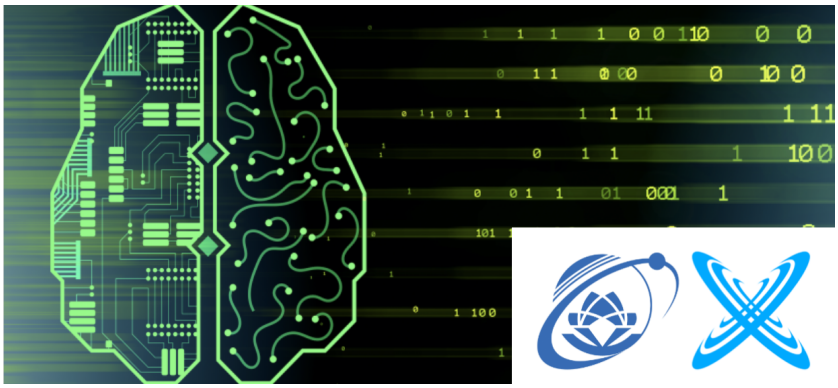
VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORKING AND COMMUNICATION



SUBJECT : MACHINE LEARNING FOR INFORMATION SECURITY

A report of

Crafting Adversarial Example to Bypass Flow-ML-based Botnet Detector via RL

INSTRUCTOR : **Phan Thế Duy**

HO CHI MINH CITY, JUNE 2023

Thành viên nhóm

STT	Họ Tên	MSSV	Phân công
1	Đỗ Quang Thắng	20521893	Threat Model-Sytem Framwork, Demo.
2	Nguyễn Đoàn Thiên Cung	20521146	Poster, Information, Related work.
3	Vũ Trọng Nghĩa	20520651	Experimental Setup, Research Detector.

Mục lục

1 Introduction

1.1 Overview Adversarial

Học máy (Machine learning - ML) đã đóng góp đáng kể vào sự phát triển của công nghệ phát hiện botnet. Khác với các phương pháp phát hiện dựa trên chữ ký, phát hiện bất thường dựa trên học máy có thể nhận diện lưu lượng tạo ra bởi phần mềm độc hại một cách hiệu quả và chính xác khi nhận ra các mẫu hành vi nhất định. Đặc biệt, sự tương đồng không gian và thời gian đáng kể của botnet khiến chúng dễ dàng bị phát hiện bởi các mô hình dựa trên học máy. Ngoài ra, các phương pháp dựa trên học máy có thể xác định các họ botnet không rõ và phù hợp hơn với việc xử lý dữ liệu lưu lượng mạng quy mô lớn.
Không có gì ngạc nhiên khi những kẻ tấn công đã bắt đầu tìm kiếm các phương pháp và kỹ thuật để cho phép họ vượt qua các hệ thống phát hiện và tránh được phát hiện dựa trên hành vi (behavior-based). Ví dụ, những kẻ tấn công có thể thay đổi địa chỉ IP tương ứng với tên miền của máy chủ điều khiển và kiểm soát (CnC) một cách thường xuyên để tránh phát hiện từ danh sách đen IP; sử dụng các giao thức tầng ứng dụng (HTTP, DNS) hoặc các dịch vụ web bên ngoài (Twitter, Facebook) cho việc giao tiếp CnC để tránh tường lửa; và mã hóa các giao tiếp CnC để tránh các trình phát hiện botnet dựa trên tải lượng. Tuy nhiên, những phương pháp tránh này không thể vượt qua các trình phát hiện botnet dựa trên luồng và học máy (Flow-ML-based) được sử dụng rộng rãi, đó là những trình phát hiện dựa trên các đặc điểm thống kê của dòng. Hơn nữa, những phương pháp này yêu cầu các sửa đổi nguồn trực tiếp phức tạp hoặc rộng rãi, do đó đặt nhu cầu cao đối với những kẻ tấn công. Một cách có thể thực hiện để vượt qua được trình phát hiện botnet dựa trên học máy là tấn công vào mô hình phát hiện học máy bằng cách tìm lỗi trong hệ thống phát hiện. Szegedy và đồng nghiệp đã phát hiện ra rằng các mô hình học máy có hiệu suất tốt dễ bị tấn công bởi các cuộc tấn công đối kháng (adversarial attacks) dưới dạng thêm các biến động nhỏ vào đầu vào để đánh lừa mô hình và cho ra đầu ra sai.

Trong bài báo này, một phương pháp dựa trên học tăng cường (reinforcement learning - RL) để vượt qua được trình phát hiện botnet dựa trên dòng lưu trình (flow-level) dựa trên học máy (ML) được nhóm tác giả giới thiệu. Họ đã cố gắng đánh lừa trình phát hiện dựa trên học máy bằng cách thêm một số biến động vào dòng botnet. Cụ thể, tác giả mô hình hóa sự thay đổi của dòng botnet như là một vấn đề quyết định tuần tự, cho phép đại diện RL học chiến lược tối ưu cho việc thay đổi trong suốt một chuỗi tương tác với trình phát hiện botnet. Để đảm bảo tính khả dụng chức năng, tác giả thiết kế không gian hành động (action space) chứa 14 hoạt động tăng dần, mỗi hoạt động chỉ thêm một gói tin được chế tạo một cách cẩn thận vào dòng gốc nhằm thay đổi một số đặc tính của dòng lưu trình. Trình phát hiện coi những đặc tính này là phân biệt, nhưng điều này không nhất thiết là một chỉ báo nguyên nhân của lưu lượng lành mạnh (benign traffic). Hơn nữa, việc thêm gói tin là một hoạt động ở tầng vận chuyển, trong khi các chức năng độc hại thường được đóng gói ở tầng ứng dụng. Do đó, có thể đảm bảo rằng chức năng ban đầu của lưu lượng độc hại sẽ không bị thay đổi. Các ưu điểm của phương pháp tấn công của nhóm tác giả bao gồm

(1) đây là một cuộc tấn công hộp đen (black box), phù hợp hơn với các kịch bản tấn công thực tế hơn các phương pháp khác;

(2) nó có tính chung và có thể sử dụng mà không cần phải tính đạo hàm của hàm mất mát của trình phát hiện;

(3) nó là "plug and play", tác nhân (agent) RL có thể tồn tại như một proxy, cuộc tấn công có chi phí tránh né thấp và phù hợp cho bất kỳ họ botnet nào. Thông qua các thử nghiệm mở rộng, nhóm tác giả đã chứng minh rằng trình phát hiện botnet dựa trên học máy hiện tại là dễ bị tấn công. Các kẻ tấn công có thể tránh được phát hiện chỉ bằng cách thêm một vài gói tin vào dòng botnet với chi phí tương đối nhỏ và không cần bất kỳ kiến thức trước đó.

