Tạo ví dụ về đối thủ để vượt qua dựa trên Flow-&ML Máy dò Botnet qua RL

Viện Kỹ thuật
Thông tin Junnan Wang , Học viện
Khoa học Trung Quốc Bắc Kinh,
Trường An ninh
Mạng Trung Quốc, Đại học Học viện
Khoa học Trung Quốc Bắc Kinh,
Trung Quốc
wangruonan@iie.ac.cn

Qixu Liu
Viện Kỹ thuật Thông tin, Học viện
Khoa học Trung Quốc Bắc Kinh,
Trường An ninh
Mạng Trung Quốc, Đại học Học viện
Khoa học Trung Quốc Bắc Kinh,
Trung Quốc
liuqixu@iie.ac.cn

Di Wu
Huawei Technologies Co., Ltd.
Shen Zhen, Trung
Quốc wudi94@huawei.com

Ying Dong
Beijing Venus Information Security
Technology Incorporated Company Bắc
Kinh, Trung
Quốc dong_ying@venustech.com.cn

Viện Công
nghệ Tiên tiến Không gian mạng
Xiang Cui , Đại học Quảng Châu
Quảng Châu, Trung
Quốc cuixiang@gzhu.edu.cn

TRỪU TƯỢNG

Các phương pháp phát hiện botnet dựa trên máy học (ML) đã trở thành xu hướng chủ đạo trong thực tiễn của công ty. Tuy nhiên, các nhà nghiên cứu đã phát hiện ra rằng các mô hình ML dễ bị tấn công bởi các đối thủ, điều này có thể đánh lừa các mô hình bằng cách thêm các nhiễu loạn tinh vi vào mẫu. Do sự phức tạp của các mẫu lưu lượng truy cập và các ràng buộc đặc biệt để giữ các chức năng độc hại, không có nghiên cứu đáng kể nào về ML đối thủ được thực hiện trong lĩnh vực phát hiện botnet, nơi các cuộc tấn công trốn tránh gây ra bởi các thử nghiệm đối thủ được tạo cẩn thận có thể trực tiếp tạo ra ML. máy dò không có sẵn và gây ra thiệt hại tài sản đáng kể. Trong bài báo này, chúng tôi đề xuất một phương pháp học tăng cường (RL) để vượt qua các trình phát hiện botnet dựa trên ML.
Cụ thể, chúng tôi đào tạo tác nhân RL làm công cụ sửa đổi luồng mạng bot bảo toàn chức năng thông qua một loạt tương tác với trình phát hiện trong tình huống hộp đen. Điều này cho phép kẻ tấn công tránh bị phát hiện mà không sửa đổi mã nguồn botnet hoặc ảnh hưởng đến tiện ích botnet. Các thử nghiệm trên 14 họ botnet chứng minh rằng phương pháp của chúng tôi có hiệu suất trốn tránh và hiệu suất thời gian đáng kể.

KHÁI NIỆM CCS

• Bảo mật và quyền riêng tư Phát hiện xâm nhập/bất thường và giảm thiểu phần mềm độc hại; • Phương pháp tính toán Trí tuệ nhân tạo.

TỪ KHÓA

Bỏ qua trình phát hiện botnet, học máy đối thủ, học tăng cường

Định dạng tham chiếu ACM:
Junnan Wang, Qixu Liu, Di Wu, Ying Dong và Xiang Cui. 2021. Tạo ví dụ về đối thủ để vượt qua Trình phát hiện Botnet dựa trên Flow-&ML thông qua RL. Trong Hội nghị chuyên đề quốc tế lần thứ 24 về nghiên cứu tấn công, xâm nhập và phòng thủ (RAID '21), ngày 6–8 tháng 10 năm 2021, San Sebastian, Tây Ban Nha. ACM, New York, NY, USA, 12 trang. https://doi.org/10.1145/3471621.3471841

