. Ví dụ trong việc phân loại số lượng thời gian học mỗi ngày phù hợp cho học sinh để đạt điểm thi tốt. Nếu cho học sinh học quá ít thì điểm của học sinh sẽ không cao. Nếu cho học quá nhiều, giờ học dần trở nên kém chất lượng do mệt mỏi và điểm thi cũng thấp đi. Như vậy chỉ có một khoảng số lượng thời gian phù hợp, không quá ít, không quá nhiều mới cải thiện được điểm của học sinh. Nếu vậy, với phương pháp tìm ngưỡng tuyến tính thì làm sao có thể tìm được hyperplane phù hợp để chia hai nhóm dữ liệu?



Hình 2.9: Mô hình tập dữ liệu một chiều không tuyến tính

Cách giải quyết: ta cần biến đổi các điểm dữ liệu này sang một cái dạng khác để tạo một ngưỡng không tuyến tính bằng cách chuyển đổi chúng từ chiều không gian thấp hơn sang chiều không gian cao hơn với các hàm Kernel để tìm ra hyperplane phân tách

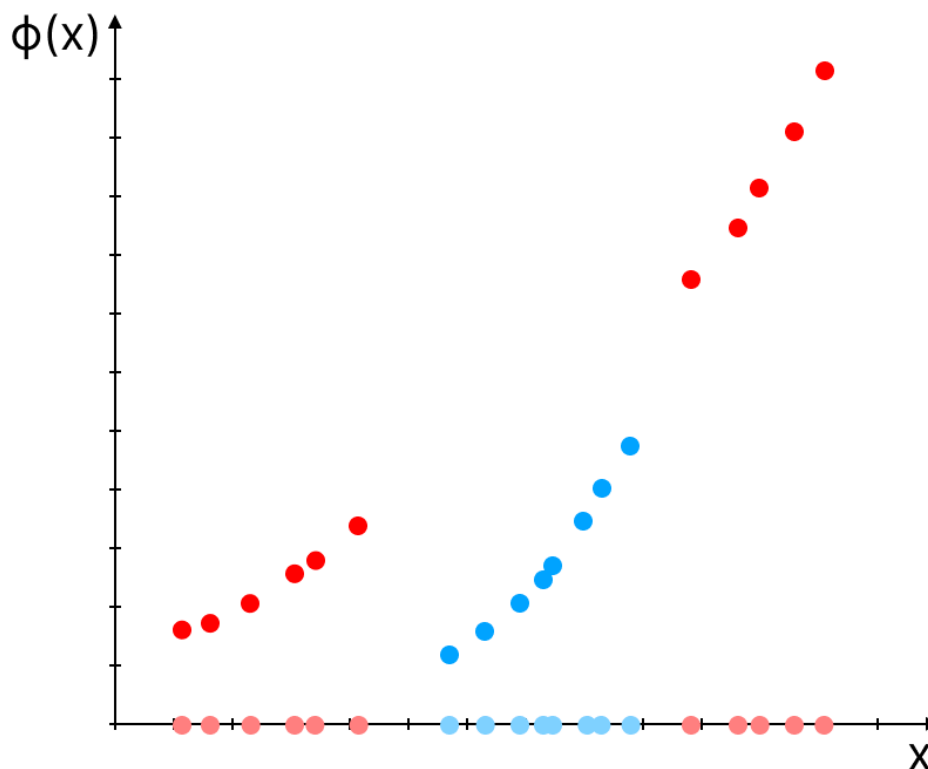
dữ liệu. Tuy nhiên, các hàm Kernel này không thực sự thực hiện việc ánh xạ tọa độ dữ liệu vào không gian cao hơn, bởi việc này rất phức tạp và làm tiêu tốn nhiều thời gian và tài nguyên tính toán, mà chúng chỉ tính toán các mối quan hệ giữa các điểm như thể chúng đang ở chiều không gian cao hơn. Kỹ thuật này được gọi là Kernel trick.



Hình 2.10: Mô hình tập dữ liệu hai chiều áp dụng Kernel Trick để tìm hyperplane

Khi ta sử dụng mô hình SVC cùng với các hàm kernel, ta gọi nó là thuật toán mô hình SVM. Thuật toán SVM không học từ đầu vào gốc (x) mà nó sẽ học dữ liệu ở chiều không gian cao hơn được tính từ ánh xạ đặc trưng (feature mapping) $\Phi(x)$. Để tối ưu mô hình và tìm hyperplane, thuật toán SVM cần tính toán mối quan hệ giữa từng cặp điểm dữ liệu của tất cả các điểm bằng cách tính tích vô hướng của hai điểm ánh xạ hay còn gọi là dot product:

$$\phi(x^{(i)}) \cdot \phi(x^{(i)})$$



Hình 2.11: Mô hình tập dữ liệu hai chiều biểu diễn tính toán dot product

Trong trường hợp các bài toán ở chiều không gian cao, việc tính toán $\phi(x^{(i)}) \cdot \phi(x^{(i)})$ trở nên rất khó khăn. Thay vì vậy ta có thể tính toán các hàm kernel với độ phức tạp tính toán giảm đi đáng kể để tìm được dot product.

$$K(x, y) = \phi(x^{(i)}) \cdot \phi(x^{(i)})$$

Có rất nhiều hàm kernel có thể được sử dụng trong SVM, và việc chọn hàm kernel phù hợp phụ thuộc vào đặc điểm của dữ liệu và bài toán cụ thể. Kernel đa thức (Polynomial Kernel) là một trong những hàm kernel phổ biến nhất.

$$K(x, y) = (x \cdot y + c)^d$$

Trong đó:

- x, y là cặp điểm dữ liệu gốc. Theo công thức, x được nhân vô hướng với y .
- d là bậc của hàm.
- c là thông số điều chỉnh.

Một hàm kernel quan trọng khác là hàm Radical Basis Function (RBF) Kernel. Hàm này đặc biệt có thể hỗ trợ cho vô số chiều không gian.

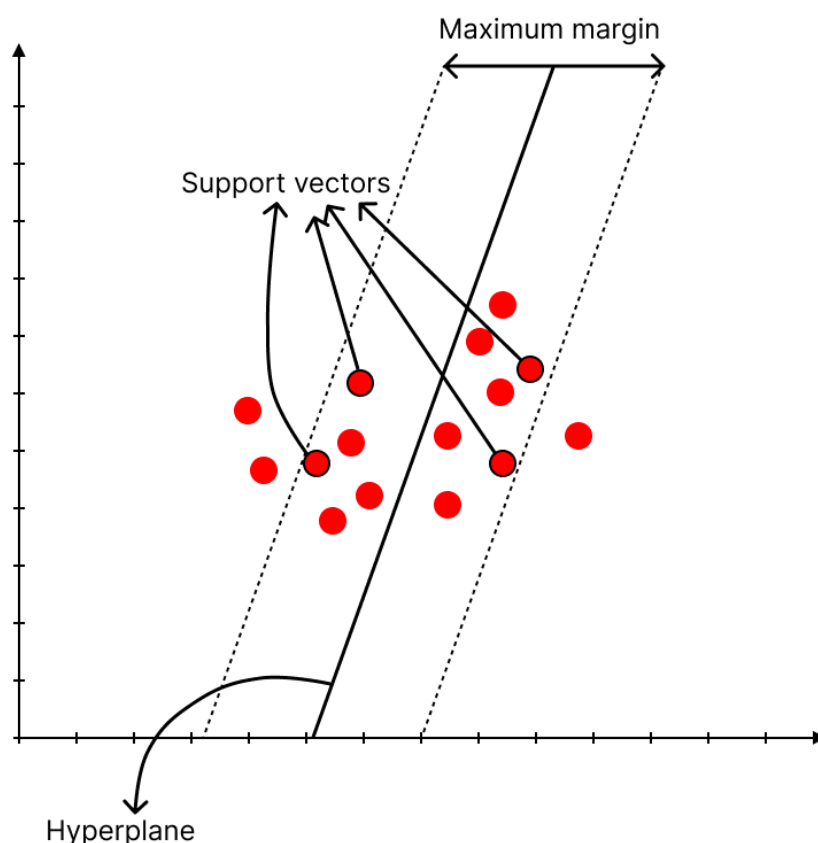
$$K(x, y) = e^{(-\gamma \|x - y\|^2)}$$

Trong đó:

- x, y là cặp điểm dữ liệu gốc.
- γ là tham số xác định độ ảnh hưởng của một điểm dữ liệu.

b. SVR (Support Vector Regressors)

Vì là một phần của thuật toán SVM, SVR cũng áp dụng các phương pháp tìm hyperplane của SVM. Tuy nhiên mục tiêu của thuật toán SVR là tìm hyperplane sao cho hầu như các dữ liệu huấn luyện đều nằm trong margin.



Hình 2.12: Mô hình tập dữ liệu hai chiều áp dụng thuật toán SVR

SVR áp dụng hiệu quả cho các bài toán hồi quy với các dữ liệu phức tạp và không tuyến tính. Tuy nhiên ta sẽ không đi sâu vào SVR vì mục tiêu của đề tài này là xác định xem hệ thống mạng là “bình thường” hay “độc hại” nên ứng dụng các thuật toán phân loại như SVC sẽ phù hợp hơn với yêu cầu của đề tài.

2.4. Các công trình liên quan

a. Các công trình nghiên cứu trong nước

Vào năm 2018, Nguyễn Trọng Hưng, Hoàng Xuân Dậu và Vũ Xuân Hạnh đã công bố nghiên cứu về “Phát hiện botnet dựa trên phân loại tên miền sử dụng các kỹ thuật học máy”. [5] Trong những năm gần đây, các botnet đã trở thành một trong các nguy cơ gây mất an toàn thông tin hàng đầu do chúng không ngừng phát triển về cả quy mô và mức độ tinh vi. Nhiều giải pháp phát hiện botnet, như dựa trên honeynet, hoặc IDS đã được đề xuất. Tuy nhiên, các giải pháp dựa trên IDS sử dụng chữ ký tỏ ra kém hiệu

quả do botnet được trang bị khả năng tự cập nhật mã và nhiều kỹ thuật lẩn tránh tinh vi. Kết quả nhiều nghiên cứu cho thấy các phương pháp phát hiện botnet dựa trên bất thường tỏ ra hiệu quả hơn. Bài báo này giới thiệu mô hình và khảo sát hiệu quả phát hiện botnet dựa trên phân loại các tên miền độc hại của botnet và các tên miền bình thường sử dụng các kỹ thuật học máy. Kết quả thử nghiệm cho thấy các thuật toán học máy có giám sát có thể sử dụng hiệu quả trong phát hiện botnet dựa trên phân loại tên miền và thuật toán học máy Rừng ngẫu nhiên cho độ chính xác phát hiện cao nhất với độ chính xác phân loại chung đạt 93,21% với tập huấn luyện có tỷ lệ giữa số lượng các tên miền lành tính và số lượng tên miền độc hại nên chọn là 3:5.

Năm 2022, Tạp chí Khoa học Công nghệ và Thực phẩm đã phát hành bài báo nghiên cứu về “Phát hiện tấn công SQL Injection bằng học máy” do Trần Đức Tốt và các cộng sự khác Nguyễn Trung Kiên, Trương Hữu Phúc, Lê Ngọc Sơn và Trần Thị Bích Vân thực hiện. [6] Lỗ hổng SQL injection đang được xem là một trong những lỗ hổng bảo mật có độ nguy hiểm thuộc dạng cao nhất và liên tục nằm trong top 10 OWASP. Các cuộc tấn công SQL injection luôn gây ảnh hưởng nghiêm trọng tới các doanh nghiệp hay các trang web cá nhân và chúng vẫn đang tăng dần từng ngày. Nghiên cứu này đã đề xuất giải pháp phát hiện tấn công SQL Injection bằng cách áp dụng các kỹ thuật học máy. Cụ thể họ sử dụng hai thuật toán Multinomial Naïve Bayes và K-Nearest neighbors để phân loại dữ liệu được lưu trên file logs dưới dạng văn bản. Cả hai mô hình này đều được thử nghiệm và so sánh để đánh giá hiệu quả trong việc phát hiện tấn công SQL injection. Kết quả nghiên cứu cho thấy rằng thuật toán Multinomial Naive Bayes có hiệu suất tốt hơn trong việc phát hiện tấn công SQL injection với chỉ số F1-Score là 88.2274% so với 88.2166%. Các thí nghiệm, thực nghiệm cho thấy rằng hệ thống phát hiện được các cuộc tấn công SQL injection có tỷ lệ chính xác và hiệu quả cao.