2 Related Work

2.1 Adversarial machine learning

Trong bài báo này, tác giả nhóm các phương pháp tấn công đối kháng gần đây thành ba kịch bản tấn công dựa trên các mức độ thông tin khác nhau của kẻ tấn công về hệ thống bị tấn công: kiến thức hoàn hảo (perfect knowledge - PK), kiến thức giới hạn (limited knowledge - LK) và không có kiến thức (zero knowledge - ZK).

Perfect knowledge: Đây là tấn công trắng (white-box attack). Trong kịch bản này, kẻ tấn công có đầy đủ thông tin về Detector. Mục tiêu của kẻ tấn công là giảm thiểu hàm phân loại mẫu. Trong [40], L-BFGS được sử dụng để giải quyết vấn đề tối ưu hóa tìm giá trị nhỏ nhất của biến động cần thiết. CW attack [13] thiết kế một hàm mất mát mới với giá trị nhỏ ở phần anti-sample nhưng giá trị lớn hơn ở mẫu sạch để có thể tìm kiếm ví dụ phản đối bằng cách tối thiểu hóa hàm mất mát. FSGM [18] giả định rằng hàm mất mát trong khu vực gần mẫu sạch là tuyến tính. Deepfool [28] là phương pháp đầu tiên sử dụng chuẩn L2 để giới hạn kích thước nhiễu và đạt được biến động tối thiểu. Universal perturbation [27] mở rộng DeepFool để tạo ra các biến động không phụ thuộc vào hình ảnh và phổ quát.

Limited knowledge (LK): Đây là tấn công xám (gray-box attack). Kẻ tấn công chỉ có thông tin hạn chế về Detector và không thể sử dụng các phương pháp dựa trên đạo hàm. Tuy nhiên, bằng cách biết loại hoặc không gian đặc trưng của mô hình, kẻ tấn công có thể đánh lừa Detector bằng cách tìm một tập hợp các tính năng không đủ phân biệt thông qua các tính năng cấu trúc của mô hình, không gian đặc trưng hoặc điểm phản hồi.

Apruzzese et al. [6] đã sử dụng một công cụ giải quyết chương trình tuyến tính nguyên để xây dựng một ví dụ phản đối tối ưu để vượt qua các trình phát hiện botnet dựa trên lưu lượng và random forest. Papernot et al. [31] đã huấn luyện một mô hình thay thế sau khi thực hiện tăng cường dữ liệu dựa trên đạo hàm Jacobian để mô phỏng chính xác ranh giới quyết định của trình phát hiện.

Zero knowledge (ZK): Đây là tấn công đen (black-box attack). Kẻ tấn công không có bất kỳ thông tin nào về mô hình ngoại trừ quyết định nhị phân của Detector. Kẻ tấn công chỉ có thể đánh lừa Detector thông qua trial and error hoặc bằng cách tạo ra các mẫu đối nghịch đối với một bộ phân loại. Ý tưởng cơ bản là nếu các mẫu đối nghịch có thể bypass được mỗi mô hình trong bộ sưu tập, thì chúng có thể tránh được mọi Detector.

MalGAN [21] giới thiệu GAN để tạo ra phần mềm độc hại PE nhằm tránh qua một trình phát hiện hộp đen. Bộ phân biệt GAN là một mô hình trình phát hiện thay thế được xây dựng bởi kẻ tấn công, trong khi bộ tạo được sử dụng để tạo ra các mẫu phần mềm độc hại đối kháng. Phương pháp boundary attack [11] bắt đầu từ một biến động đối kháng lớn và sau đó cố gắng giảm kích thước nó trong khi vẫn giữ tính đối kháng bằng cách thực hiện một bước đi ngẫu nhiên dọc theo ranh giới giữa khu vực đối kháng và khu vực không đối kháng.

Table 1: Taxonomy of the latest academic studies on adversarial attack methods.

PK/LK/ZK	Method	Object	Disadvantage
PK	L-BFGS [40]	image	Unable to handle non-differentiable loss functions
	FSGM [18] [24]	image/malware	
	SLEIPNIR [4]	Windows PE	
	DeepFool [28]	image	
	JBSM [32] [19]	image/malware	
	C&W attack [13]	image	
	ATN [7]	image	
	Adv_MalConv [22]	malware	
	Universal perturbation [27]	image	
LK-score feedback	Train substitute model [31]	image	Targeting against a particular type of ML model
LK-model type	Evade-RF [6]	botnet flow	
ZK	MalGAN [21]	malware	Actually belong to LK scenario, the attacker needs to know the complete feature space
ZK	EvadeHC [14]	pdf malware	Need help with verifying information
ZK	Boundary attack [11]	image	Need the most iterations to converge

PK: Perfect knowledge; LK: Limited knowledge; ZK: Zero knowledge.