1. GIỚI THIỆU

Học máy (ML) đã thúc đẩy mạnh mẽ sự phát triển của công nghệ phát hiện mạng bot. Không giống như các phương pháp phát hiện dựa trên chữ ký , tính năng phát hiện bất thường dựa trên máy học có thể xác định chính xác và hiệu quả lưu lượng truy cập do phần mềm độc hại tạo ra khi một số mẫu hành vi nhất định được nhận dạng. Đặc biệt, sự giống nhau đáng kể về không gian và thời gian của các botnet khiến chúng dễ dàng bị phát hiện bởi các mô hình dựa trên ML. Ngoài ra, các phương pháp dựa trên ML có thể xác định các họ botnet không xác định và phù hợp hơn để 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 cho phép chúng vượt qua sự tiến bộ trong các hệ thống phát hiện và vượt qua phát hiện dựa trên hành vi. Ví dụ: kẻ tấn công có thể thường xuyên thay đổi địa chỉ IP tương ứng với tên miền của máy chủ chỉ huy và kiểm soát (C&C) để tránh bị phát hiện trong danh sách đen IP; sử dụng các giao thức lớp ứng dụng (HTTP, DNS) hoặc các dịch vụ web bên ngoài (Twitter, Facebook) để liên lạc C&C nhằm vượt qua tường lửa; và mã hóa thông tin liên lạc C&C để tránh trình phát hiện botnet dựa trên tải trọng. Tuy nhiên, các phương pháp trốn tránh này không thể vượt qua các công cụ phát hiện botnet dựa trên Flow-& ML được sử dụng rộng rãi , giúp phân biệt độc hại dựa trên các đặc điểm thống kê của các luồng. Hơn nữa, các phương pháp này yêu cầu sửa đổi nguồn trực tiếp rộng rãi hoặc phức tạp, do đó đặt ra yêu cầu cao đối với những kẻ tấn công [38].

Một cách trực tiếp để vượt qua trình phát hiện botnet dựa trên ML là sử dụng lỗ hổng của ML để tấn công mô hình phát hiện ML. Szegedy và cộng sự. [40] lần đầu tiên phát hiện ra rằng các mô hình ML hoạt động tốt dễ bị

Tác giả tương ứng

RAID '21, ngày 6-8 tháng 10 năm 2021, San Sebastian,
Tây Ban Nha © 2021 Hiệp hội Máy tính.
ACM ISBN 978-1-4503-9058-3/21/10. . . $15,00
https://doi.org/10.1145/3471621.3471841

các cuộc tấn công đối nghịch dưới hình thức thêm một số nhiễu loạn nhỏ vào đầu vào để đánh lừa một mô hình tạo ra kết quả đầu ra không chính xác. Nhiều công trình gần đây cũng đã đề xuất các phương pháp khéo léo (L-BFGS [40], FGSM [18], Deepfool [28] và cộng sự) để tạo các mẫu đối thủ, nhưng các phương pháp này khó áp dụng trực tiếp để tạo các mẫu đối thủ của mạng botnet.

Trong bối cảnh phát hiện botnet, các ràng buộc đối với nhiễu loạn được thêm vào ví dụ đối nghịch không còn là không thể chấp nhận được đối với con người, nhưng nó không thể ảnh hưởng đến ý định độc hại ban đầu của lưu lượng botnet. Độ phức tạp và định dạng cụ thể của dữ liệu lưu lượng botnet xác định nhu cầu phát triển các phương pháp mới để xây dựng các mẫu đối thủ của botnet nhằm vượt qua các công cụ phát hiện botnet dựa trên ML .

Trong bài báo này, một phương pháp dựa trên học tăng cường (RL) để vượt qua bộ phát hiện botnet mức lưu lượng dựa trên ML được trình bày. Chúng tôi cố gắng đánh lừa trình phát hiện dựa trên ML bằng cách thêm một số nhiễu loạn vào luồng botnet. Cụ thể, chúng tôi mô hình hóa việc sửa đổi luồng botnet dưới dạng một vấn đề quyết định tuần tự, để tác nhân RL tìm hiểu chiến lược sửa đổi tối ưu trong một loạt các tương tác với trình phát hiện botnet. Để đảm bảo duy trì chức năng, chúng tôi thiết kế một không gian hành động chứa 14 thao tác gia tăng, mỗi thao tác chỉ thêm một gói được chế tạo cẩn thận vào luồng ban đầu nhằm cố gắng thay đổi một số đặc điểm cấp độ luồng.
Máy dò coi những đặc điểm này là phân biệt đối xử, nhưng đây có thể không phải là dấu hiệu nhân quả của lưu lượng truy cập lành tính. Hơn nữa, việc thêm các gói là một hoạt động gia tăng ở lớp vận chuyển, trong khi các chức năng độc hại thường được gói gọn trong lớp ứng dụng. Do đó, có thể đảm bảo rằng ý định độc hại ban đầu sẽ không bị phá hủy.