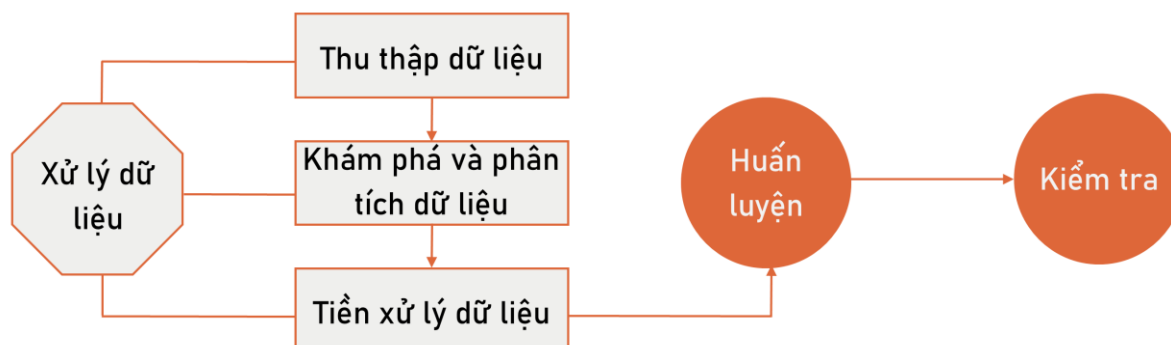
b. Các công trình nghiên cứu trên thế giới

Năm 2022, Rachid Tahri cùng các cộng sự đã công bố một nghiên cứu về “Intrusion Detection System Using machine learning Algorithms”. [7] Thế giới ngày nay có nhiều sự thay đổi do sự phát triển của internet. Internet cung cấp, hỗ trợ cho con người rất nhiều lợi ích như liên lạc, truy cập thông tin nhanh chóng và dễ dàng. Tuy nhiên, việc chia sẻ dữ liệu chuyên nghiệp và cá nhân đi kèm với nhiều rủi ro cho cả cá nhân và tổ chức. Internet đã trở thành một yếu tố quan trọng trong cuộc sống hàng ngày của chúng ta, do đó, bảo mật dữ liệu của chúng ta có thể bị đe dọa bất cứ lúc nào. Vì lý do này, Hệ thống Phát hiện Xâm nhập (IDS) đóng vai trò quan trọng trong việc bảo vệ người dùng internet chống lại các cuộc tấn công mạng độc hại. Trong bài báo này, các nhà nghiên cứu tập trung vào ba giai đoạn chính; bắt đầu bằng các thuật toán học máy Naive Bayes, SVM, và K-nearest-neighbors. Những thuật toán này sẽ được sử dụng để xác định độ chính xác tốt nhất bằng cách sử dụng bộ dữ liệu UNSW-NB15 trong giai đoạn đầu tiên. Dựa trên kết quả của giai đoạn đầu tiên, giai đoạn thứ hai sẽ được sử dụng để xử lý cơ sở dữ liệu của họ với thuật toán hiệu quả nhất. Hai bộ dữ liệu khác nhau sẽ được sử dụng trong các thí nghiệm của họ để đánh giá hiệu suất của mô hình. Các bộ dữ liệu NSL-KDD và UNSW-NB15 được sử dụng để đo lường hiệu suất của phương pháp đề xuất nhằm đảm bảo tính hiệu quả của nó. Kết quả cho thấy, so với hai thuật toán còn lại, SVM cho thấy hiệu suất tốt nhất bất kể kích thước của cơ sở dữ liệu hay loại tấn công mà nó chứa. Ta có thể thấy điều đó qua độ chính xác của SVM là 97.77777% với bộ UNSW-NB15 và 97,29% với bộ NSL-KDD.

Cũng vào năm 2022, Mohammad Kazim đã công bố nghiên cứu “Network Anomaly Detection Models Using Machine Learning Algorithms”. [8] Bài nghiên cứu này cũng có vài điểm tương đồng với nghiên cứu của Rachid Tahri cùng các cộng sự đã đề cập trước đó đề cập cơ sở lý thuyết về hệ thống phát hiện xâm nhập IDS và tác giả cũng sử dụng hai bộ dữ liệu NSL-KDD và UNSW-NB15 cho việc nghiên cứu. Tuy nhiên, nghiên cứu này thiết kế các mô hình không chỉ sử dụng học máy mà còn có mô hình học sâu để phân loại dị thường mạng với độ chính xác cao và tỷ lệ cảnh báo sai thấp. Trong đó, bốn mô hình được phát triển: mô hình chọn đặc trưng, mô hình phát hiện dị thường mạng sử dụng học máy và học sâu, và mô hình phát hiện dị thường mạng theo các giao thức tiêu chuẩn. Các mô hình được đánh giá trên hai bộ dữ liệu UNSW-NB15 và NSL-KDD. Phương pháp chọn đặc trưng dựa trên học tập ensemble và xử lý mất cân bằng dữ liệu bằng oversampling và undersampling. Tác giả sử dụng Denoising Autoencoder và XGBoost để phát hiện dị thường cho mô hình, cùng với 1D CNN cho phân loại. Cuối cùng, mô hình xử lý dữ liệu theo các giao thức TCP, UDP và khác, tối ưu hóa tham số và xử lý vấn đề chồng lấn trong dữ liệu IDS. Kết quả nghiên cứu cho thấy, đối với phân loại nhị phân (Binary Classification), Mô hình SKM-XGB đạt tỷ lệ phát hiện 99.01% và 99.37% cho các bộ dữ liệu UNSW-NB15 và NSL-KDD tương ứng, với tỷ lệ lỗi giả (FAR) lần lượt là 0.58% và 0.64%. Còn đối với phân loại nhiều lớp (Multiclass Classification), Mô hình SKM-XGB vượt trội hơn tất cả các mô hình tập hợp khác về các chỉ số như Accuracy (99.08%, 99.57%), Precision(98.46%, 99.26%), Recall (97.49%, 99.22%), F-score(97.84%, 99.24%), và FAR (0.61%, 0.2%) cho bộ UNSW-NB15 và NSL-KDD tương ứng. Về thời gian huấn luyện, phương pháp Bagging với KNN là tốt nhất (0.06s – UNSW-NB15, 0.03s – NSL-KDD, và về thời gian kiểm tra, MLP là tốt nhất trên cả hai bộ dữ liệu (0.17s – UNSW-NB15, 0.01s – NSL-KDD).

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH HỌC MÁY SVM PHÁT HIỆN TẤN CÔNG DOS

Quá trình phát triển mô hình SVM phát hiện tấn công DoS bao gồm 3 giai đoạn: (1) giai đoạn xử lý dữ liệu, (2) giai đoạn huấn luyện và (3) giai đoạn kiểm tra.

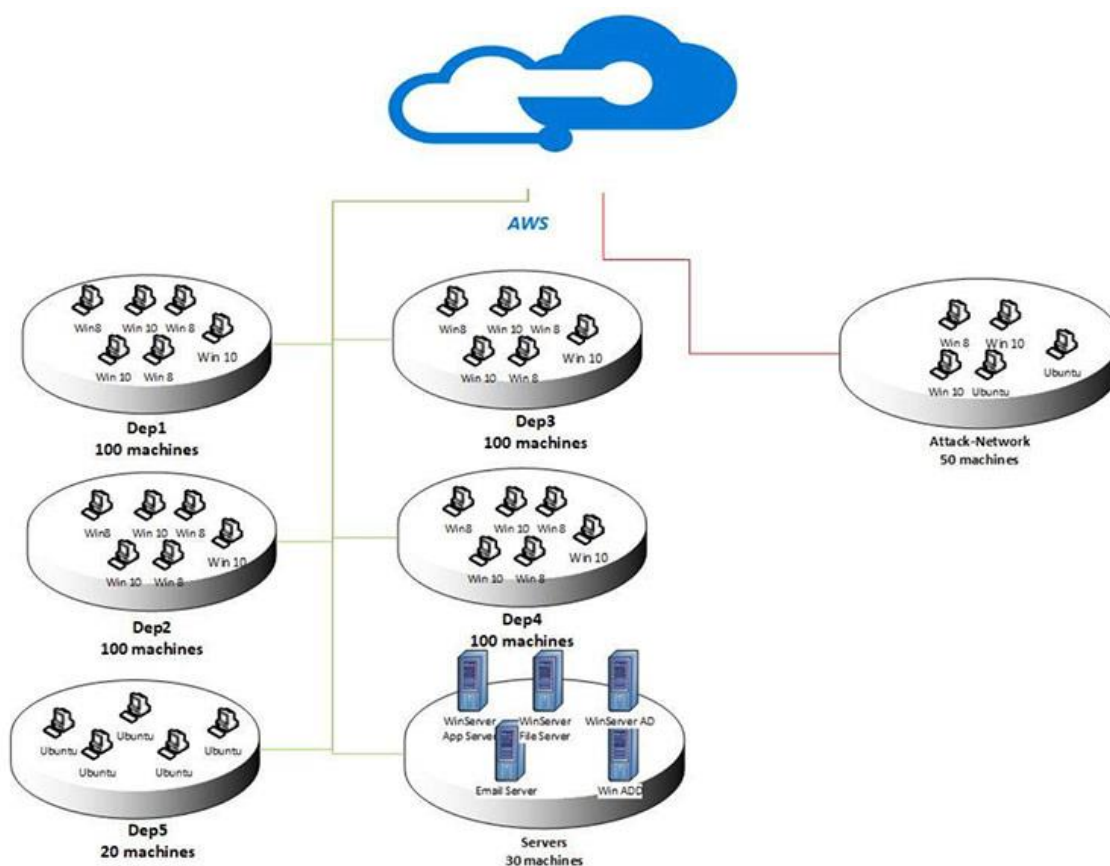


Hình 3.1: Sơ đồ tổng quát quá trình xây dựng mô hình SVM

3.1. Giới thiệu bộ dữ liệu

Bộ dữ liệu tên CSE-CIC-IDS2018 on AWS của trường đại học New Brunswick [9] sẽ là tập dữ liệu sử dụng cho công việc xây dựng mô hình học máy SVM phát hiện tấn công DoS. CSE-CIC-IDS2018 on AWS là một dự án hợp tác giữa Cơ quan An ninh Truyền thông (CSE) và Viện An ninh Mạng Canada (CIC), được tạo vào năm 2018, nhằm khắc phục tình trạng nhiều bộ dữ liệu không thực tế hoặc thiếu sót các thông tin quan trọng không thể chia sẻ do vấn đề bảo mật.

Dự án mô phỏng các kịch bản tấn công trên cấu trúc liên kết mạng LAN dựa trên nền tảng AWS. Để có sự đa dạng và phức tạp như các mạng thực tế, họ đã cài đặt 5 mạng con, cụ thể là: Phòng R&D (Dep1), Phòng Quản lý (Dep2), Phòng Kỹ thuật (Dep3), Phòng Thư ký và Vận hành (Dep4), Phòng IT (Dep5), và các phòng máy chủ. Đối với tất cả các phòng ban ngoại trừ phòng IT, các máy đã được cài đặt các hệ điều hành MS Windows khác nhau (Windows 8.1 và Windows 10) và tất cả các máy tính trong phòng IT đều sử dụng Ubuntu. Đối với phòng máy chủ, họ triển khai các máy chủ MS Windows khác nhau như 2012 và 2016.



Hình 3.2: Mô hình cấu trúc mạng [9]

Mô hình được mô phỏng bao gồm bảy kịch bản tấn công:

Loại tấn công	Công cụ sử dụng	Thời lượng
Bruteforce	FTP – Patator SSH – Patator	Một ngày
DoS	Hulk, GoldenEye, Slowloris, Slowhttptest	Một ngày
Heartbleed	Heartleech	Một ngày
Web	Damn Vulnerable Web App (DVWA) In-house selenium framework (XSS và Brute-force)	Hai ngày
Infiltration	Cấp độ 1: Dropbox download trong thiết bị windows Cấp độ 2: Nmap và portscan	Hai ngày

Botnet	Ares (developed by Python): remote shell, file upload/download, capturing screenshots và key logging	Một ngày
DDoS+PortScan	Low Orbit Ion Canon (LOIC) cho UDP, TCP, hoặc HTTP requests	Hai ngày

Bảng 3.1: Các kịch bản tấn công của bộ dữ liệu

Tổng cộng, có tám mươi cột trong bộ dữ liệu này, mỗi cột tương ứng với một mục nhập trong hệ thống ghi nhật ký IDS mà Đại học New Brunswick đã triển khai. Vì hệ thống của họ phân loại lưu lượng mạng là cả đi tới (forward) và đi lui (backward), nên có các cột cho cả hai loại lưu lượng này. Những cột quan trọng nhất trong bộ dữ liệu này bao gồm [10]:

- Dst Port (Cổng đích)
- Protocol (Giao thức)
- Flow Duration (Thời gian luồng)
- Tot Fwd Pkts (Tổng số gói tin đi tới)
- Tot Bwd Pkts (Tổng số gói tin đi lui)
- Label (Nhãn)

Nghiên cứu của bộ dữ liệu này cụ thể do Iman Sharafaldin, Arash Habibi Lashkari và Ali A. Ghorbani thực hiện [11]. Bài nghiên cứu cho biết các đặc trưng quan trọng cho từng loại tấn công bằng phương pháp lựa chọn đặc trưng (feature selection) sử dụng thuật toán RandomForestRegressor. Cụ thể các đặc trưng quan trọng cho phát hiện tấn công DoS và cho hoạt động bình thường (Benign) là:

Label	Feature	Weight
Benign	B.Packet Len Min	0.0479
	Subflow F.Bytes	0.0007
	Total Len F.Packets	0.0004
	F.Packet Len Mean	0.0002
DoS GoldenEye	B.Packet Len Std	0.1585
	Flow IAT Min	0.0317

	Fwd IAT Min	0.0257
	Flow IAT Mean	0.0214
DoS Hulk	B.Packet Len Std	0.2028
	B.Packet Len Std	0.1277
	Flow Duration	0.0437
	Flow IAT Std	0.0227
DoS Slowhttp	Flow Duration	0.0443
	Active Min	0.0228
	Active Mean	0.0219
	Flow IAT Std	0.0200
DoS slowloris	Flow Duration	0.0431
	F.IAT Min	0.0378
	B.IAT Mean	0.0300
	F.IAT Mean	0.0265

Bảng 3.2: Lựa chọn đặc trưng

Toàn bộ dataset có thể tải về được bằng lệnh command: `aws s3 sync s3://cse-cic-ids2018/ [INSERT DOWNLOAD DIRECTORY HERE] --no-sign-request`. Tuy nhiên vì bộ dữ liệu quá lớn, khoảng 450 GB, nên ta chỉ sử dụng một phần của tập dữ liệu này. Tập dữ liệu trích dẫn (6.89 GB) được tổng hợp từ trang web Kaggle – kho dữ liệu lớn và đa dạng các tập dữ liệu miễn phí [10]. Tập dữ liệu được chia ra làm 10 file csv, mỗi file là dữ liệu log được tổng hợp dựa vào thời gian tổng hợp. Sau đây là thông tin chi tiết của từng tập dữ liệu:

Tập dữ liệu	Ngày tổng hợp	Mô tả
02-14-2018.csv	14/2/2018	Phần lớn lưu lượng truy cập bình thường (Benign) so với các cuộc tấn công thực tế.
02-15-2018.csv	15/2/2018	Phần lớn lưu lượng truy cập thông thường (Benign) so với các cuộc tấn công thực tế, với 95% lưu lượng truy cập thực tế là không có hại.
02-16-2018.csv	16/2/2018	Phần lớn các cuộc tấn công DoS so với lưu lượng truy cập thông thường, trong đó cuộc tấn công Hulk được ưu tiên hơn.
02-20-2018.csv	20/2/2018	Phần lớn lưu lượng truy cập bình thường, với 93% là vô hại.