2.2 Botnet Evasion

Trong khi phương pháp phát hiện botnet đang liên tục được cải tiến, các kẻ tấn công cũng đang khám phá các kỹ thuật để tránh được trình phát hiện botnet dựa trên học máy. Các kỹ thuật tránh né của botnet truyền thống làm cho lưu lượng CC khó phát hiện bằng cách mã hóa lưu lượng, ẩn thông tin CnC trong các trường dư thừa của các giao thức TCP/IP, hoặc sử dụng mạng xã hội trực tuyến (OSN) để xây dựng các kênh ngầm .Tuy nhiên, những phương pháp này yêu cầu các sửa đổi phức tạp đối với kiến trúc hoặc mã nguồn của botnet, đòi hỏi năng lực cao từ các người điều khiển botnet và cũng ảnh hưởng đến tính khả dụng và giá trị thị trường của botnet. Với sự ứng dụng rộng rãi của học máy trong lĩnh vực phát hiện botnet ,các nhà nghiên cứu bảo mật dần tìm kiếm các phương pháp học máy đối kháng (AML - adversarial machine learning) để né tránh các trình phát hiện lưu lượng botnet. Đó là xây dựng các mẫu đối đầu với lưu lượng botnet bằng cách thêm biến động vào các mẫu lưu lượng botnet, qua đó tránh qua trình phát hiện lưu lượng botnet dựa trên học máy. Những phương pháp này có thể được thực hiện bằng cách áp dụng các luồng proxy thay vì sửa đổi mã nguồn botnet, điều này có vẻ là một giải pháp hấp dẫn và chi phí thấp hơn. Hơn nữa, các phương pháp trốn tránh dựa trên học máy đối kháng (AML) có thể được chia thành hai loại khác nhau theo đầu ra khác nhau. Feature space attack là một phương pháp tấn công đối nghịch trong học máy, được sử dụng để đánh lừa các detector botnet dựa trên học máy. Phương pháp này tập trung vào việc thay đổi các đặc trưng (features) của các mẫu lưu lượng truy cập mạng của botnet để tạo ra các mẫu tấn công giả mạo. Khi các đặc trưng được thay đổi, các mẫu tấn công giả mạo này có thể tránh được các detector botnet dựa trên học máy. Các phương pháp tấn công đối nghịch trong không gian đặc trưng thường chỉ tạo ra các vector đặc trưng tấn công, chứ không phải các mẫu lưu lượng truy cập mạng thực tế. Vì vậy, chúng không gây ra các mối đe dọa bảo mật thực tế, mà chỉ được sử dụng để chứng minh khả năng phát hiện lỗ hổng của các detector botnet dựa trên học máy. EvadeRF cố gắng làm cho lưu lượng botnet khác với các luồng độc hại chứa trong tập dữ liệu huấn luyện của trình phát hiện bằng cách thay đổi một chút các đặc trưng thống kê cấp luồng, chẳng hạn như thời lượng luồng, cũng như số byte và số gói trao đổi. Tác giả đã huấn luyện random forest như một bộ phát hiện botnet trên tập dữ liệu CTU và đánh giá hiệu suất của các mẫu phản đối được tạo ra. Tuy nhiên, những công trình này đặt ra giả định không thực tế rằng kẻ tấn công có kiến thức hoàn hảo về bộ phân loại, và điều này rõ ràng không nhất quán với các kịch bản tấn công thực tế. End-to-end attack: Khác với phương pháp tấn công trong không gian đặc trưng, phương pháp tấn công end-to-end tập trung vào việc tạo ra các mẫu lưu lượng truy cập mạng thực tế, giống như các mẫu lưu lượng truy cập mạng của botnet thực sự. Phương

pháp tấn công end-to-end đòi hỏi tương đối khắt khe để tạo ra các mẫu lưu lượng giả mạo, bao gồm việc giữ nguyên chức năng độc hại trong các mẫu lưu lượng giả mạo và không thể phá hủy cấu trúc tệp và cấu trúc giao thức mạng. Vì vậy, phương pháp này đòi hỏi các kỹ sư an ninh phải hiểu rõ cấu trúc và hoạt động của botnet để có thể tạo ra các mẫu lưu lượng giả mạo hiệu quả.