Những ưu điểm của phương pháp tấn công của chúng tôi bao gồm (1) đó là một cuộc tấn công hộp đen, phù hợp với các tình huống tấn công thực tế hơn các phương pháp khác; (2) nó là chung và có thể được sử dụng bất kể hàm mất mát của máy dò có khả vi hay không; (3) nó là plug and play, tác nhân RL có thể tồn tại dưới dạng proxy, cuộc tấn công có chi phí trốn tránh thấp và phù hợp với bất kỳ họ botnet nào.

Thông qua các thử nghiệm mở rộng, chúng tôi chứng minh rằng trình phát hiện botnet dựa trên ML hiện tại dễ bị tấn công. Những kẻ tấn công có thể tránh bị phát hiện bằng cách chỉ thêm một vài gói vào luồng botnet với chi phí tương đối nhỏ và không cần biết trước.

Những đóng góp của bài viết này có thể được tóm tắt như sau:

- Chúng tôi đề xuất một khuôn khổ tấn công hộp đen chung cho các máy phát hiện botnet dựa trên ML. Chúng tôi cho rằng kẻ tấn công có thể truy cập vào trình phát hiện và nhận được kết quả phân biệt nhị phân đầu vào (phần mềm độc hại/lành tính) nhưng không có bất kỳ kiến thức nào trước đó về thuật toán và không gian đặc trưng được trình phát hiện sử dụng. Theo hiểu biết tốt nhất của chúng tôi, công trình này là nghiên cứu đầu tiên về các cuộc tấn công đối nghịch hộp đen trong lĩnh vực trốn tránh botnet.
- Chúng tôi thiết kế một loạt các không gian hành động phổ quát và đóng gói chúng trong khuôn khổ RL. Một mặt, tất cả các hành động đều là các hoạt động gia tăng, do đó đảm bảo rằng việc truyền các chức năng và thông tin độc hại không bị ảnh hưởng.
  Mặt khác, các hành động là phổ biến và được đóng gói tốt, cho phép kẻ tấn công thoát khỏi sự phát hiện mà không cần sửa đổi phức tạp đối với phần mềm độc hại botnet.
- Chúng tôi trình bày cách đào tạo và triển khai hệ thống của mình để tránh bị phát hiện ML trên bộ dữ liệu luồng mạng botnet được xây dựng cẩn thận và đánh giá toàn diện hiệu suất trốn tránh, chi phí thời gian và tính phổ biến của khung.

Bài báo này được chia thành 6 phần như sau: Phần 2 giới thiệu các công trình liên quan. Phần 3 mô tả khung hệ thống.
Phần 4 cho thấy cách chúng tôi thiết lập thử nghiệm của mình. Phần 5 thảo luận về kết quả thí nghiệm và kết luận. Ở phần cuối của bài báo, chúng tôi tóm tắt và triển vọng công việc của bài báo này.

2 CÔNG VIỆC LIÊN QUAN

2.1 Học máy đối nghịch

Trong văn học, có nhiều công trình tập trung vào các cuộc tấn công đối nghịch. Các nhà nghiên cứu đã đề xuất nhiều phương thức tấn công tiên tiến khác nhau dựa trên lượng thông tin máy dò có sẵn. Như là
kiến thức về máy dò có thể bao gồm [9]:

- tập huấn luyện hoặc một phần của nó;
- không gian đặc trưng của thuật toán ML;
- loại thuật toán ML (SVM, LR, DecisionTree, CNN, v.v.)
- trình phân loại được đào tạo (chi tiết về mô hình, chẳng hạn như kho lưu trữ của nó kiến trúc và siêu tham số);
- phản hồi từ bộ phát hiện (phản hồi dựa trên điểm bao gồm phản hồi dựa trên xác suất hoặc nhị phân cho nhãn cuối cùng)

Trong bài báo này, chúng tôi nhóm các phương pháp tấn công đối thủ gần đây thành ba kịch bản tấn công dựa trên các mức độ hiểu biết khác nhau của đối thủ về hệ thống bị tấn công: kiến thức hoàn hảo (PK), kiến thức hạn chế (LK) và kiến thức không (ZK).