02-21-2018.csv	21/2/2018	Có phần lớn các cuộc tấn công DDoS so với lưu lượng truy cập thông thường, trong đó cuộc tấn công HOIC được ưu tiên.
02-22-2018.csv	22/2/2018	Tất cả thông tin trong tệp cụ thể này được coi là vô hại.
02-23-2018.csv	23/2/2018	Tất cả thông tin trong tệp cụ thể này được coi là vô hại.
02-28-2018.csv	28/2/2018	Phần lớn thông tin trong tệp này là vô hại. Tuy nhiên, nó cũng chứa 11% các cuộc tấn công xâm nhập (Infiltration).
03-01-2018.csv	1/3/2018	Phần lớn thông tin trong tệp này được coi là vô hại. Tuy nhiên, 28% lưu lượng truy cập được coi là một cuộc tấn công xâm nhập (Infiltration).
03-02-2018.csv	2/3/2018	Phần lớn thông tin trong tệp này được coi là vô hại, tuy nhiên, 27% lưu lượng truy cập được coi là một cuộc tấn công botnet.

Bảng 3.3: Chi tiết các tập dữ liệu định dạng csv

3.2. Xử lý dữ liệu

Đầu tiên, ta import các thư viện cần thiết cho công việc xử lý dữ liệu:

```
import numpy as np
import pandas as pd
import warnings
pd.set_option('display.max_columns', None)
warnings.filterwarnings('ignore')
import os
import gc
import matplotlib.pyplot as plt
from sklearn.utils import resample
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import make_scorer, hinge_loss
import matplotlib.colors
from sklearn.inspection import DecisionBoundaryDisplay
```

```
from sklearn.decomposition import PCA
```

3.2.1. Phân tích dữ liệu

Công việc tiếp theo là đọc và khám phá dữ liệu từ nguồn. Các tập dữ liệu được tách ra để dễ xử lý nhưng quá trình đọc và khám phá dữ liệu ban đầu cũng tương tự nhau.

- Đọc một tệp dữ liệu:

```
df_01 = pd.read_csv('../Dataset/IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)/Remain/03-02-2018.csv')
```

- Đọc nhiều tệp dữ liệu:

```
df_dos = pd.DataFrame(data=None)
for dirname, _, filenames in os.walk('../Dataset/IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)/Remain/DoS/'):
    for filename in filenames:
        file_path = os.path.join(dirname, filename)
        print(file_path)
        df_temp = pd.read_csv(file_path)
        df_dos = pd.concat([df_dos, df_temp], ignore_index=True)
        del df_temp
        gc.collect()

../Dataset/IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)/Remain/DoS/02-15-2018.csv
../Dataset/IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)/Remain/DoS/02-16-2018.csv
```

Khám phá dữ liệu ban đầu bao gồm tìm hiểu các thông tin cơ bản của tập dữ liệu như số lượng hàng, cột, kiểu dữ liệu, các thống kê mô tả và các giá trị của các đặc trưng quan trọng.

Các tập dữ liệu có các cột, và giá trị gần như tương tự nhau, có thể có thêm các kịch bản tấn công khác và kiểu dữ liệu khác nhau. Các tập dữ liệu có nhãn tấn công DoS là: 02-15-2018.csv và 02-16-2018.csv. Các file còn lại bao gồm các kịch bản tấn công khác. Tổng cộng hai tập dữ liệu đó có các nhãn sau:

```
# check the number of values for labels
df_dos['Label'].value_counts()
```

```
Label
```

Benign	1442849
DoS attacks-Hulk	461912
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
Label	1
Name: count, dtype: int64	

Số lượng hàng: 2097150

Số lượng cột: 80

```

•[7]: # check the number of rows and columns
print('Number of Rows (Samples): %s' % str((df_dos.shape[0])))
print('Number of Columns (Features): %s' % str((df_dos.shape[1])))

Number of Rows (Samples): 2097150
Number of Columns (Features): 80

```

Hình 3.3: Code hiển thị số lượng hàng (mẫu) và cột (đặc trưng)

Chi tiết các cột:

`df_dos.columns`

Index(['Dst Port', 'Protocol', 'Timestamp', 'Flow Duration', 'Tot Fwd Pkts',
'Tot Bwd Pkts', 'TotLen Fwd Pkts', 'TotLen Bwd Pkts', 'Fwd Pkt Len Max',
'Fwd Pkt Len Min', 'Fwd Pkt Len Mean', 'Fwd Pkt Len Std',
'Bwd Pkt Len Max', 'Bwd Pkt Len Min', 'Bwd Pkt Len Mean',
'Bwd Pkt Len Std', 'Flow Byts/s', 'Flow Pkts/s', 'Flow IAT Mean',
'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Tot',
'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min',
'Bwd IAT Tot', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max',
'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags',
'Bwd URG Flags', 'Fwd Header Len', 'Bwd Header Len', 'Fwd Pkts/s',
'Bwd Pkts/s', 'Pkt Len Min', 'Pkt Len Max', 'Pkt Len Mean',
'Pkt Len Std', 'Pkt Len Var', 'FIN Flag Cnt', 'SYN Flag Cnt',
'RST Flag Cnt', 'PSH Flag Cnt', 'ACK Flag Cnt', 'URG Flag Cnt',

```
'CWE Flag Count', 'ECE Flag Cnt', 'Down/Up Ratio', 'Pkt Size Avg',
'Fwd Seg Size Avg', 'Bwd Seg Size Avg', 'Fwd Byts/b Avg',
'Fwd Pkts/b Avg', 'Fwd Blk Rate Avg', 'Bwd Byts/b Avg',
'Bwd Pkts/b Avg', 'Bwd Blk Rate Avg', 'Subflow Fwd Pkts',
'Pkts', 'Subflow Bwd Pkts', 'Subflow Bwd Byts',
'Init Fwd Win Byts', 'Init Bwd Win Byts', 'Fwd Act Data Pkts',
'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max',
'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min', 'Label'],
dtype='object')
```

Kiểu dữ liệu:

```
df_dos.info()

_____

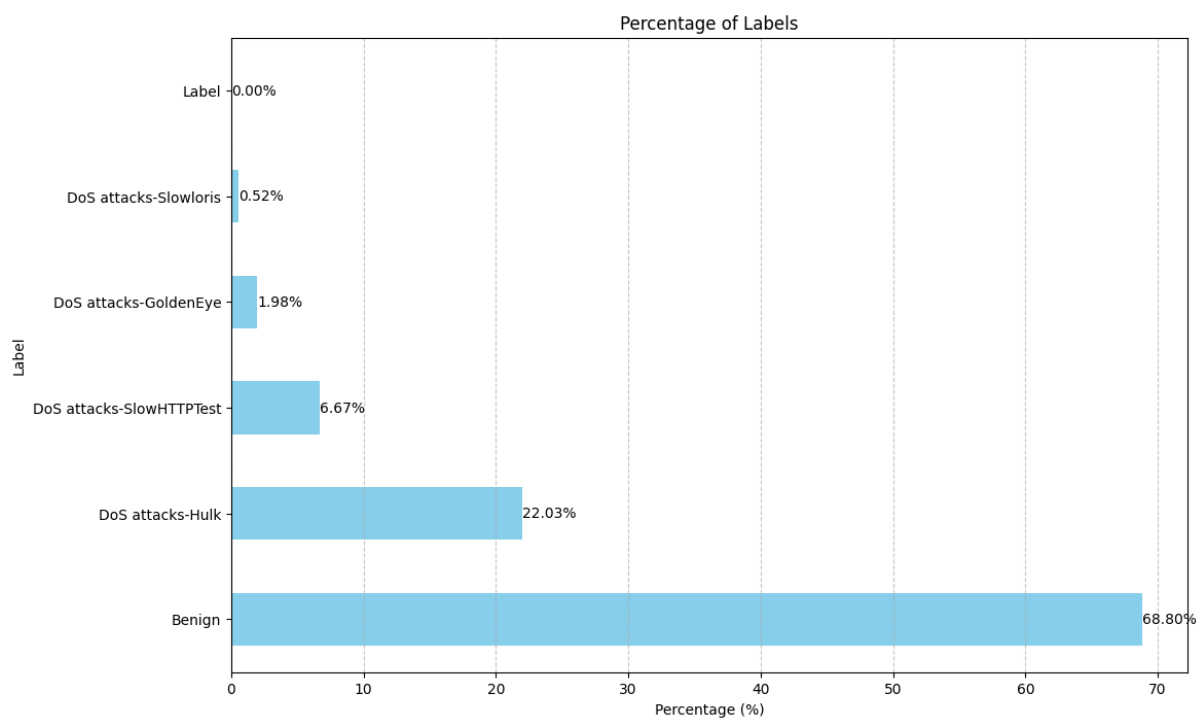
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2097150 entries, 0 to 2097149
Data columns (total 80 columns):
#   Column                Dtype
---  -
0   Dst Port              object
1   Protocol              object
2   Timestamp            object
3   Flow Duration        object
...
76  Idle Std              object
77  Idle Max              object
78  Idle Min              object
79  Label                 object
dtypes: dtypes: dtypes: object(80)
```

Tỉ lệ các loại nhãn:

```
label_counts = df_dos['Label'].value_counts()
# Tính tỷ lệ phần trăm cho mỗi nhãn
label_percentages = (label_counts / label_counts.sum()) * 100
```

```
# Tạo biểu đồ cột ngang hiển thị phần trăm
plt.figure(figsize=(12, 8))
label_percentages.plot(kind='barh', color='skyblue')
plt.xlabel('Percentage (%)')
plt.ylabel('Label')
plt.title('Percentage of Labels')
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Hiển thị giá trị phần trăm trên mỗi thanh của biểu đồ
for index, value in enumerate(label_percentages):
    plt.text(value, index, f'{value:.2f}%', va='center')
plt.show()
```



Hình 3.4: Biểu đồ cột ngang tỉ lệ phần trăm của mỗi nhãn

Tỉ lệ các mẫu bình thường và các mẫu tấn công:

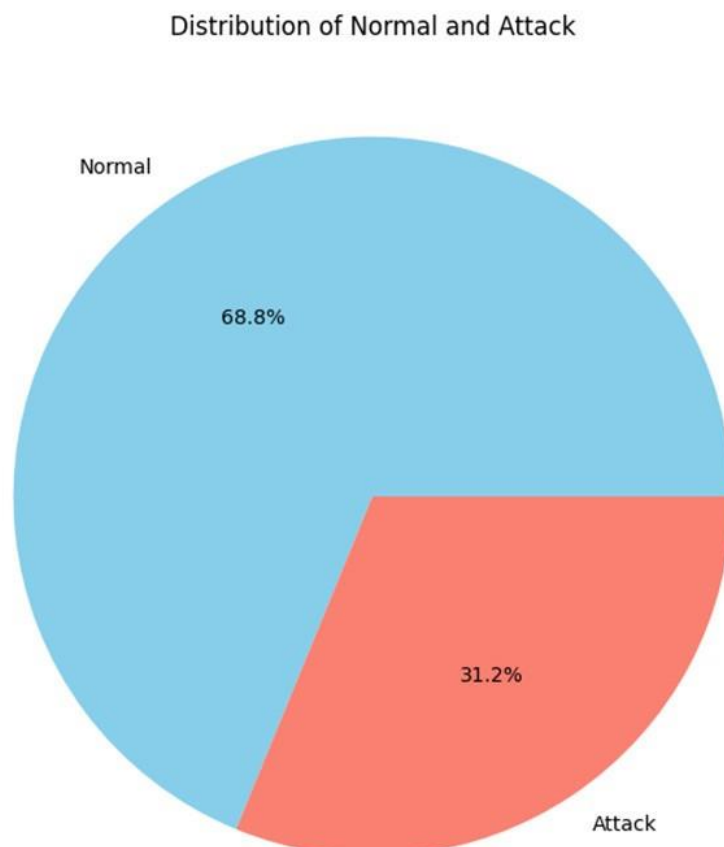
```
label_counts = df_dos['Label'].value_counts()
# Chuyển đổi các nhãn thành hai nhóm: "Normal" và "Attack"
label_groups = label_counts.copy()
label_groups['Normal'] = label_groups.pop('Benign')
```

```
label_groups['Attack'] = label_groups.sum() - label_groups['Normal']

# Tạo DataFrame cho biểu đồ tròn
pie_data = pd.Series({
    'Normal': label_groups['Normal'],
    'Attack': label_groups['Attack']
})

# Tính tỷ lệ phần trăm
pie_data_percentage = (pie_data / pie_data.sum()) * 100

# Tạo biểu đồ tròn
plt.figure(figsize=(8, 8))
plt.pie(pie_data_percentage, labels=pie_data_percentage.index, autopct='%1.1f%%',
        colors=['skyblue', 'salmon'])
plt.title('Distribution of Normal and Attack')
plt.show()
```