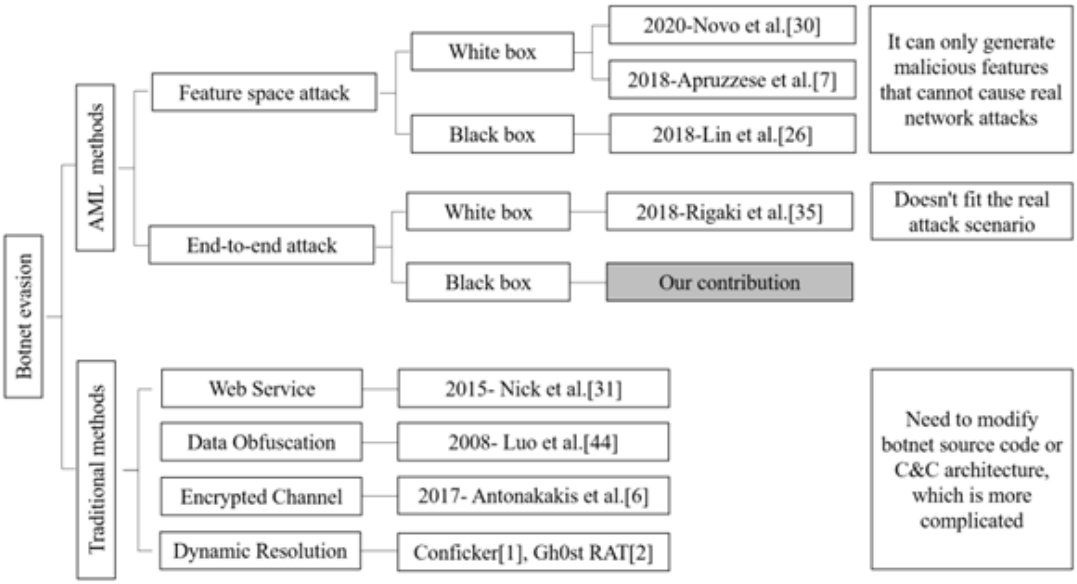


Figure 1: Taxonomy of the Botnet Evasion

Trong đó [35] đề xuất một phương pháp sử dụng generative adversarial network (GAN) để mô phỏng hành vi của lưu lượng mạng Facebook chat để né tránh một Stratospheric IPS tự động thích nghi. Bài báo của họ cố gắng điều chỉnh hành vi của kênh giao tiếp để mô phỏng hành vi của lưu lượng mạng Facebook chat dựa trên các đặc điểm nhận được từ GAN, chẳng hạn như tổng kích thước byte, thời lượng và khoảng thời gian giữa luồng hiện tại và luồng tiếp theo. Tuy nhiên, bài báo không giải thích rõ ràng cách thay đổi mã nguồn để đạt được các thay đổi được chỉ ra bởi GAN, và IPS được đề cập trong bài báo không phải là một bộ phát hiện luồng mà là một bộ phát hiện 3-tuple (IP nạn nhân, IP máy chủ, cổng máy chủ). Cũng có một số nghiên cứu khác đã sử dụng khả năng tạo ra của GAN để tạo ra lưu lượng mạng giống như thật nhất có thể. Mục đích của các nghiên cứu này là cải thiện chất lượng tập dữ liệu để huấn luyện các bộ phát hiện lưu lượng độc hại để giải quyết vấn đề mất cân bằng dữ liệu, nhưng không thể đảm bảo rằng lưu lượng mạng được tạo ra thực sự xảy ra trên liên kết dữ liệu. Hơn nữa, các phương pháp này không xem xét xem lưu lượng được tạo ra có thể thực hiện chức năng độc hại của botnet hay không. Điều này dẫn đến một kịch bản hoàn toàn khác với công trình của nhóm tác giả, trong đó tác giả nhằm mục đích vượt qua trình phát hiện lưu lượng botnet bằng cách tạo ra các mẫu đối kháng botnet.

3 Threat Model and System Framework

3.1 Threat model

- **Adversary's goal**: Mục tiêu chính của attacker là đánh lừa máy dò bằng cách tạo ra các mẫu đối nghịch, đó là dữ liệu kiểm tra được sửa đổi có thể trốn tránh sự phát hiện bằng cách thêm những phần nhiễu linh hoạt vào mẫu nhằm tăng khả năng tàng hình của dòng botnet bằng cách ngụy trang nó, cố gắng giảm sự khả dụng của hệ thống phát hiện xâm nhập mạng che giấu luồng botnet.
- **Adversary's knowledge**: Attacker hiểu rằng mạng đích có thể được bảo vệ bởi một hệ thống phát hiện xâm nhập mạng cấp luồng dựa trên máy học. Tuy nhiên, attacker không cần phải có kiến thức trước về thuật toán, tham số, tính năng, hoặc dữ liệu huấn luyện của máy dò.
- **Adversary's capability**: Trong kịch bản trốn tránh dựa trên các cuộc tấn công đối nghịch, attacker có khả năng sửa đổi dữ liệu kiểm tra nhưng không phải dữ liệu huấn luyện của máy dò. Điều này có nghĩa là attacker có thể sửa đổi luồng botnet mà không ảnh hưởng đến tiện ích hoặc mã nguồn của nó. Ngoài ra attacker có toàn quyền kiểm soát botmaster và kiểm soát một phần các bot, có nghĩa là kẻ tấn công có thể cập nhật bot để thay đổi hành vi giao tiếp của họ bằng cách thiết lập một proxy. Điều này cho phép kiểm soát nhiều hơn dòng botnet và làm cho nó dễ dàng hơn để trốn tránh sự phát hiện.

3.2 System design

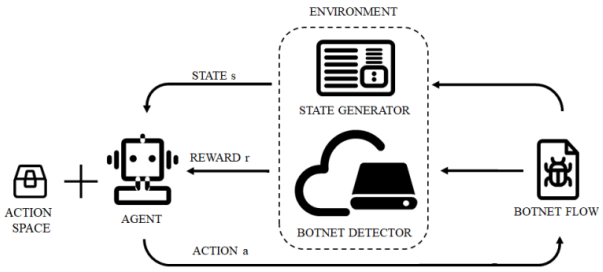


Figure 2: System framework

Thiết lập hệ thống để bỏ qua máy dò botnet dựa trên ML bao gồm hai thành phần: **Agent** và **Environment**.(Figure 2) **Agent** nhận được phản hồi từ môi trường và chọn hành động từ không gian hành động để sửa đổi luồng botnet theo thuật toán RL. **Environment** bao gồm hai module: một botnet detector và state generator. Botnet detector là máy dò cần được bỏ qua, và nhiệm vụ của nó là liên tục dự đoán lưu lượng truy cập độc hại và đưa kết quả nhị phân trở lại cho agent như một phần thưởng. Phần thưởng là một số thực 0;R, đưa ra phản hồi tích cực mạnh mẽ khi tác nhân sửa đổi thành công mẫu lưu lượng botnet để bỏ qua máy dò. Trình tạo trạng thái tạo ra một mô tả về trạng thái mẫu hiện tại và giúp tác nhân đưa ra quyết định hành động. Khi hệ thống bắt đầu chạy, bộ tạo trạng thái tạo ra một trạng thái s dựa trên mẫu đầu vào. Tác nhân nhận thức trạng thái và chọn một action a từ không gian action để sửa đổi mẫu dòng botnet theo thuật toán RL. Mẫu sửa đổi sau đó được nhập vào máy dò botnet, và thu được kết quả dự đoán. Các agent nhận được reward r theo phản hồi nhị phân, và bộ tạo trạng thái tạo ra trạng thái tiếp theo theo mẫu sửa đổi. Vòng lặp tiếp tục cho đến khi máy dò bị nhầm thành công hoặc số hoạt động đạt đến giới hạn trên.Nếu detector vẫn không được bypass thành công, một nỗ lực sửa đổi mẫu khác được bắt đầu.

3.3 Reinforcement Learning Algorithm

Học tăng cường là một loại kỹ thuật học máy cho phép một tác nhân học trong môi trường tương tác bằng cách thử và sai bằng cách sử dụng phản hồi từ các hành động và trải nghiệm của chính nó.

Nó được sử dụng bởi các phần mềm và máy móc khác nhau để tìm ra hành vi hoặc con đường tốt nhất có thể mà chúng nên thực hiện trong một tình huống cụ thể. Để giải quyết vấn đề RL, agent chọn một thuật toán RL để học cách thực hiện hành động tốt nhất trong mỗi trạng thái có thể xảy ra mà nó gặp phải.