Kiến thức hoàn hảo. Tấn công hộp trắng. Trong trường hợp này, kẻ tấn công có đầy đủ thông tin về máy dò. Mục tiêu của kẻ tấn công là giảm thiểu chức năng phân loại sai mẫu. [40] đã tận dụng L-BFGS để giải quyết vấn đề tối ưu hóa trong việc tìm ra số lượng nhiễu loạn cần thiết tối thiểu. C&W at tack [13] đã thiết kế một hàm mất mát mới với giá trị nhỏ trong mẫu đối kháng nhưng giá trị lớn hơn trong mẫu sạch để có thể tìm kiếm ví dụ sarial của đối thủ bằng cách giảm thiểu hàm mất mát.

FSGM [18] giả định rằng hàm mất mát trong vùng lân cận của mẫu sạch là tuyến tính.

Deepfool [28] là phương pháp đầu tiên sử dụng chuẩn L2 để giới hạn kích thước nhiễu nhằm đạt được nhiễu tối thiểu. Nhiễu loạn phổ quát [27] mở rộng DeepFool để tạo nhiễu loạn phổ quát và bất khả tri về hình ảnh.

Kiến thức hạn hẹp. Tấn công hộp xám Kẻ tấn công chỉ có kiến thức hạn chế về bộ phát hiện và không thể sử dụng các phương pháp dựa trên độ dốc. Tuy nhiên, khi biết loại hoặc không gian đặc trưng của mô hình, kẻ tấn công có thể đánh lừa bộ phát hiện bằng cách tìm một tập hợp các đặc điểm không đủ phân biệt thông qua các đặc điểm cấu trúc của mô hình, không gian đặc trưng hoặc chuỗi điểm phản hồi.

Apruzzese et al. [6] đã sử dụng một bộ giải chương trình tuyến tính số nguyên hỗn hợp để xây dựng một ví dụ đối nghịch tối ưu để tránh các máy dò botnet dựa trên luồng và rừng ngẫu nhiên.

Papernot et al. [31] đã đào tạo một mô hình thay thế sau khi thực hiện tăng cường tập dữ liệu dựa trên Jacobian để mô phỏng chính xác ranh giới quyết định của máy dò.

Bảng 1: Phân loại các nghiên cứu học thuật mới nhất về các phương pháp tấn công đối thủ.

PK/LK/ZK	Phương pháp	Hình ảnh	Điều bất lợi
PK	L-BFGS [40]	đối tượng	Không thể xử lý các hàm mất không phân biệt
	FSGM [18] [24]	hình ảnh/phần mềm độc hại	
	SLEIPNIR [4]	Windows PE	
	DeepFool [28]	hình ảnh	
	JBSM [32] [19]	hình ảnh hình ảnh/phần	
	Tấn công C&W [13]	mềm độc	
	ATNĐ [7]	hại hình	
	Adv_MalConv [22]	ảnh hình ảnh	
	nhiễu loạn phổ quát [27]	phần mềm	
Phản hồi điểm số LK Mô	hình thay thế tàu [31]	độc hại	Nhắm mục tiêu theo một loại mô hình ML cụ thể
loại mô hình LK	Né tránh-RF [6]	hình ảnh luồng mạng botnet	
ZK	MalGAN [21]	phần mềm độc hại	Trên thực tế thuộc về kịch bản LK, kẻ tấn công cần biết toàn bộ không gian tính năng
ZK	Né tránh HC [14]	phần mềm độc hại pdf	Cần trợ giúp xác minh thông tin
ZK	Tấn công ranh giới [11]	hình	Cần nhiều lần lặp nhất để hội tụ

ảnh PK: Tri thức viên mãn; LK: Kiến thức hạn hẹp; ZK: Không có kiến thức.

Không có kiến thức. . Hộp đen tấn công. Giả sử kẻ tấn công không biết gì về mô hình ngoại trừ quyết định nhị phân của máy dò. Kẻ tấn công chỉ có thể trốn tránh bộ phát hiện thông qua thử và sai hoặc bằng cách tạo ra các ví dụ đối nghịch với bộ phân loại chung Ý tưởng cơ bản là nếu các ví dụ đối nghịch có thể bỏ qua từng mô hình trong bộ sưu tập, thì nó có thể bỏ qua mọi máy dò.