Hình 3.5: Biểu đồ tròn tỉ lệ phần trăm các mẫu bình thường và mẫu tấn công

Các giá trị của đặc trưng Protocol:

```
# check the number of values for protocol
df_protocol = df_dos.copy()

# Chuyển đổi giá trị 'Protocol' sang kiểu chuỗi
df_protocol = df_protocol.astype({"Protocol": str})
df_protocol['Protocol'].value_counts()
```

```
Protocol
6          5295644
17         2508972
0          144132
Name: count, dtype: int64
```

Dựa vào các giá trị của đặc trưng Protocol, các loại giao thức bao gồm:

- 0: HOPOPT (IPv6 Hop-by-Hop Option)
- 6: TCP (Transmission Control Protocol)
- 17: UDP (User Datagram Protocol)

Tỉ lệ các loại giao thức:

```
df_protocol = df_dos.copy()

# Chuyển đổi giá trị 'Protocol' sang kiểu chuỗi
df_protocol = df_protocol.astype({"Protocol": str})
protocol_counts = df_protocol['Protocol'].value_counts()

# Tính tỉ lệ phần trăm cho mỗi giao thức
protocol_percentages = (protocol_counts / protocol_counts.sum()) * 100

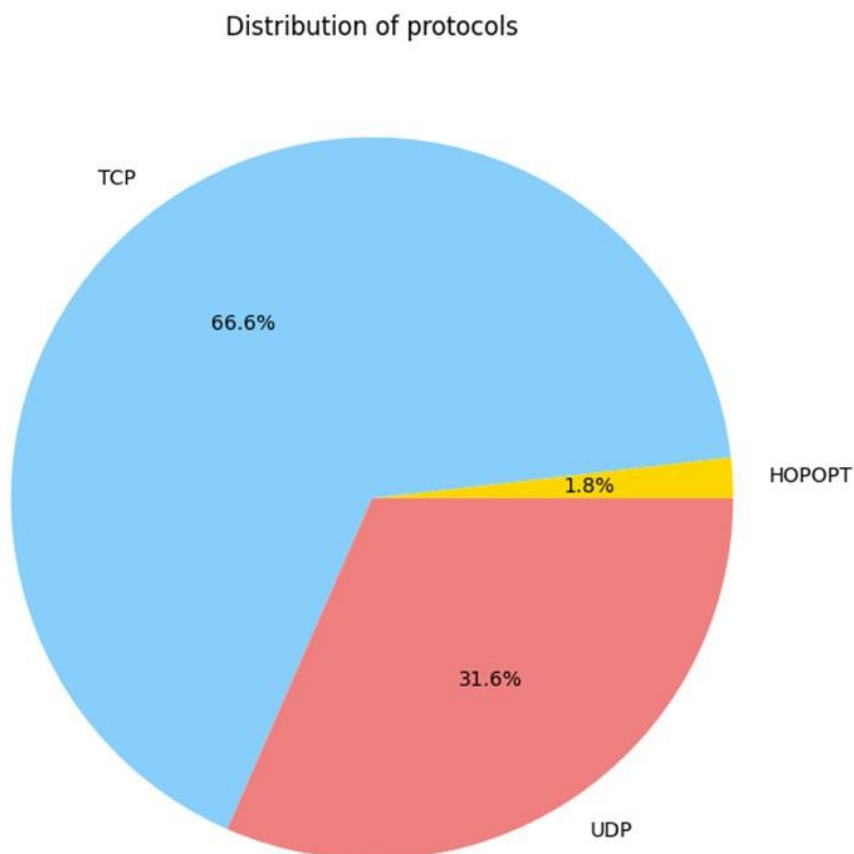
# Tạo biểu đồ cột ngang hiển thị phần trăm
plt.figure(figsize=(12, 8))
```

```

protocol_percentages.plot(kind='barh', color='skyblue')
plt.xlabel('Percentage (%)')
plt.ylabel('Protocol')
plt.title('Percentage of Protocols')
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Hiển thị giá trị phần trăm trên mỗi thanh của biểu đồ
for index, value in enumerate(protocol_percentages):
    plt.text(value, index, f'{value:.2f}%', va='center')
plt.show()

```



Hình 3.6: Biểu đồ cột tỉ lệ phần trăm các loại giao thức

3.2.2. Tiền xử lý dữ liệu

Vì mục tiêu là xây dựng mô hình phát hiện tấn công DoS nên hai tập dữ liệu chính được dùng để huấn luyện là 02-15-2018.csv và 02-16-2018.csv. Các tập dữ liệu còn lại không có nhãn tấn công DoS sẽ không được sử dụng.

a. Làm sạch dữ liệu

- Chuyển đổi các giá trị của mẫu có số vô cùng về dạng NaN:

```
# Replace inf values to nan
df_dos = df_dos.replace([np.inf, -np.inf], np.nan)

# Count nan values
df_dos.isna().sum().sum()
```

- Loại bỏ các mẫu lặp lại, không cần thiết hoặc có chứa giá trị null:

```
# Drop nan values
df_dos.dropna(inplace=True)
df_dos.isna().sum().sum()

# Drop duplicate values
df_dos.drop_duplicates(inplace=True)
df_dos.shape

# Remove row with Label["Label"]
df_dos.drop(df_dos.loc[df_dos["Label"] == "Label"].index, inplace=True)
```

- Loại bỏ các cột không có giá trị:

```
# Danh sách đặc trưng cần giữ lại
features_to_keep = [
    'Bwd Pkt Len Min', 'Subflow Fwd Byts', 'TotLen Fwd Pkts',
    'Fwd Pkt Len Mean', 'Bwd Pkt Len Std', 'Flow IAT Min',
    'Fwd IAT Min', 'Flow IAT Mean', 'Flow Duration',
    'Flow IAT Std', 'Active Min', 'Active Mean',
    'Bwd IAT Mean', 'Fwd IAT Mean', 'Dst Port',
    'Tot Fwd Pkts', 'Tot Bwd Pkts', 'Protocol',
    'Label'
]
df_dos = df_dos[features_to_keep]
```

- Loại bỏ các cột hằng:

```
# Drop constant columns
variances = df_numeric.var(numeric_only=True)
```

```
constant_columns = variances[variances == 0].index
df_numeric.drop(constant_columns, axis=1, inplace=True)
```

- Chuyển đổi kiểu dữ liệu của các cột mang thông tin là số nhưng thuộc dạng object sang dạng số như int64 hoặc float64:

```
def convert_to_numeric(df):
    """Converts all features (except label) from object to float64 or int64.
    Args:
        df: A pandas DataFrame.

    Returns:
        A DataFrame with features converted to float64 (if possible).
    """
    # Select all columns except the label column (assuming 'Label' is the name)
    numeric_cols = df.columns.difference(['Dst Port', 'Protocol', 'Label'])
    # Try converting each column to float, ignoring errors for non-numeric values
    for col in numeric_cols:
        try:
            df[col] = pd.to_numeric(df[col], errors='coerce')
        except:
            pass
    return df

df_numeric = convert_to_numeric(df_dos.copy())
del df_dos
gc.collect()

# Convert Dst Port column to int64
df_numeric = df_numeric.astype({"Dst Port": 'int64'})
df_numeric.info()
```

- Encoding cho những cột mang thông tin không phải số (dạng chữ, object,...):

```
df_dos['Label'] = df_dos['Label'].apply(lambda x: 0 if x.startswith("Benign") else 1)
```

```
df_dos['Label'].value_counts()
```

- Thực hiện One-hot Encoding để chuyển đổi giá trị thành dữ liệu phân loại dạng số:

```
# Process Protocol columns
df_numeric = df_numeric.astype({"Protocol": str})
df_numeric["Protocol"].unique()

# Categorical data to onehot
df_numeric = pd.get_dummies(df_numeric, columns=['Protocol'])
df_numeric.head()

df_numeric = df_numeric.astype({"Protocol_0": 'int64', "Protocol_17": 'int64',
"Protocol_6": 'int64'})
df_numeric.head()
```

b. Cân bằng dữ liệu

Số lượng mẫu có nhãn là 0 (1251404) chiếm nhiều hơn nhãn 1 (196818) gấp 6 lần.

```
Cân bằng dữ liệu

len(df_numeric)

1448222

len(df_numeric[df_numeric['Label'] == 0])

1251404

len(df_numeric[df_numeric['Label'] == 1])

196818
```

Hình 3.7: Số lượng mẫu khi chưa cân bằng dữ liệu

Điều này khiến cho tập dữ liệu bị mất cân bằng. Khi tập dữ liệu không cân bằng, mô hình học máy có thể gặp nhiều vấn đề nghiêm trọng. Độ chính xác tổng thể của mô hình có thể cao một cách giả tạo nếu mô hình chỉ cần dự đoán lớp chiếm đa số. Điều này dẫn đến sự thiên vị trong dự đoán, khiến mô hình không phát hiện được các mẫu thuộc lớp chiếm thiểu số. Do đó, cần sử dụng các kỹ thuật như undersampling,

oversampling, SMOTE, hoặc điều chỉnh trọng số trong hàm mất mát để cân bằng dữ liệu và cải thiện hiệu suất mô hình.

Ta sẽ cân bằng dữ liệu bằng kỹ thuật undersampling. Cụ thể, undersampling sẽ giảm số lượng mẫu của lớp chiếm đa số để cân bằng số lượng mẫu giữa các lớp:

```
df_normal = df_numeric[df_numeric['Label'] == 0]
df_attack = df_numeric[df_numeric['Label'] == 1]

df_normal_downsampled = resample(df_normal, replace=False,
n_samples=len(df_attack), random_state=42)

df_downsample = pd.concat([df_normal_downsampled, df_attack])
```

```
df_downsample = pd.concat([df_normal_downsampled, df_attack])
len(df_downsample)

393636

len(df_downsample[df_downsample['Label'] == 0])

196818

len(df_downsample[df_downsample['Label'] == 1])

196818
```

Hình 3.8: Số lượng mẫu sau khi cân bằng dữ liệu

c. Chia dữ liệu

Chia bộ dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test) theo tỉ lệ 80:20, sử dụng tham số `random_state` để đảm bảo tính nhất quán và tránh lỗi thiên kiến (bias).

```
X = df_main.drop(columns='Label')
y = df_main['Label']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

d. Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là cần thiết trong huấn luyện SVM vì nó đưa các đặc trưng về cùng thang đo, giúp mô hình tìm siêu phẳng tối ưu một cách chính xác. Điều này cải thiện độ chính xác, tăng tốc quá trình huấn luyện và giảm nguy cơ lệch do sự khác biệt về thang đo giữa các đặc trưng.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

e. Giảm chiều dữ liệu bằng Principal Component Analysis

Giảm chiều dữ liệu (dimensionality reduction) bằng Principal Component Analysis (PCA) trước khi huấn luyện mô hình SVM là một kỹ thuật phổ biến để cải thiện hiệu suất và giảm thời gian tính toán. Cụ thể, PCA giúp giảm số lượng đặc trưng trong một tập dữ liệu mà vẫn giữ được những thông tin quan trọng nhất. Điều này được thực hiện bằng cách chuyển đổi các biến ban đầu thành một tập các biến mới, gọi là các thành phần chính (principal components), mà vẫn có tính độc lập và giảm thiểu sự mất mát thông tin.

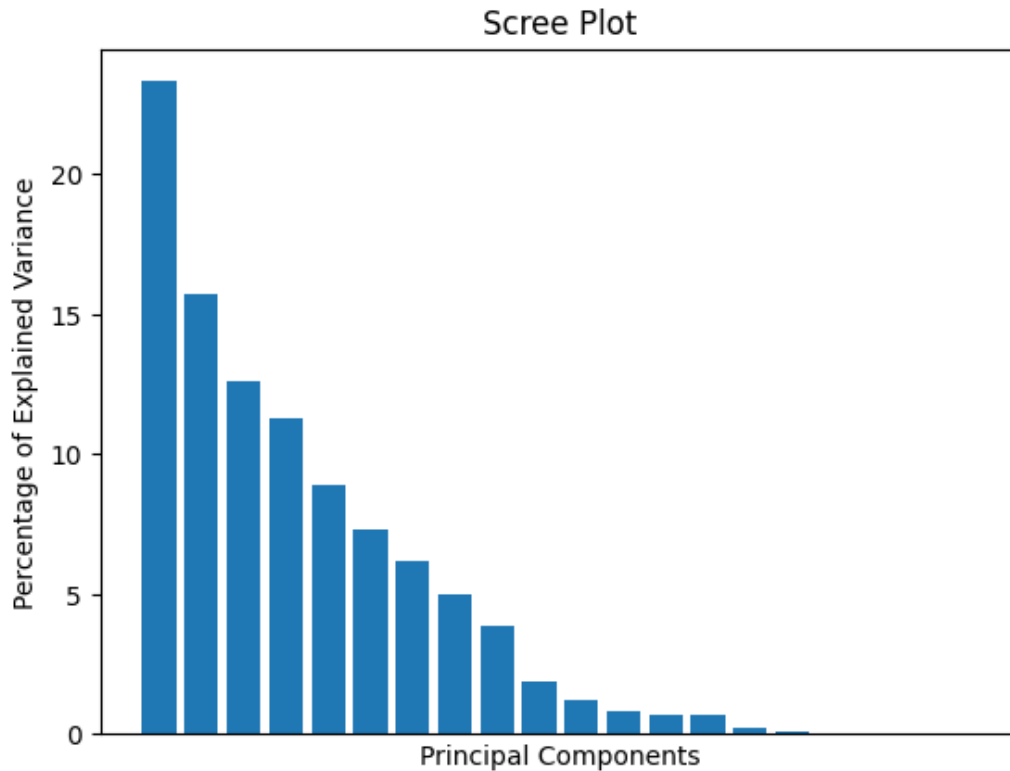
Sau khi tập dữ liệu được tách và chuẩn hóa dữ liệu, tập X huấn luyện và tập X kiểm tra sẽ được chuyển đổi sang dạng PCA. Độ quan trọng của thành phần chính có thể được biểu thị qua Phần trăm phương sai giải thích được (Percentage of Explained Variance), là một khái niệm quan trọng cho biết mỗi thành phần chính giải thích được bao nhiêu phần trăm của tổng phương sai trong dữ liệu ban đầu. Ta có thể tạo đồ thị biểu diễn độ quan trọng của các thành phần chính.