Nó cảm nhận được một trạng thái nhất định từ môi trường và tối đa hóa giá trị phần thưởng reward dài hạn của nó bằng cách học cách chọn một hành động thích hợp dựa trên tín hiệu nâng cao do môi trường cung cấp.

Có 2 thuật toán được đề xuất trong bài là DQN và SARSA.

DQN.Q-learning là một phương pháp off-policy temporal-difference (TD) được sử dụng để ước lượng Q-value để chọn hành vi tốt nhất. Tuy nhiên, khi không gian trạng thái (state space) hoặc không gian hành động (action space) lớn, Q-learning gặp vấn đề "dimensionality disaster"(thảm họa chiều không gian).

$$L(\theta) = E[(TargetQ - Q(s, a; \theta))^2] \quad (1)$$

where

$$TargetQ = r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a; \theta)$$

DQN sử dụng một mạng nơ-ron để ước lượng hàm Q-value thay vì sử dụng Q-table trong Q-learning. Mạng nơ-ron này được định nghĩa là một hàm Q(s, a; theta), trong đó theta là tham số mạng. Hàm mất mát (loss function) L(theta) của mạng được định nghĩa là sai số bình phương giữa giá trị Q-value mục tiêu và giá trị Q-value được đầu ra từ mạng.

DQN giúp giải quyết vấn đề "dimensionality disaster"của Q-learning bằng cách sử dụng một mạng nơ-ron để ước lượng Q-value thay vì sử dụng bảng Q-value (Q-table) trong Q-learning.

SARSA. một thuật toán on-policy cho TD learning. Giá trị Q tối ưu, được ký hiệu là Q(st, at), có thể được biểu diễn như sau:

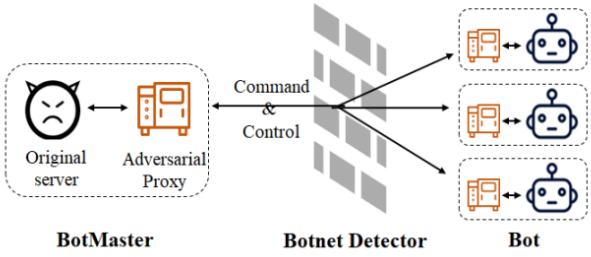
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2)$$

Sự khác biệt chính giữa Q-learning và SARSA là SARSA là thuật toán on-policy học giá trị Q dựa trên hành động được thực hiện bởi chính chính sách hiện tại, trong khi Q-learning là thuật toán off-policy học giá trị Q dựa trên chính sách tham lam. SARSA cẩn thận hơn trong việc chọn hành động và thường tránh các hành động có nguy cơ cao hơn, trong khi Q-learning có thể dẫn đến các hành động nguy hiểm hơn trong quá trình khám phá.

Thứ hai, dưới một số điều kiện phổ biến, cả hai thuật toán đều hội tụ đến hàm giá trị thực tế, nhưng tốc độ hội tụ khác nhau. Q-learning có xu hướng hội tụ chậm hơn so với SARSA do chính sách tham lam, nhưng nó có khả năng tiếp tục học tập trong khi thay đổi chính sách. Q-learning trực tiếp hội tụ đến chính sách tối ưu, trong khi SARSA chỉ học được chính sách gần tối ưu bằng cách khám phá.

4 Experimental Setup

4.1 Implement



Bài báo mô tả một phương pháp để bypass phát hiện botnet bằng cách dùng adversarial proxy có thể dễ dàng triển khai ở phía botmaster. Bằng cách cập nhật bot thông qua CnC, attacker có thể đạt được mà ko cần sửa đổi phức tạp đối với phầm mềm độc hại ban đầu. Adversarial proxy với agent RL được đào tạo có thể thực hiện các hành động gia tăng luồng botnet cho đến khi nó vượt qua bộ phát hiện thành công cho phép attacker đạt được adversarial attack-based botnet detector trong ngữ cảnh blackbox.

4.2 Dataset

Để đánh giá hiệu quả của các phương pháp phát hiện botnet, điều quan trọng là phải kiểm tra chúng trên các bộ dữ liệu đa dạng mô phỏng lưu lượng mạng trong thế giới thực. Sử dụng hai bộ dữ liệu công khai, CTU và ISOT, chứa cả lưu lượng độc hại và không độc hại, bao gồm dấu vết của các mạng botnet nổi tiếng như Storm và Zeus. Các bộ dữ liệu này bao gồm một loạt các hoạt động mạng, chẳng hạn như duyệt web, email và phương tiện truyền phát trực tuyến, làm cho chúng đại diện cho lưu lượng truy cập trong thế giới thực.

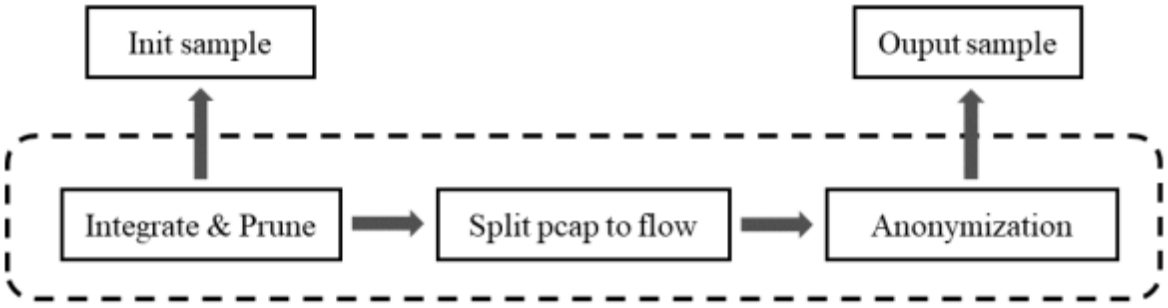
Used for	Channel	Botnet Family	File Serial No.	Original sessions	Used sessions	Captured year	Notes
Train & Test	IRC	Menti	CTU-47	4,507	4,000	2011	
		Rbot	CTU-44/45/52	31,736	10,000	2011	An IRC-based botnet with 40,000 active members
		Murio	CTU-49	8,381	8,000	2011	
	HTTP	Virut	CTU-46/54	31,173	30,000	2011	Responsible for 5.5 percent of malware infections during the third quarter of 2012
		Miuref(3ve)	CTU-127	13,426	10,000	2015	It infected around 1.7 million computers
		Neris	CTU-42/43	31,133	30,000	2011	
		HTbot	CTU-348	36,855	30,000	2018	One of the largest reported Android banking botnets known to date
		Dridex/Necurs	CTU-346	10,029	10,000	2018	One of the world's largest spam botnets
		Trickbot	CTU-327	30,052	30,000	2018	The top business threat in 2018
	P2P	Storm	ISOT-Storm	4,672	4,672	2012	Infected more than 1 million systems
Train	Normal	–	ISOT-normal	511,322	80,000	2010	