MalGAN [21] đã giới thiệu GAN để tạo phần mềm độc hại PE bằng cách vượt qua trình phát hiện hộp đen. Công cụ phân biệt đối xử GAN là một mô hình máy dò thay thế do kẻ tấn công xây dựng, trong khi trình tạo được sử dụng để tạo các mẫu phần mềm độc hại đối nghịch.

Tấn công ranh giới [11] bắt đầu từ một nhiễu loạn lớn của đối thủ và sau đó tìm cách giảm bớt nó trong khi vẫn còn đối thủ bằng cách thực hiện một bước đi ngẫu nhiên dọc theo ranh giới giữa vùng đối nghịch và không đối nghịch.

Từ Bảng 1, chúng ta có thể thấy rằng hầu hết các cuộc tấn công hộp trắng hiện tại chỉ có thể xử lý các hàm mất mát khả vi, trong khi các cuộc tấn công hộp xám chỉ có thể vượt qua các loại máy dò cụ thể. Một số phương pháp tấn công hộp đen hiện có hoàn toàn không cần bất kỳ kiến thức nào; thay vào đó, các phương pháp này phần nào dựa vào thông tin bên ngoài hoặc thông tin liên quan đến mô hình. Thách thức này thúc đẩy chúng tôi xây dựng một phương pháp tấn công đối thủ hộp đen chung có thể vượt qua mọi trình phát hiện botnet dựa trên ML.

2.2 Trốn tránh Botnet

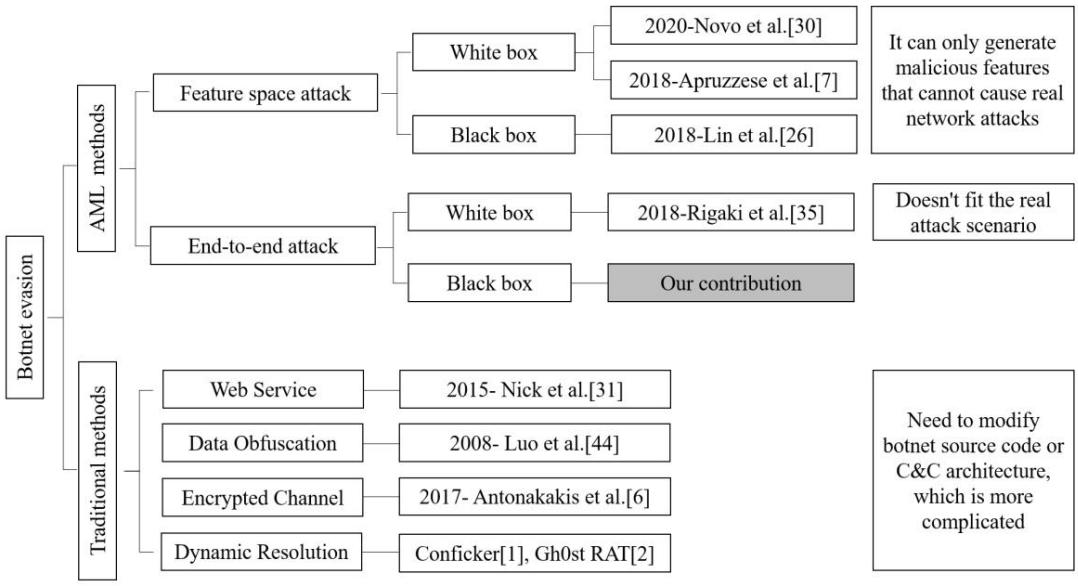
Mặc dù phương pháp phát hiện botnet không ngừng được cải thiện, nhưng những kẻ tấn công cũng đang khám phá các kỹ thuật để tránh bị phát hiện botnet dựa trên ML . Các kỹ thuật tránh botnet truyền thống làm cho lưu lượng truy cập C&C khó bị phát hiện bằng cách mã hóa lưu lượng, ẩn thông tin C&C trong các trường dự phòng của giao thức TCP/IP hoặc sử dụng mạng xã hội trực tuyến (OSN) để xây dựng các kênh bí mật [30] [44] [5] [1] [2]. Tuy nhiên, các phương pháp này yêu cầu sửa đổi phức tạp đối với kiến trúc botnet hoặc mã nguồn, điều này đặt ra yêu cầu cao về năng lực đối với bộ điều khiển botnet và cũng có tác động nhất định đến tính khả dụng và giá trị thị trường của botnet.