```
pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)

per_var = np.round(pca.explained_variance_ratio_*100, decimals=1)
labels = [str(x) for x in range(1, len(per_var)+1)]

plt.bar(x=range(1, len(per_var)+1), height=per_var)
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
```

```
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Components')
plt.title('Scree Plot')
plt.show()
```



Hình 3.9: Biểu đồ cột phần trăm phương sai giải thích được

Ở đây ta chỉ giữ lại hai thành phần chính cao hơn tất cả các thành phần chính còn lại:

```
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

f. Điều chỉnh siêu tham số (Hyperparameter Tuning)

Hyperparameter tuning, hay điều chỉnh siêu tham số, là một phần quan trọng trong việc xây dựng và tối ưu hóa các mô hình học máy. Siêu tham số là những tham số không được học từ dữ liệu mà được xác định trước quá trình huấn luyện. Việc điều chỉnh các siêu tham số này là cần thiết để cải thiện hiệu suất của mô hình. Một mô hình với các siêu tham số không tối ưu có thể dẫn đến underfitting hoặc overfitting, làm giảm khả năng dự đoán chính xác. Bằng cách tinh chỉnh siêu tham số, chúng ta có thể tìm ra các giá trị tốt nhất, giúp mô hình hoạt động hiệu quả và đạt độ chính xác cao hơn trên tập dữ liệu kiểm tra.

Ngoài ra, hyper parameter tuning cũng giúp tối ưu hóa thời gian huấn luyện và sử dụng tài nguyên tính toán. Các giá trị hyper parameter không phù hợp có thể làm cho quá trình huấn luyện trở nên chậm chạp và tốn kém. Vì vậy, việc tìm ra các cấu hình tối ưu giúp tiết kiệm thời gian và chi phí, đồng thời đảm bảo rằng mô hình có thể được triển khai trong môi trường thực tế một cách hiệu quả.

Trong SVM sử dụng thuật toán SVC, có một số siêu tham số quan trọng cần tối ưu hóa để đạt hiệu suất tốt nhất, chẳng hạn như C, loại hàm kernel, gamma, v.v. Việc tối ưu hóa các siêu tham số này thường được thực hiện thông qua các kỹ thuật như Grid Search, Random Search, hoặc các thuật toán tối ưu hóa Bayesian để tìm ra cấu hình tốt nhất cho mô hình SVC.

Ta sẽ sử dụng hàm GridSearchCV để điều chỉnh siêu tham số. GridSearchCV tìm kiếm các siêu tham số tốt nhất bằng cách sử dụng kỹ thuật cross-validation (CV). Cụ thể, dữ liệu huấn luyện được chia thành K phần (folds) bằng nhau. Quá trình huấn luyện và đánh giá được lặp lại K lần, mỗi lần giữ lại một phần làm validation set và dùng K-1 phần còn lại để huấn luyện mô hình. Hiệu suất của mô hình với các siêu tham số cụ thể được đánh giá trên tập validation của mỗi fold, và kết quả được tính trung bình để đưa ra một đánh giá tổng quát về hiệu suất của các siêu tham số đó. Quá trình này lặp lại cho tất cả các kết hợp siêu tham số trong lưới (grid). Sau khi hoàn thành quá trình cross-validation cho tất cả các kết hợp siêu tham số, GridSearchCV sẽ chọn ra tổ hợp siêu tham số có hiệu suất trung bình tốt nhất dựa trên scoring metric được cung cấp. Nhờ sử dụng cross-validation, GridSearchCV có thể đánh giá hiệu suất của các siêu tham số một cách đáng tin cậy mà không cần một tập kiểm tra riêng biệt, giúp tránh overfitting và đảm bảo rằng các siêu tham số tốt nhất được chọn ra dựa trên khả năng tổng quát hóa của mô hình đối với dữ liệu chưa từng thấy.

Vì quá trình này đòi hỏi công việc huấn luyện và đánh giá nhiều lần nên ta cần phải downsample tập dữ liệu, trích một số lượng mẫu vừa đủ cho việc điều chỉnh siêu tham số:

```
df_normal = df_main[df_main['Label'] == 0]
df_attack = df_main[df_main['Label'] == 1]

df_normal_downsampled = resample(df_normal, replace=False, n_samples=5000,
random_state=42)
len(df_normal_downsampled)

df_attack_downsampled = resample(df_attack, replace=False, n_samples=5000,
random_state=42)
len(df_attack_downsampled)

df_downsample = pd.concat([df_normal_downsampled, df_attack_downsampled])
```

```
len(df_downsample)
```

Sau khi được chia tách và chuẩn hóa dữ liệu. Tập huấn luyện sẽ được sử dụng để điều chỉnh siêu tham số bằng GridSearchCV:

```
%%time
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)

train_pc1_coords = X_train_pca[:, 0]
train_pc2_coords = X_train_pca[:, 1]
pca_train_scaled = StandardScaler().fit_transform(np.column_stack((train_pc1_coords,
train_pc2_coords)))

param_grid = {
    'C': [0.1, 0.5, 1, 10, 100],
    'gamma': ['scale', 1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf']
}

# Tạo scorer dựa trên Hinge Loss
hinge_scorer = make_scorer(hinge_loss, greater_is_better=False)

optimal_params = GridSearchCV(
    SVC(),
    param_grid,
    cv=5,
    scoring=hinge_scorer,
    verbose=3
)

optimal_params.fit(pca_train_scaled, y_train)
print(optimal_params.best_params_)
```

```
print(f'Best score: {optimal_params.best_score_}')
```

Các siêu tham số trong lưới bao gồm:

Tên tham số	Giá trị
C	0.5, 1, 10, 100
gamma	'scale', 1, 0.1, 0.01, 0.001, 0.0001
Kernel	'rbf', 'sigmoid'

Bảng 3.4: Các siêu tham số cho Hyperparameter Tuning

Scoring metric được sử dụng là hàm Hinge Loss. Hinge loss là hàm mất mát (loss function) được thiết kế đặc biệt cho thuật toán SVM và phản ánh trực tiếp mục tiêu tối ưu hóa của mô hình này. Hinge loss không chỉ xem xét nhãn dự đoán đúng hay sai mà còn đánh giá mức độ mà một điểm dữ liệu vi phạm biên phân loại, giúp tối ưu hóa mô hình để có biên phân loại tốt nhất.

3.3. Giai đoạn huấn luyện

Sau khi hoàn tất giai đoạn xử lý dữ liệu, ta tiến hành huấn luyện mô hình. Các thư viện cần thiết cho quá trình huấn luyện và kiểm tra bao gồm:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
accuracy_score, precision_score, recall_score, f1_score, hinge_loss
import matplotlib.pyplot as plt
from sklearn.utils import resample
import matplotlib.colors as mcolors
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.decomposition import PCA
```

Tiến hành huấn luyện mô hình:

```
clf_svm = SVC(random_state=42, C=100, gamma=1, kernel='rbf')
clf_svm.fit(X_train_pca, y_train)
```

Trong đó:

- X_train_pca: chứa các mẫu huấn luyện đã được xử lý.
- y_train: chứa các mẫu của đặc trưng mục tiêu.

- `random_state`: để đảm bảo tính nhất quán và tránh lỗi bias.

Các siêu tham số được chọn từ Hyperparameter Tuning:

- `kernel = 'rbf'`: là Radical Basis Function, một trong những hàm kernel phổ biến nhất cho phép tìm hyperplane ở chiều không gian cao hơn.
- `C = 100`: là thông số điều chỉnh độ nghiêm ngặt của biên phân loại.
- `gamma = 1`: là γ tham số xác định độ ảnh hưởng của mỗi điểm dữ liệu.

```
[15]: %%time

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
# Huấn luyện mô hình SVM với kernel RBF trên dữ liệu PCA
# clf_svm_pca = SVC(random_state=42, **optimal_params.best_params_)
clf_svm_pca = SVC(random_state=42, C=100, gamma=1, kernel='rbf')
clf_svm_pca.fit(X_train_pca, y_train)

CPU times: total: 20min 49s
Wall time: 24min 48s

[15]: SVC
      SVC(C=100, gamma=1, random_state=42)
```

Hình 3.10: Kết quả huấn luyện mô hình SVC

3.4. Giai đoạn kiểm tra

Sau khi hoàn thành giai đoạn huấn luyện, ta sẽ sử dụng mô hình phát hiện đã tạo để dự đoán đối với tập X huấn luyện và tập X kiểm tra. Tập kiểm tra được trích từ 20% của tập dữ liệu ban đầu và đã được chuẩn hóa. Mô hình SVM sẽ tạo ra `y_predict` chứa kết quả dự đoán. Sau đó, ta sẽ so sánh `y_predict` với `y_test` và `y_train`. Kết quả phân loại sẽ được tính toán theo các chỉ số đánh giá hiệu suất của mô hình như Accuracy, Precision, Recall, F1-Score, Confusion Matrix và Hinge Loss. Bên cạnh đó ta sẽ kiểm tra mô hình với một mẫu dữ liệu log mới để xem mô hình phát hiện hiệu quả hay không.

3.4.1. Confusion Matrix

Confusion Matrix (ma trận nhầm lẫn) là một công cụ hữu ích để đánh giá hiệu suất của mô hình học máy trong các bài toán phân loại. Nó cung cấp một cái nhìn chi tiết về cách mà mô hình phân loại các mẫu và các loại dự đoán sai sót mà nó thực hiện. Confusion Matrix thường được sử dụng để tính toán các chỉ số đánh giá như Accuracy, Precision, Recall và F1-Score.

Đối với bài toán phân loại nhị phân Confusion Matrix là một bảng có dạng 2x2, trong đó các thành phần chính bao gồm:

- True Positive (TP): Số lượng các mẫu thực tế thuộc lớp dương (positive class) và được mô hình dự đoán đúng là thuộc lớp dương.
- False Positive (FP): Số lượng các mẫu thực tế thuộc lớp âm (negative class) nhưng lại bị mô hình dự đoán sai thành thuộc lớp dương.
- True Negative (TN): Số lượng các mẫu thực tế thuộc lớp âm và được mô hình dự đoán đúng là thuộc lớp âm.
- False Negative (FN): Số lượng các mẫu thực tế thuộc lớp dương nhưng lại bị mô hình dự đoán sai thành thuộc lớp âm.



Hình 3.11: Code và kết quả chỉ số ma trận nhầm lẫn trên tập kiểm tra

3.4.2. Accuracy

Accuracy (độ chính xác) là một phép đo để đánh giá khả năng của một mô hình phân loại trong học máy. Nó đo lường tỉ lệ phần trăm các mẫu được phân loại đúng so với tổng số mẫu trong tập dữ liệu kiểm tra. Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của ta càng chuẩn xác. Tuy nhiên, trong một số trường hợp cụ thể, nó có thể không đủ để cung cấp thông tin chi tiết về hiệu suất của mô hình trong từng lớp.

$$Accuracy = \frac{TP + TN}{Total\ test\ samples}$$

3.4.3. Precision

Precision đo lường tỷ lệ các dự đoán positive mà mô hình phân loại đúng so với tổng số dự đoán positive của nó. Nghĩa là nó trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng. Một giá trị Precision cao cho thấy mô hình có khả năng dự đoán positive một cách chính xác và ít phát sinh các dự đoán sai positive. Tuy nhiên, chỉ sử dụng Precision mà không xem xét Recall có thể dẫn đến việc bỏ qua những trường hợp positive thực sự mà mô hình không phát hiện được.

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

3.4.4. Recall

Recall đo lường tỷ lệ các mẫu thuộc lớp positive mà mô hình dự đoán đúng so với tổng số các mẫu thuộc lớp positive trong tập dữ liệu. Recall đo lường khả năng của mô hình trong việc phát hiện các mẫu thuộc lớp positive. Một giá trị Recall cao cho thấy mô hình có khả năng bao quát tốt hơn trong việc dự đoán các mẫu positive, ít bỏ sót các mẫu thực sự thuộc lớp positive.

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

3.4.5. F1-Score

F1-score là một chỉ số đánh giá tổng hợp của mô hình phân loại, kết hợp giữa Precision và Recall. Nó là trung bình điều hòa (harmonic mean) của hai chỉ số này. F1-score đo lường sự cân bằng giữa Precision và Recall. Điều này có ý nghĩa trong các bài toán phân loại khi chúng ta muốn một mô hình có cả khả năng dự đoán chính xác (Precision cao) và có khả năng phát hiện tất cả các trường hợp positive thực sự (Recall cao).

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

3.4.6. Hinge Loss

Hinge Loss là một hàm mất mát thường dùng trong các mô hình phân loại nhị phân, đặc biệt là thuật toán SVM. Nó đo lường mức độ chính xác của các dự đoán mô hình dựa trên khoảng cách đến đường quyết định (Decision Boundary). Cụ thể, Hinge Loss tính toán khoảng cách từ các điểm dữ liệu đến đường quyết định của mô hình SVM. Nếu điểm dữ liệu được phân loại đúng và nằm ngoài khoảng cách biên (margin), Hinge Loss là 0. Ngược lại, nếu điểm nằm trong khoảng cách biên hoặc bị phân loại sai, Hinge

Loss tăng lên, tỷ lệ với khoảng cách của điểm đó đến đường quyết định. Hinge Loss là một phần của hàm mục tiêu trong SVM, mục tiêu là tối ưu hóa khoảng cách biên giữa các lớp, tức là tìm đường quyết định với khoảng cách biên lớn nhất.

Hinge Loss cho mỗi điểm dữ liệu (x_i, y_i) được định nghĩa như sau:

$$\text{Hinge Loss} = \max(0, 1 - y_i f(x_i))$$

Trong đó:

- y_i : là nhãn thực tế của điểm dữ liệu
- $f(x_i)$: là giá trị dự đoán của mô hình cho điểm dữ liệu x_i