Bộ dữ liệu được sử dụng trong nghiên cứu bao gồm nhiều kênh giao tiếp botnet khác nhau, bao gồm IRC, HTTP và P2P, đồng thời bao gồm lưu lượng truy cập từ các dòng botnet chính đã gây ra các cuộc tấn công quan trọng hoặc có các phương pháp ẩn náu nâng cao. Bộ dữ liệu cũng kéo dài một khoảng thời gian lớn từ 2011-2018, khiến nó trở nên linh hoạt và mới lạ hơn các bộ dữ liệu hiện có khác. Thực hiền tiền xử lý dữ liệu qua 3 bước:

• step 1: Kết hợp lưu lượng được thu thập thuộc cùng một họ botnet. Nếu tệp pcap quá lớn, nó sẽ bị cắt. Mục đích của bước này là để cân bằng cỡ mẫu của mỗi gia đình.

• step 2: Chia pcap thành các phiên. Điều này được thực hiện để có được thông tin liên lạc đầy đủ hơn. Một phiên đề cập đến tất cả các gói bao gồm các luồng hai chiều, nghĩa là IP nguồn và IP đích có thể hoán đổi cho nhau trong một bộ 5 (SIP, SPort, DIP, DPort, Giao thức). sử dụng SplitCap [3] để chia từng tệp pcap thành các phiên.

• Step 3: Tệp lưu lượng truy cập của chúng tôi được tạo từ các môi trường mạng khác nhau. Để loại bỏ hiệu ứng của địa chỉ IP và địa chỉ MAC trên máy dò, thông tin duy nhất của dữ liệu lưu lượng truy cập cần phải được chọn ngẫu nhiên. Cụ thể, chúng tôi thay thế chúng bằng một địa chỉ mới được tạo ngẫu nhiên.



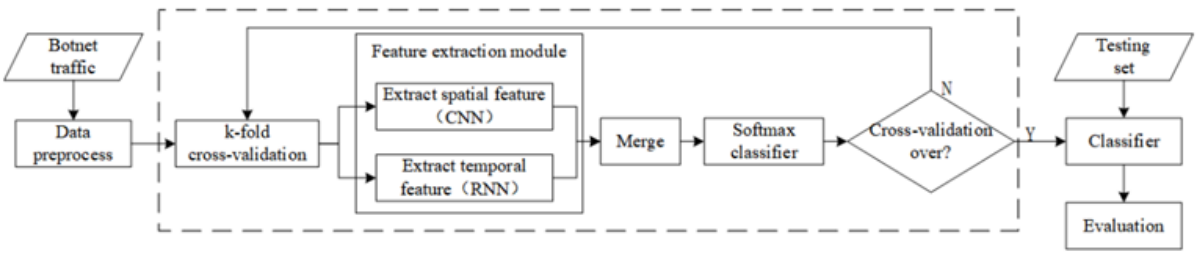
4.3 Detector

Trình phát hiện trong hệ thống chịu trách nhiệm dự đoán luồng botnet đã sửa đổi và cung cấp phản hồi cho agent ở dạng nhị phân. Bài báo so sánh hai mô hình phát hiện tiên tiến nhất trong các thử nghiệm của họ: mô hình học sâu tổng hợp kết hợp CNN với LSTM và mô hình học máy không phân biệt dựa trên XGBoost, cả hai đều được sử dụng làm công cụ phát hiện mạng botnet.

Mô hình phát hiện BotCatcher là một thuật toán học sâu có thể phát hiện lưu lượng truy cập bất thường do botnet tạo ra thông qua phân tích lưu lượng truy cập. Nó tự động trích xuất các tính năng không gian và thời gian từ lưu lượng mạng và đào tạo một bộ phân loại softmax tương ứng. Mô hình được thiết kế để phân biệt giữa các chế độ giao tiếp của máy chủ CnC và bot so với người dùng thông thường.

Mô hình phát hiện BotCatcher sử dụng Convolutional Neural Networks (CNN) để trích xuất các tính năng không gian và Recurrent Neural Networks (RNN), cụ thể là Long Short-Term Memory (LSTM), để tìm hiểu các tính năng tạm thời từ dữ liệu phiên botnet. Các tính năng này sau đó được xử lý thông qua mạng thần kinh nhiều lớp và được đưa vào lớp softmax để xác định các mẫu lưu lượng truy cập bất thường.

Mô hình phát hiện XGBoost là một thuật toán máy học sử dụng tăng cường độ dốc để cải thiện độ chính xác của dự đoán. Nó được tạo bởi Tianqi Chen và được biết đến với tốc độ và hiệu quả trong việc xử lý dữ liệu có cấu trúc hoặc dạng bảng. XGBoost đã được sử dụng rộng rãi trong các cuộc thi và học máy ứng dụng như Kaggle do hiệu suất cao của nó.



5 Result

5.1 Paper

Bài báo đã chia tập dữ liệu của họ thành bốn tập con để thử nghiệm mô hình tấn công của họ: detector training set, agent training set, detector testing set và agent testing set. Để mô phỏng một kịch bản tấn công thực sự trong đó attacker có thể không có quyền truy cập vào dữ liệu đào tạo của máy dò mục tiêu. Sau đó, họ so sánh hiệu suất của các thuật toán học tăng cường và trình phát hiện khác nhau bằng cách triển khai bốn phiên bản hệ thống, với mỗi tác nhân được đào tạo cho một số vòng cụ thể.