Với việc áp dụng rộng rãi học máy trong lĩnh vực phát hiện botnet [23] [10] [16] [20] [34] [42] [41], các nhà nghiên cứu bảo mật đã dần tìm kiếm các phương pháp học máy đối nghịch (AML) để vượt qua các bộ phát hiện lưu lượng botnet. . Đó là, xây dựng các mẫu đối đầu lưu lượng botnet bằng cách thêm nhiễu loạn vào các mẫu lưu lượng botnet, do đó bỏ qua bộ phát hiện lưu lượng botnet dựa trên ML.
Những phương pháp này có thể đạt được bằng cách áp dụng proxy lưu lượng thay vì sửa đổi mã nguồn botnet, đây dường như là một giải pháp hấp dẫn hơn và chi phí thấp hơn.

Hơn nữa, các phương pháp trốn tránh dựa trên AML có thể được chia thành hai loại tùy theo đầu ra khác nhau.

Tính năng tấn công không gian. đề cập đến các phương pháp chỉ có thể tạo các vectơ đặc trưng lưu lượng truy cập đối nghịch. Tuy nhiên, xét rằng quá trình ánh xạ các mẫu lưu lượng tới các vectơ đặc trưng là không thể đảo ngược, một cuộc tấn công như vậy không thể gây ra các mối đe dọa bảo mật thực sự và chỉ có thể sử dụng để chứng minh lỗ hổng của trình phát hiện dựa trên ML [25]. Evade RF [6] cố gắng làm cho lưu lượng mạng botnet khác với các luồng độc hại có trong bộ dữ liệu đào tạo máy dò bằng cách thay đổi một chút các đặc điểm thống kê cấp độ luồng, chẳng hạn như thời lượng luồng, cũng như số lượng byte được trao đổi và các gói được trao đổi. Tác giả đã đào tạo một khu rừng ngẫu nhiên làm công cụ phát hiện botnet trên CTU tập dữ liệu và đánh giá hiệu suất của các mẫu đối thủ được tạo. [29] đã thảo luận về hậu quả học tập và trốn tránh của khoảng cách giữa các mẫu đối thủ được tạo và tạo thủ công, và họ đã đạt được một cuộc tấn công đối thủ hộp trắng chống lại máy dò lưu lượng phần mềm độc hại C&C được mã hóa. Tuy nhiên, những công việc này đã đưa ra giả định phi thực tế rằng kẻ tấn công có kiến thức hoàn hảo về bộ phân loại và điều này rõ ràng là không phù hợp với các kịch bản tấn công thực tế.

Tấn công từ đầu đến cuối. đề cập đến một phương pháp tấn công có thể tạo ra lưu lượng truy cập thực như đầu ra. Phương thức tấn công này phù hợp hơn với các kịch bản tấn công mạng thực, bởi vì đầu ra có thể được sử dụng trực tiếp bởi kẻ tấn công. Tuy nhiên, một số ràng buộc chặt chẽ hơn sẽ được đưa ra khi áp dụng AML để tạo các mẫu lưu lượng truy cập độc hại:



Hình 1: Phân loại tránh Botnet

(1) Giữ nguyên chức năng độc hại trong mẫu bị xáo trộn.
(2) Không thể phá hủy cấu trúc tệp và cấu trúc giao thức mạng của tệp pcap. Điều này dẫn đến nhiều phương pháp AML trong lĩnh vực hình ảnh không thể được áp dụng trực tiếp, bởi vì chúng gây nhiễu các pixel hình ảnh một cách bừa bãi. [35] đã