```
[17]: # Dự đoán nhãn của tập huấn luyện
y_pred = clf_svm_pca.predict(X_train_pca)

# Đánh giá độ chính xác
accuracy = accuracy_score(y_train, y_pred)
precision = precision_score(y_train, y_pred)
recall = recall_score(y_train, y_pred)
f1 = f1_score(y_train, y_pred)
loss = hinge_loss(y_train, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'Hinge Loss: {loss}')

Accuracy: 0.9685622467514322
Precision: 0.9537861094038107
Recall: 0.9848884883410974
F1-Score: 0.9690878093560897
Hinge Loss: 0.5310947959404017
```

Hình 3.12: Code và kết quả các chỉ số đánh giá trên tập huấn luyện

```
[18]: # Dự đoán nhãn của tập kiểm tra
y_pred = clf_svm_pca.predict(X_test_pca)

# Đánh giá độ chính xác
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
loss = hinge_loss(y_test, y_pred)

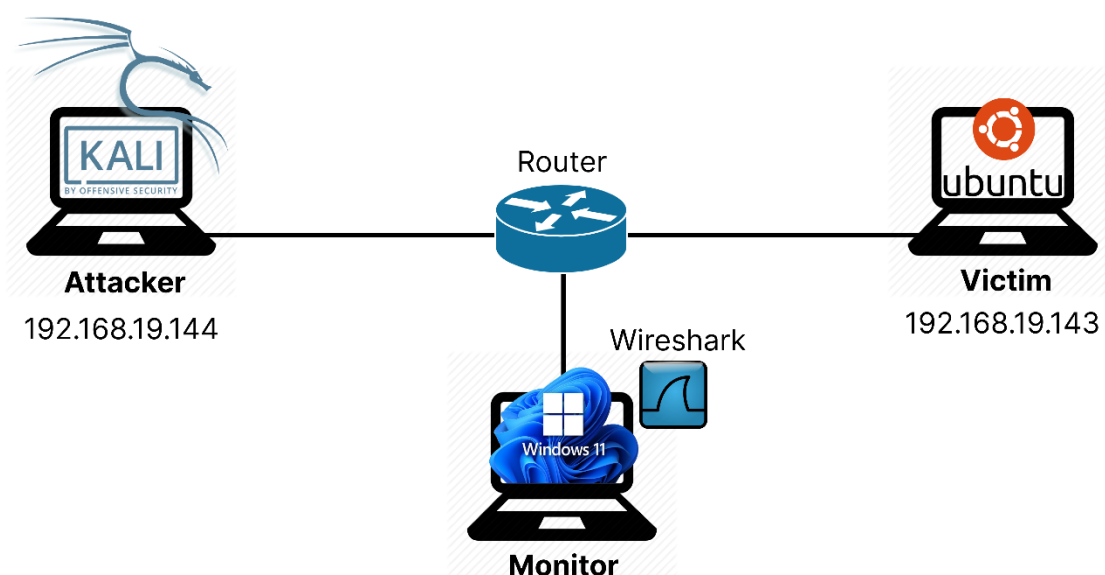
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'Hinge Loss: {loss}')

Accuracy: 0.9682832029265318
Precision: 0.9544094741266349
Recall: 0.9833656001630324
F1-Score: 0.9686711917995559
Hinge Loss: 0.5330886088812112
```

Hình 3.13: Code và kết quả các chỉ số đánh giá trên tập kiểm tra

3.4.7. Mô phỏng tấn công DoS

Dữ liệu log mới được tạo ra từ một cuộc tấn công mô phỏng DoS bằng công cụ Golden Eye. Trong đó, máy Kali sẽ tấn công máy chủ Ubuntu (Apache2) và một máy Windows đóng vai trò giám sát lưu lượng mạng.



Hình 3.14: Mô hình mô phỏng tấn công DoS bằng công cụ Golden Eye

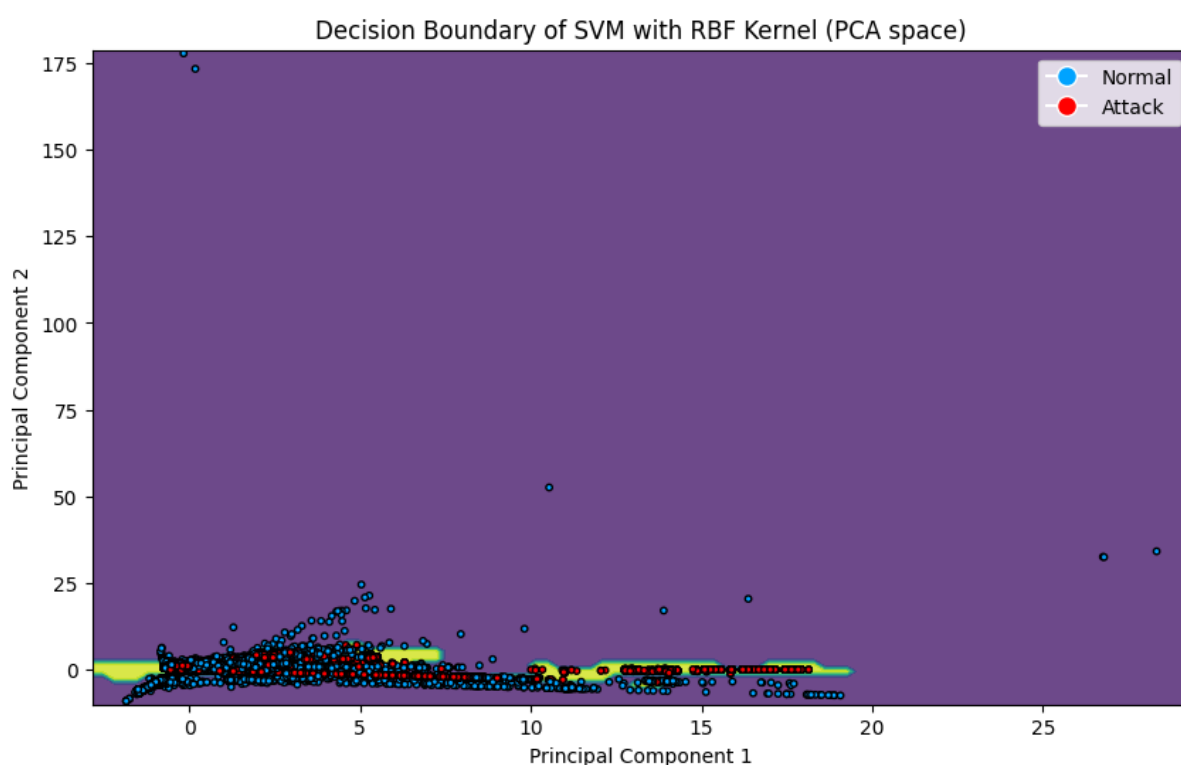
Trong quá trình tấn công, máy Windows 11 giám sát bằng công cụ Wireshark và thu thập được dữ liệu log dưới định dạng pcap. Dữ liệu này sau đó được chuyển sang định dạng csv sao cho các đặc trưng được đồng bộ so với bộ dữ liệu được sử dụng cho công việc phát triển mô hình SVM. Công cụ CICFlowMeter là công cụ phù hợp nhất cho yêu cầu này vì bộ dữ liệu huấn luyện đã được chuyển đổi tương tự. Bên cạnh bộ dữ liệu CSE-CIC-IDS2018 on AWS, công cụ CICFlowMeter cũng được phát triển bởi Viện An ninh Mạng Canada (CIC), một đơn vị của đại học New Brunswick [12].

3.5. Kết quả thực nghiệm

3.5.1. Kiểm tra trên bộ dữ liệu

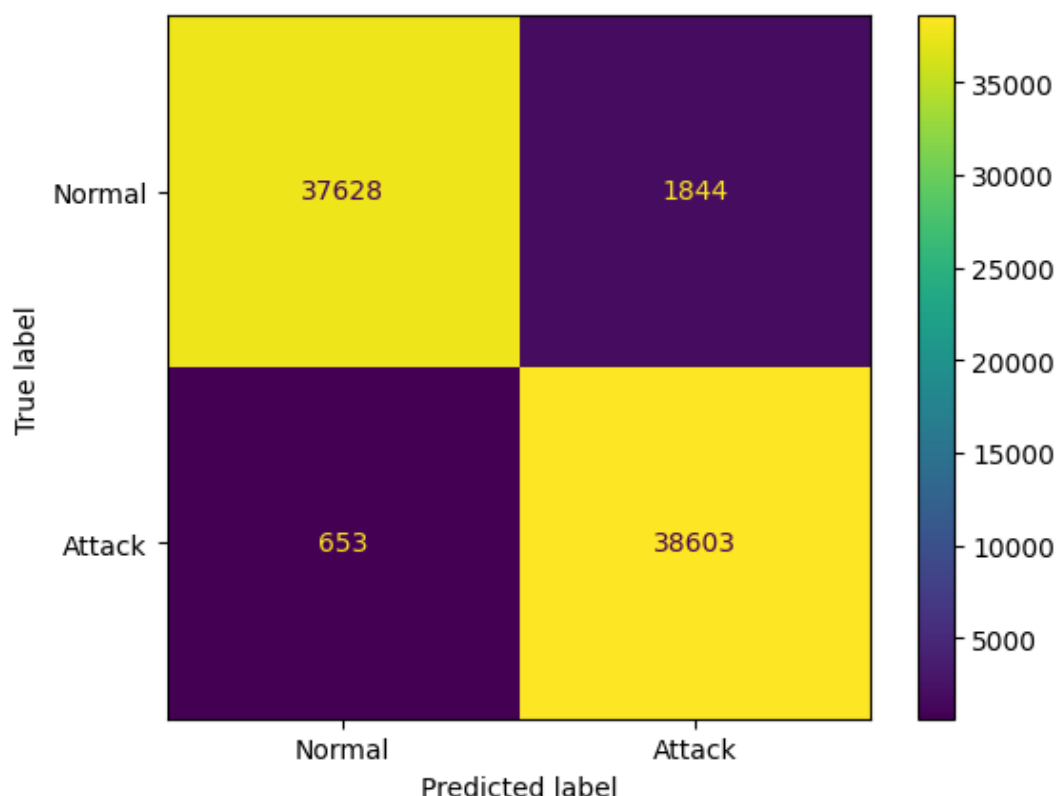
Sau khi thực hiện kiểm tra mô hình, ta được kết quả kiểm tra trên bộ dữ liệu sau:

- Biểu diễn đồ thị:



Hình 3.15: Biểu đồ ranh giới quyết định của SVM với Kernel RBF (không gian PCA)

- Confusion Matrix:



Hình 3.16: Ma trận nhầm lẫn trên tập kiểm tra

Trong đó:

- True Positives (Tấn công được xác định đúng là tấn công): 38603
- True Negatives (Bình thường được xác định đúng là bình thường): 37628
- False Positives (Bình thường bị xác định sai là tấn công): 1844
- False Negatives (Tấn công bị xác định sai là bình thường): 653

Mô hình phân loại chính xác phân lớn các mẫu, với số lượng mẫu tấn công bị nhầm lẫn với mẫu bình thường (653) là rất nhỏ so với tổng số mẫu. Số lượng mẫu bình thường bị nhầm lẫn là mẫu tấn công (1844) cũng khá nhỏ so với tổng số mẫu.

- Các chỉ số hiệu suất trên tập huấn luyện và tập kiểm tra:

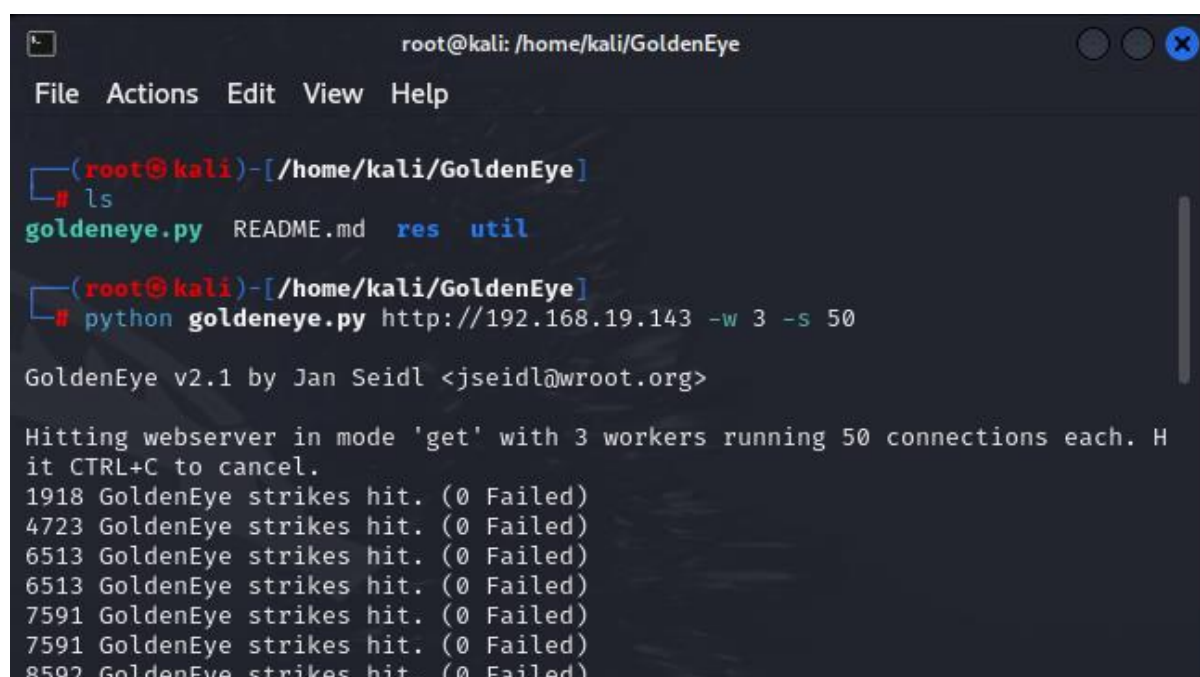
Chỉ số hiệu suất \ Tập dữ liệu	Tập huấn luyện	Tập kiểm tra
Accuracy	96.85622467514322%	96.82832029265318%
Precision	95.37861094038107%	95.44094741266349%
Recall	98.48884883410974%	98.33656001630324%
F1-Score	96.90878093560897%	96.86711917995559%
Hinge Loss	0.5310947959404017	0.5330886088812112

Bảng 3.5: Các chỉ số hiệu suất trên tập huấn luyện và tập kiểm tra

Mô hình SVM hoạt động hiệu quả với tất cả các chỉ số hiệu suất đều ở mức cao trên cả tập huấn luyện và tập kiểm tra. Accuracy, Precision, Recall, và F1-score đều cao, cho thấy mô hình có khả năng phân loại tốt. Không có dấu hiệu của overfitting hay underfitting, vì các chỉ số trên tập huấn luyện và tập kiểm tra gần như giống nhau. Hinge Loss ở mức trung bình cũng cho thấy mô hình phân loại tốt với biên phân cách (margin) khá tốt.

3.5.2. Kiểm tra trên dữ liệu log mới

Dữ liệu log mới được thu thập từ cuộc tấn công mô phỏng. Máy Kali thực hiện tấn công bằng công cụ GoldenEye nhắm vào web server Apache của máy chủ Ubuntu:



```
root@kali: /home/kali/GoldenEye
File Actions Edit View Help

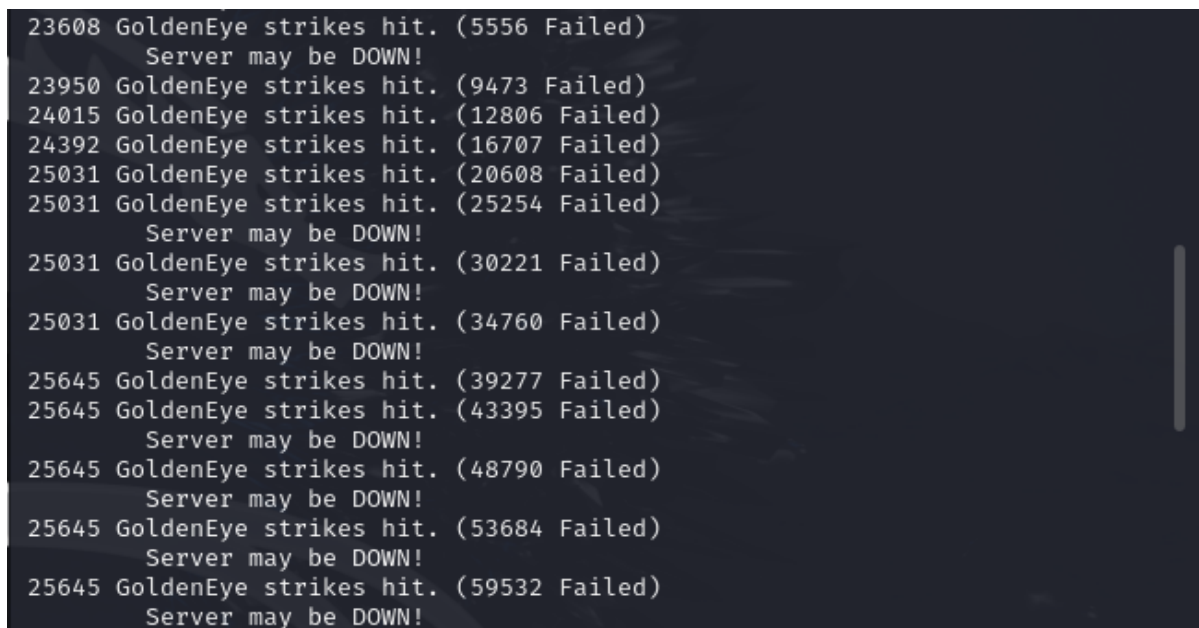
(root@kali)-[/home/kali/GoldenEye]
# ls
goldeneye.py README.md res util

(root@kali)-[/home/kali/GoldenEye]
# python goldeneye.py http://192.168.19.143 -w 3 -s 50

GoldenEye v2.1 by Jan Seidl <jseidl@wroot.org>

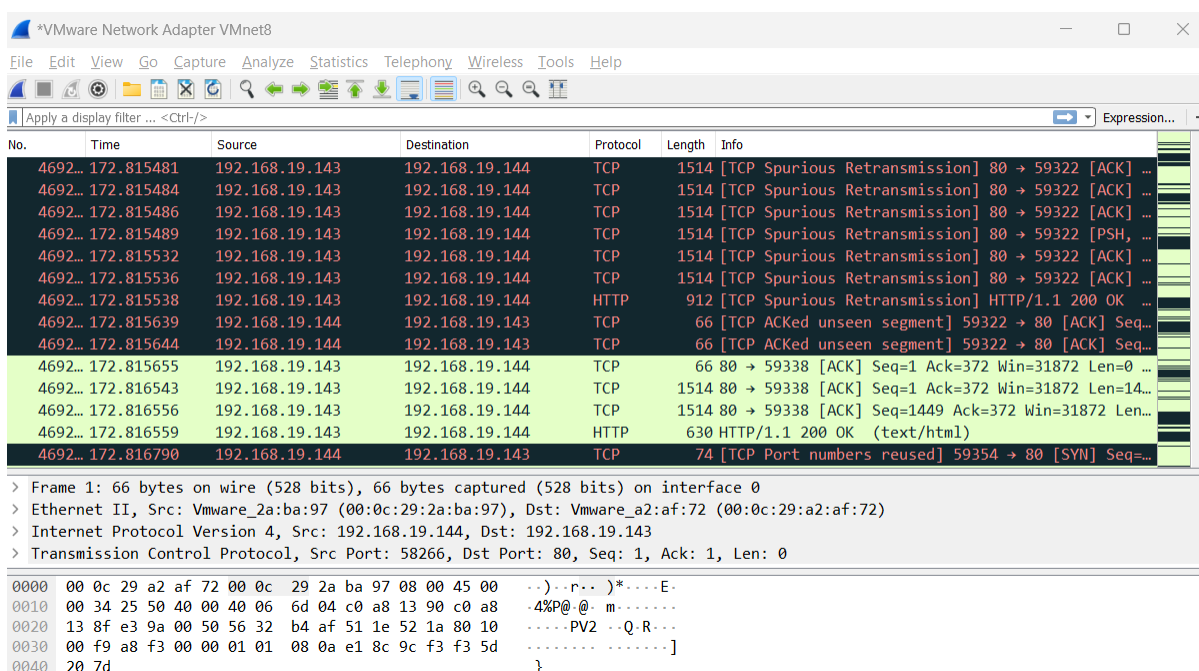
Hitting webserver in mode 'get' with 3 workers running 50 connections each. Hit CTRL+C to cancel.
1918 GoldenEye strikes hit. (0 Failed)
4723 GoldenEye strikes hit. (0 Failed)
6513 GoldenEye strikes hit. (0 Failed)
6513 GoldenEye strikes hit. (0 Failed)
7591 GoldenEye strikes hit. (0 Failed)
7591 GoldenEye strikes hit. (0 Failed)
8592 GoldenEye strikes hit. (0 Failed)
```

Hình 3.17: Thực hiện tấn công DoS vào web server Ubuntu bằng GoldenEye (1)



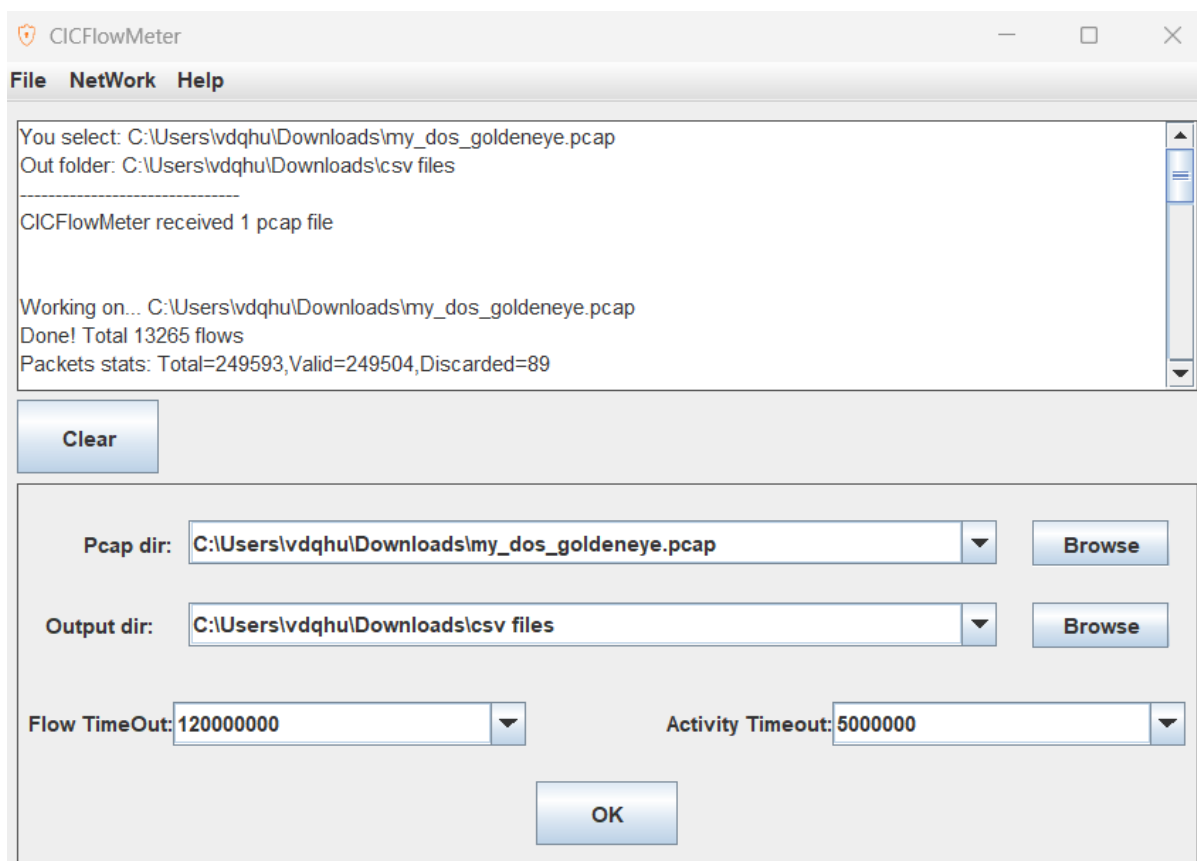
Hình 3.18: Thực hiện tấn công DoS vào web server Ubuntu bằng GoldenEye (2)

Cùng lúc đó, máy giám sát Windows 11 quan sát được các gói tin request của kẻ tấn công và response của nạn nhân:



Hình 3.19: Bắt gói tin dữ liệu tấn công DoS bằng Wireshark

Các gói tin trên được lưu lại thành tập tin pcap. Sau đó công cụ CICFlowMeter chuyển đổi tập tin đó sang dạng csv:



Hình 3.20: Chuyển đổi tập tin pcap sang định dạng csv

Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Durat	Tot Fwd Pk	Tot Bwd Pk	Tot Len Fwd	Tot Len Bwd	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le
1	192.168.1.1	192.168.1.1	80	192.168.1.1	37022	6	26/07/2022	121675	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
2	192.168.1.1	192.168.1.1	80	192.168.1.1	36716	6	26/07/2022	58538	3	2	2012	1448	1448	0	670.6667	729.8694	1448	1448
4	192.168.1.1	192.168.1.1	80	192.168.1.1	37278	6	26/07/2022	65345	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
5	192.168.1.1	192.168.1.1	80	192.168.1.1	37026	6	26/07/2022	122900	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
6	192.168.1.1	192.168.1.1	80	192.168.1.1	35368	6	26/07/2022	84328	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
7	192.168.1.1	192.168.1.1	80	192.168.1.1	36700	6	26/07/2022	81294	3	2	2012	1448	1448	0	670.6667	729.8694	1448	1448
8	192.168.1.1	192.168.1.1	80	192.168.1.1	37016	6	26/07/2022	127571	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
9	192.168.1.1	192.168.1.1	80	192.168.1.1	36692	6	26/07/2022	83269	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
10	192.168.1.1	192.168.1.1	80	192.168.1.1	37006	6	26/07/2022	129164	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
11	192.168.1.1	192.168.1.1	80	192.168.1.1	36728	6	26/07/2022	90032	3	2	2012	1448	1448	0	670.6667	729.8694	1448	1448
12	192.168.1.1	192.168.1.1	80	192.168.1.1	36684	6	26/07/2022	88972	3	2	2012	1448	1448	0	670.6667	729.8694	1448	1448
13	192.168.1.1	192.168.1.1	80	192.168.1.1	36996	6	26/07/2022	131063	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
14	192.168.1.1	192.168.1.1	80	192.168.1.1	37036	6	26/07/2022	103976	8	3	9534	1448	1448	0	1191.75	525.6011	1448	1448
15	192.168.1.1	192.168.1.1	80	192.168.1.1	36832	6	26/07/2022	132248	8	4	9534	1448	1448	0	1191.75	525.6011	1448	1448
16	192.168.1.1	192.168.1.1	80	192.168.1.1	37048	6	26/07/2022	22011	3	2	1975	1448	1448	0	658.3333	732.8795	1448	1448
17	192.168.1.1	192.168.1.1	80	192.168.1.1	35660	6	26/07/2022	51907	2	3	2012	1448	1448	564	1006	625.0824	1448	1448
18	192.168.1.1	192.168.1.1	80	192.168.1.1	35904	6	26/07/2022	163584	2	3	2012	1448	1448	564	1006	625.0824	1448	1448
19	192.168.1.1	192.168.1.1	80	192.168.1.1	35932	6	26/07/2022	129616	2	3	2012	1448	1448	564	1006	625.0824	1448	1448
20	192.168.1.1	192.168.1.1	37048	192.168.1.1	80	6	26/07/2022	25295	1	1	0	0	0	0	0	0	0	0

Hình 3.21: Nội dung dữ liệu log trong định dạng csv

Ta sẽ chỉ lấy 2 mẫu trong dữ liệu log trên cùng với các đặc trưng cần thiết để kiểm tra mô hình đã xây dựng.