Table 4: Evasion performances of system instances.

	Menti	Rbot	Murio	virut	Miuref	Neris	HTBot	Dridex	Trickbot	Storm
XGBoost-SARSA	87%	83%	76%	86%	66%	66%	61%	41%	31%	0%
XGBoost-DQN	85%	77%	75%	85%	60%	56%	49%	41%	30%	1%
XGBoost-Random	75%	72%	68%	71%	54%	42%	45%	34%	24%	0%
BotCatcher-SARSA	21%	26%	24%	40%	42%	34%	54%	38%	59%	73%
BotCatcher-DQN	21%	22%	22%	41%	38%	28%	52%	50%	51%	64%
BotCatcher-Random	17%	19%	20%	37%	37%	27%	48%	37%	42%	59%

Thuật toán :
So sánh giữa hai thuật toán học tăng cường, SARSA và DQN, về hiệu suất của chúng trong việc tạo ra các mẫu đối thủ hiệu quả để trốn tránh sự phát hiện của các máy dò botnet khác nhau. Kết quả cho thấy rằng các agent SARSA thường hoạt động tốt hơn các agent DQN, có thể là do cách tiếp cận chính sách thận trọng của SARSA. Ngoài ra, bài báo gợi ý rằng các agent SARSA có thể phù hợp hơn để tạo các ví dụ về đối thủ với ít bước hơn và nhiễu loạn nhỏ.
Dectector :
Evasion performance : XGBoost và BotCatcher, về khả năng phát hiện botnet của chúng. Tác giả nhận thấy rằng XGBoost có tỷ lệ trốn tránh cao hơn BotCatcher. Họ kết luận rằng sự sửa đổi có mục tiêu và tính nhất quán cao giữa không gian hành động và không gian đặc trưng của máy dò chịu trách nhiệm chính cho lỗ hổng của máy dò.
Time performances : hiệu suất của hai công cụ phát hiện khác nhau, XGBoost và BotCatcher, trong việc phát hiện các cuộc tấn công đối nghịch. Kết quả cho thấy XGBoost dễ bị tấn công hơn BotCatcher, vì nó có tỷ lệ né tránh cao hơn và cần ít bước hơn để đánh lừa. Điều này cho thấy rằng BotCatcher có thể là một công cụ phát hiện hiệu quả hơn để phát hiện các cuộc tấn công đối nghịch.

5.2 Kết quả của nhóm

Xây dựng các Agent SARSA và DQN theo các bước như sau :
- **Bước 1** :Xây dựng và xử lý dataset. Sau đó tạo 1 môi trường với openAI gym và import các thư viện liên quan đến việc xử lý thuật toán.

```
[ ]  # Load data
     df = pd.read_csv('/content/drive/MyDrive/Nam3(2022-2023)/HK_2/Phương pháp học máy/ĐỒ ÁN/CTU13_Attack_Traffic.csv')
     df_normal = pd.read_csv('/content/drive/MyDrive/Nam3(2022-2023)/HK_2/Phương pháp học máy/ĐỒ ÁN/CTU13_Normal_Traffic.csv')
     df = pd.concat([df, df_normal]) # combine attack and normal traffic data
     X = df.drop(columns=['Label']).values
     y = df['Label'].values
     le = LabelEncoder()
     y = le.fit_transform(y)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ]  # Define environment
     class CustomEnv(gym.Env):
         def __init__(self, X, y):
             super(CustomEnv, self).__init__()
             self.X = X
             self.y = y
             self.observation_space = gym.spaces.Box(low=0, high=1, shape=(X.shape[1],))
             self.action_space = gym.spaces.Discrete(len(np.unique(y)))
             self.current_step = 0

         def reset(self):
             self.current_step = 0
             obs = self.X[self.current_step]
             return obs

         def step(self, action):
             obs = self.X[self.current_step]
             done = self.current_step == len(self.X) - 1
             reward = int(action == self.y[self.current_step])
             self.current_step += 1
             return obs, reward, done, {}
```

- **Bước 2** : Xây dựng các bộ detector liên quan BotCatcher/XGBoost

```
class BotCatcherDetector:
    def __init__(self):
        self.model = Sequential()
        self.model.add(Dense(16, input_shape=X_train.shape[1:], activation='relu'))
        self.model.add(Dense(1, activation='sigmoid'))
        self.model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    def fit(self, X, y, **kwargs):
        self.model.fit(X, y, **kwargs)

    def predict(self, X):
        return self.model.predict(X)

    def evaluate(self, X, y):
        return self.model.evaluate(X, y)
```

- **Bước 3** : Sau khi định nghĩa các detector , ta sẽ xây dựng các Agent train các thuộc tính dữ liệu trong 1 action space với hàm kích hoạt và lớp Environment đã khởi tạo.Sau đó sẽ train trên step (ở đây steps= 10000) để đánh giá giá trị reward. Theo bài báo giá trị reward r nằm trong 0;R. Nó sẽ dự đoán lưu lượng độc hại trong vòng lặp và trả lại kết quả nhị phân cho agent là giá trị reward r. Nếu bộ phát hiện bị đánh lừa thành công, giá trị reward sẽ được trả về 1 giá trị nào đó; nếu không, nó sẽ là 0.