đề xuất một phương pháp sử dụng công việc mạng đối nghịch chung (GAN) để bắt chước hành vi của lưu lượng truy cập mạng trò chuyện trên Facebook để vượt qua IPS tầng bình lưu tự điều chỉnh. Bài báo của họ đã cố gắng điều chỉnh hành vi của kênh liên lạc để bắt chước hành vi của lưu lượng truy cập mạng trò chuyện trên Facebook theo các đặc điểm (tổng kích thước byte, thời lượng và đồng bằng thời gian giữa luồng hiện tại và luồng tiếp theo) nhận được từ GAN. Tuy nhiên, bài viết không giải thích rõ ràng cách sửa đổi mã nguồn để đạt được những thay đổi được chỉ định bởi GAN và IPS được đề cập trong bài viết không phải là bộ phát hiện luồng mà là bộ 3 (IP nạn nhân, IP máy chủ, cổng máy chủ) máy dò.

Ngoài ra còn có một số công trình [36] đã sử dụng khả năng tạo của GAN để tạo lưu lượng mạng trông giống như thật nhất có thể. Mục đích của những công việc này là cải thiện chất lượng của bộ dữ liệu để huấn luyện các bộ phát hiện lưu lượng độc hại nhằm giải quyết vấn đề mất cân bằng dữ liệu, tuy nhiên không thể đảm bảo rằng lưu lượng mạng được tạo thực sự xảy ra trong liên kết dữ liệu. Hơn nữa, các phương pháp này không xem xét liệu lưu lượng được tạo thủ công có thể thực hiện chức năng độc hại của botnet hay không. Điều này dẫn đến một kịch bản hoàn toàn khác với kịch bản trong công việc của chúng tôi, trong đó chúng tôi nhắm đến việc bỏ qua việc phát hiện lưu lượng botnet bằng cách xây dựng các mẫu đối thủ của botnet.

3 MÔ HÌNH VÀ HỆ THỐNG Đe dọa KHUNG

Chúng tôi sử dụng quy trình quyết định Markov (MDP) để mô hình hóa vấn đề tạo mẫu luồng botnet đối thủ và thực hiện các cuộc tấn công hộp đen chung thông qua thuật toán RL. Phần này trước tiên giải thích mô hình mối đe dọa của phương pháp tấn công của chúng tôi, sau đó giới thiệu toàn bộ khuôn khổ của hệ thống được đề xuất và cuối cùng giới thiệu chi tiết các thành phần hệ thống quan trọng.

3.1 Mô hình mối đe dọa

Chúng tôi mô tả mô hình mối đe dọa của mình theo phương pháp được đề xuất trong [9].

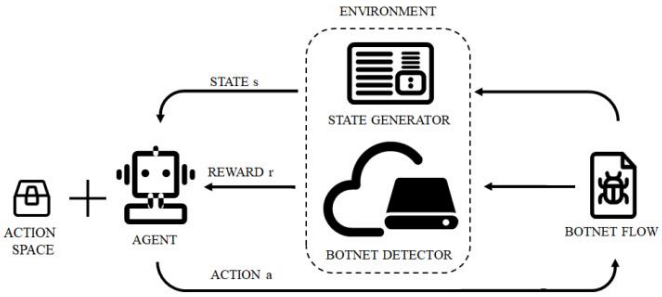
Mục tiêu của đối phương. Mục tiêu của kẻ tấn công là có thể đánh lừa máy dò bằng cách tạo ra các mẫu đối nghịch, do đó làm tăng khả năng tàng hình của nó. Từ quan điểm của bộ ba CIA (tính bảo mật, tính toàn vẹn và tính khả dụng), những kẻ tấn công cố gắng giảm tính khả dụng của hệ thống phát hiện xâm nhập mạng bằng cách ngụy trang luồng botnet.

Kiến thức của đối thủ. Kẻ tấn công hiểu rằng mạng mục tiêu có thể được bảo vệ bởi hệ thống phát hiện xâm nhập mạng cấp lưu lượng dựa trên học máy. Tuy nhiên, kẻ tấn công không cần nắm vững bất kỳ kiến thức nào trước đó về máy dò, chẳng hạn như thuật toán, tham số, tính năng hoặc dữ liệu huấn luyện.