```
[1]: import numpy as np
import pandas as pd
import warnings
pd.set_option('display.max_columns', None)
warnings.filterwarnings('ignore')
import joblib
```

Hình 3.22: Import các thư viện cần thiết cho công việc dự đoán mẫu dữ liệu

Đoạn mã dưới đây phục vụ cho việc tạo mẫu và gộp nó thành một dataframe:

```
data_1 = {
    'Bwd Pkt Len Min': [0], 'Subflow Fwd Byts': [0], 'TotLen Fwd Pkts': [0], 'Fwd Pkt
    Len Mean': [0],
    'Bwd Pkt Len Std': [0], 'Flow IAT Min': [0], 'Fwd IAT Min': [0], 'Flow IAT Mean':
    [25295],
    'Flow Duration': [25295], 'Flow IAT Std': [0], 'Active Min': [0], 'Active Mean': [0],
    'Bwd IAT Mean': [0], 'Fwd IAT Mean': [0], 'Dst Port': [80], 'Tot Fwd Pkts': [1],
    'Tot Bwd Pkts': [1], 'Protocol_0': [0], 'Protocol_17': [0], 'Protocol_6': [1]
}

data_2 = {
    'Bwd Pkt Len Min': [0], 'Subflow Fwd Byts': [0], 'TotLen Fwd Pkts': [0], 'Fwd Pkt
    Len Mean': [0],
    'Bwd Pkt Len Std': [0], 'Flow IAT Min': [4750], 'Fwd IAT Min': [0], 'Flow IAT
    Mean': [22863.5],
    'Flow Duration': [45727], 'Flow IAT Std': [25616.357362045], 'Active Min': [0],
    'Active Mean': [0],
    'Bwd IAT Mean': [25616.357362045], 'Fwd IAT Mean': [0], 'Dst Port': [80], 'Tot
    Fwd Pkts': [0],
    'Tot Bwd Pkts': [3], 'Protocol_0': [0], 'Protocol_17': [0], 'Protocol_6': [1]
}

df_1 = pd.DataFrame(data_1)
df_2 = pd.DataFrame(data_2)
```

```
# Gộp hai dataframe thành một
df_combined = pd.concat([df_1, df_2], ignore_index=True)
df_combined.head()
```

[2]:

Bwd Pkt Len Min	Subflow Fwd Byts	TotLen Fwd Pkts	Fwd Pkt Len Mean	Bwd Pkt Len Std	Flow IAT Min	Fwd IAT Min	Flow IAT Mean	Flow Duration	Flow IAT Std
0	0	0	0	0	0	0	25295.0	25295	0.000000
0	0	0	0	0	4750	0	22863.5	45727	25616.357362

Hình 3.23: Kết quả dataframe trích từ dữ liệu log mô phỏng tấn công DoS (1)

Active Min	Active Mean	Bwd IAT Mean	Fwd IAT Mean	Dst Port	Tot Fwd Pkts	Tot Bwd Pkts	Protocol_0	Protocol_17	Protocol_6
0	0	0.000000	0	80	1	1	0	0	1
0	0	25616.357362	0	80	0	3	0	0	1

Hình 3.24: Kết quả dataframe trích từ dữ liệu log mô phỏng tấn công DoS (2)

Để dự đoán hai mẫu trên, đầu tiên ta cần mô hình StandardScaler và mô hình PCA đã được sử dụng trong giai đoạn tiền xử lý, cuối cùng là mô hình dự đoán SVM đã tạo trong giai đoạn huấn luyện.

Các mô hình trong giai đoạn xử lý dữ liệu và huấn luyện mô hình trên có thể được xuất ra file bằng thư viện joblib:

```
import joblib

# joblib.dump([Mô hình], ' [Nơi lưu file] ')
joblib.dump(scaler, '../Test/scaler_model.joblib')

# Lưu mô hình pca vào một file
joblib.dump(pca, '../Test/pca_model.joblib')
```

```
# Lưu mô hình SVM vào một file
joblib.dump(clf_svm_pca, './Test/svm_model.joblib')
```

Từ đó, ta có thể load các mô hình đó để chuẩn hóa dữ liệu, chuyển dữ liệu sang dạng PCA để đồng bộ với mô hình dự đoán và kiểm tra xem mô hình dự đoán chính xác hay không.

```
[3]: scaler = joblib.load('scaler_model.joblib')
     df_scaled = scaler.transform(df_combined)

[4]: pca = joblib.load('pca_model.joblib')
     df_pca = pca.transform(df_scaled)
```

Hình 3.25: Tải các mô hình chuẩn hóa dữ liệu và PCA bằng joblib

```
[5]: # Tải mô hình SVM
     model = joblib.load('svm_model.joblib')

[6]: predicted_label = model.predict(df_pca)
     print(predicted_label)

[1 1]
```

Hình 3.26: Tải mô hình SVM bằng joblib và kết quả dự đoán mẫu dữ liệu

Kết quả cho thấy cả hai mẫu đều cho ra nhãn 1 (0 – bình thường, 1 – tấn công), nghĩa là mô hình đã phát hiện thành công tấn công DoS.

CHƯƠNG 4: KẾT LUẬN VÀ ĐÁNH GIÁ

4.1. Kết quả đạt được

4.1.1. Về mặt lý thuyết

Nắm được nền tảng lý thuyết về tấn công mạng, hệ thống phát hiện tấn công, học máy và thuật toán Support Vector Machine (SVM). Cụ thể, đã làm rõ các khái niệm về tấn công mạng, các giai đoạn và loại hình tấn công, cũng như lý thuyết về malware và các kỹ thuật né tránh và lý thuyết tổng quan về hệ thống phát hiện tấn công mạng IDS.

Về học máy, đề tài đã trình bày chi tiết các thuật toán học máy cơ bản và đi sâu vào phân tích SVM, bao gồm lý thuyết, nguyên lý hoạt động và các ưu nhược điểm của thuật toán này trong việc phát hiện tấn công mạng.

4.1.2. Về mặt thực tiễn

Trên cơ sở lý thuyết đã nghiên cứu, đề tài đã triển khai và thử nghiệm một mô hình phát hiện tấn công mạng sử dụng thuật toán SVM. Mô hình được huấn luyện và kiểm tra trên tập dữ liệu thực tế, với các bước xử lý dữ liệu, tiền xử lý và tối ưu hóa tham số được thực hiện cẩn thận. Kết quả thử nghiệm cho thấy mô hình đạt được độ chính xác và hiệu quả cao trong việc phát hiện các loại tấn công mạng DoS với các công cụ khác nhau. Bên cạnh đó, đề tài còn triển khai tấn công mô phỏng DoS bằng công cụ Golden Eye và dự đoán mẫu dữ liệu tấn công bằng mô hình đã phát triển.

4.2. Đánh giá

Mô hình SVM được phát triển đã cho thấy tiềm năng lớn trong việc ứng dụng vào thực tế. Các chỉ số hiệu suất cao và dự đoán đúng mẫu dữ liệu tấn công mô phỏng chứng minh rằng mô hình có khả năng phân loại và phát hiện tấn công mạng một cách hiệu quả. Đặc biệt, mô hình không gặp vấn đề về overfitting hay underfitting, khi mà các chỉ số hiệu suất trên tập huấn luyện và tập kiểm tra rất tương đồng. Tuy nhiên, vẫn cần tiếp tục nghiên cứu và cải tiến để đối phó với các hình thức tấn công ngày càng tinh vi. Cần lưu ý đến việc giảm thiểu các cảnh báo sai và nâng cao độ chính xác trong những trường hợp đặc biệt để mô hình có thể áp dụng rộng rãi hơn trong các hệ thống mạng thực tế.

4.3. Hướng phát triển

Trong tương lai, đề tài có thể được mở rộng và phát triển theo các hướng sau:

- Thu thập và sử dụng các tập dữ liệu phong phú hơn từ nhiều nguồn khác nhau để cải thiện độ chính xác và khả năng tổng quát của mô hình.
- Kết hợp SVM với các thuật toán học máy khác hoặc sử dụng các phương pháp ensemble để nâng cao hiệu quả phát hiện tấn công.
- Nghiên cứu và áp dụng các kỹ thuật tiền xử lý dữ liệu mới để tối ưu hóa quá trình huấn luyện mô hình, đặc biệt là trong việc xử lý các đặc trưng phức tạp của dữ liệu mạng.

- Triển khai mô hình trong các hệ thống mạng thực tế để kiểm tra và đánh giá hiệu suất trong điều kiện vận hành thực tế, từ đó có những điều chỉnh phù hợp nhằm nâng cao hiệu quả.
- Nâng cấp mô hình để có thể giám sát và phát hiện tấn công mạng trong thời gian thực, giúp nâng cao khả năng phản ứng nhanh chóng và hiệu quả đối với các mối đe dọa, đồng thời giảm thiểu thiệt hại có thể gây ra bởi các cuộc tấn công.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Cyfirma, "Malware Detection: Evasion Techniques," Cyfirma, 13 tháng Chín 2023. [Trực tuyến]. <https://www.cyfirma.com/research/malware-detection-evasion-techniques/>. [Đã truy cập 24 tháng Sáu 2024].
- [2] A. Hosseini, "Ten process injection techniques: A technical survey of common and trending process injection techniques," 18 tháng Sáu 2017. [Trực tuyến]. <https://www.elastic.co/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process>. [Đã truy cập 24 tháng Sáu 2024].
- [3] N-able, "Intrusion Detection System (IDS): Signature vs. Anomaly-Based," N-able, 15 tháng Ba 2021. [Trực tuyến]. <https://www.n-able.com/blog/intrusion-detection-system>. [Đã truy cập 25 tháng Sáu 2024].
- [4] P. Check, "Classification of Intrusion Detection Systems," Check Point, [Trực tuyến]. <https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/>. [Đã truy cập 25 tháng Sáu 2024].
- [5] Nguyễn Trọng Hưng, Hoàng Xuân Diệu, Vũ Xuân Hạnh, "Phát hiện botnet dựa trên phân loại tên miền sử dụng các kỹ thuật học máy," *Hội thảo lần thứ III: Một số vấn đề chọn lọc về an toàn thông tin, Đà Nẵng*, 2018.
- [6] Trần Đắc Tốt, Nguyễn Trung Kiên, Trương Hữu Phúc, Lê Ngọc Sơn, Trần Thị Bích Vân, "Phát hiện tấn công SQL Injection bằng học máy," *Tạp chí Khoa học Công nghệ và Thực phẩm*, 13 tháng Bảy 2022.
- [7] Rachid Tahri, Youssef Balouki, Abdessamad Jarrar, Abdellatif Lasbahani, "Intrusion Detection System Using machine learning Algorithms," *ITM Web Conf.*, vol. 46, p. 4, 2022.
- [8] M. Kazim, *Network Anomaly Detection Models Using Algorithms*, Mangalore University, 2022.
- [9] Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani, "CSE-CIC-IDS2018 on AWS," tháng Một 2018. [Trực tuyến]. <https://www.unb.ca/cic/datasets/ids-2018.html>. [Đã truy cập 20 tháng Bảy 2024].
- [10] Solarmainframe, "IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)," 2018. [Trực tuyến]. <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv/data>. [Đã truy cập 20 tháng Bảy 2024].
- [11] Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion," *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, pp. 108-116, 2018.

- [12] C. I. f. Cybersecurity, "CICFlowMeter," 2018. [Trực tuyến]. <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>. [Đã truy cập 27 tháng Bảy 2024].