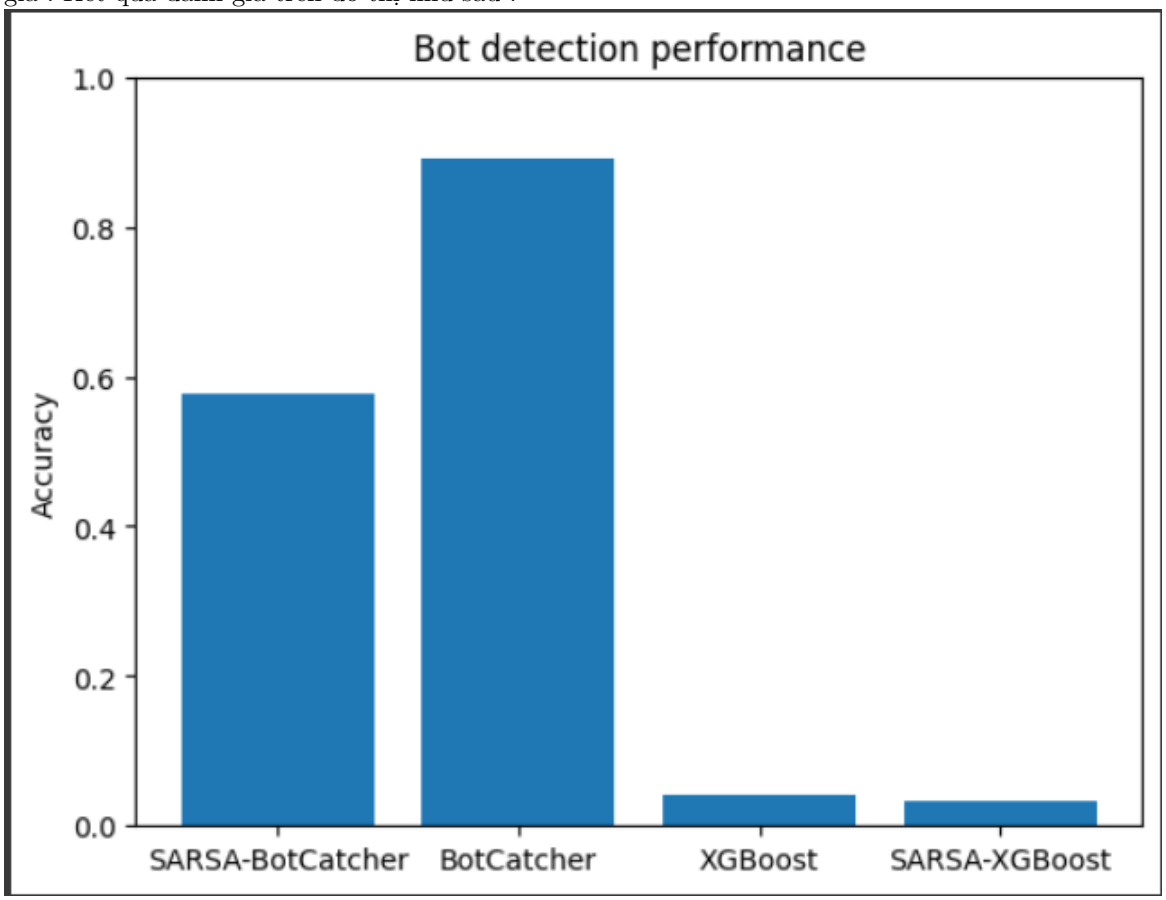
```
# Define SARSA agent with BotCatcher detector
sarsa_env = CustomEnv(X_train, y_train)
sarsa_model = Sequential()
sarsa_model.add(Flatten(input_shape=(1,) + sarsa_env.observation_space.shape))
sarsa_model.add(Dense(16, activation='relu'))
sarsa_model.add(Dense(sarsa_env.action_space.n, activation='linear'))
sarsa_agent = SARSAAgent(model=sarsa_model, nb_actions=sarsa_env.action_space.n, policy=EpsGreedyQPolicy(),
                         nb_steps_warmup=10)
sarsa_agent.compile(Adam(lr=1e-3), metrics=['mae'])
sarsa_agent.fit(sarsa_env, nb_steps=10000, visualize=False, verbose=1, nb_max_episode_steps=len(X_train))
```

```
Training for 10000 steps ...
Interval 1 (0 steps performed)
    9/10000 [..............................] - ETA: 1:04 - reward: 0.6667/usr/local/lib/python3.10/dist-packages/
  super().__init__(name, **kwargs)
/usr/local/lib/python3.10/dist-packages/rl/policy.py:146: DeprecationWarning: This function is deprecated. Please
  action = np.random.random_integers(0, nb_actions-1)
10000/10000 [==============================] - 264s 26ms/step - reward: 0.5488
done, took 263.888 seconds
<keras.callbacks.History at 0x7efead3f2b90>
```

-**Bước 4** : Cuối cùng ta sẽ đánh giá detector phát hiện lưu lượng mẫu tạo ra và đánh giá mức độ Bypass của các mẫu khi áp dụng Agent để giảm độ chính xác mà detector đã đánh giá .

```
2306/2306 [==============================] - 6s 2ms/step - loss: 20926.8027 - accuracy: 0.8404
577/577 [==============================] - 1s 1ms/step
SARSA with BotCatcher detector test accuracy: 0.577509082036545
BotCatcher detector test accuracy: 0.8903106869815106
```

-**Đánh giá trên đồ thị** : Với XGBoost detector , ta cũng xây dựng tương tự trên bộ dataset khác và đánh giá . Kết quả đánh giá trên đồ thị như sau :



5.3 Đánh giá

- Kết quả nhóm demo được dựa trên việc tự tìm hiểu xây dựng thuộc tính và tìm kiếm các dataset thay thế và thực hiện đánh giá chung số lượng mẫu bypass.
- Kết quả bài báo dựa trên dataset việc kết hợp nhiều thuộc tính đặc trưng , tích hợp, lược bỏ, kết hợp các lưu lượng để tạo thành 1 bộ dataset mới nên các kết quả đánh giá đánh giá chi tiết hơn trên từng họ botnet.

Khó khăn khi thực hiện :

- Không tìm được mã nguồn tham khảo.
- Bộ dataset tác giả xây dựng cũng không công khai nên việc tự xây dựng đánh giá không đúng số liệu mong muốn mà phải dùng bộ dataset thay thế để đánh giá chung toàn thể.

Tài liệu

[1] Qixu L. Di W. Dong Y. Cui X. Wang, J. *Crafting adversarial example to bypass flow-ML-based botnet detector via RL.* Intrusions and Defenses (pp. 193-204), (2021, October).