Khả năng của đối thủ. Trong kịch bản trốn tránh dựa trên các cuộc tấn công sarial của đối thủ, kẻ tấn công có khả năng sửa đổi dữ liệu thử nghiệm chứ không phải dữ liệu đào tạo của máy dò. Mặc dù vậy, vì kẻ tấn công có toàn quyền kiểm soát botmaster và kiểm soát một phần bot, chúng tôi tin rằng kẻ tấn công cũng có thể cập nhật bot để thay đổi hành vi giao tiếp của chúng bằng cách thiết lập proxy. Đồng thời, chúng tôi giả định rằng kẻ tấn công có thể liên tục truy cập vào máy dò để lấy kết quả dự đoán nhị phân từ máy dò.

3.2 Thiết kế hệ thống Trong

bối cảnh tránh bị phát hiện botnet, chúng tôi khởi tạo vòng lặp phản hồi của RL, như trong Hình 2. Nhiệm vụ của chúng tôi là đào tạo tác nhân thông qua một loạt các tương tác với trình phát hiện botnet để nó có thể học cách tạo ra đối thủ ví dụ về luồng botnet để vượt qua trình phát hiện botnet.



Hình 2: Khung hệ thống

Nói chung, thiết lập hệ thống của chúng tôi bao gồm hai thành phần: agent và môi trường. Agent nhận phản hồi từ môi trường và sau đó chọn các hành động từ không gian hành động để sửa đổi luồng botnet theo thuật toán RL.

Môi trường chủ yếu bao gồm hai mô-đun: bộ phát hiện botnet và bộ tạo trạng thái. Trình phát hiện botnet là trình phát hiện mà chúng tôi muốn bỏ qua. Nhiệm vụ của nó là liên tục dự đoán lưu lượng độc hại trong vòng lặp và cung cấp kết quả nhị phân trở lại cho tác nhân như một phần thưởng.
Cần lưu ý rằng phần thưởng là một số thực {0, R}; nếu máy dò bị định hướng sai thành công, phần thưởng là R; mặt khác, nó là 0. Chúng tôi đặt R = 10 để đưa ra phản hồi tích cực mạnh mẽ khi tác nhân sửa đổi thành công mẫu lưu lượng botnet để vượt qua trình phát hiện. Trình tạo trạng thái được sử dụng để tạo mô tả về trạng thái mẫu hiện tại và giúp tác nhân đưa ra quyết định hành động.
Nói chung, trạng thái càng toàn diện và chính xác thì quyết định của đại lý sẽ càng tốt hơn.

Khi hệ thống bắt đầu chạy, bộ tạo trạng thái sẽ tạo trạng thái s dựa trên mẫu đầu vào. Sau đó, agent nhận biết trạng thái và chọn một hành động a từ không gian hành động để sửa đổi mẫu luồng botnet theo thuật toán RL. Sau đó, mẫu đã sửa đổi được nhập vào bộ phát hiện botnet và thu được kết quả dự đoán. Tiếp theo, agent nhận phần thưởng r theo phản hồi nhị phân và trình tạo trạng thái tạo trạng thái tiếp theo theo mẫu đã sửa đổi. Vòng lặp tiếp tục cho đến khi máy dò bị đánh lừa thành công hoặc số lượng thao ´ s tác đạt đến giới hạn trên (nếu máy dò vẫn không được bỏ qua thành công, chúng tôi bắt đầu một lần thử sửa đổi mẫu khác).

3.3 Thuật toán học tăng cường

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 [39].
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, tác nhân 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 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.
Xem xét rằng không gian hành động của chúng tôi không liên tục, chúng tôi chọn hai thuật toán RL dựa trên giá trị cổ điển để so sánh xem thuật toán nào phù hợp hơn cho kịch bản ứng dụng được đề xuất.

DQN. Q-learning là một phương pháp chênh lệch thời gian (TD) ngoài chính sách sử dụng bảng Q để ước tính giá trị Q để chọn hành vi tốt nhất. Tuy nhiên, khi không gian trạng thái hoặc không gian hành động lớn, Q-learning gặp vấn đề về "thảm họa chiều". Đòn bẩy DQN tăng tuổi mạng nơ-ron để ước tính hàm giá trị Q [26] và chuyển sang hàm Q(s, a; θ) để tính gần đúng Q(s, a) thay vì Q_table, trong đó θ là tham số mạng. Hàm mất mát cho mạng L(θ) được định nghĩa là sai số bình phương giữa giá trị Q mục tiêu và đầu ra giá trị Q từ mạng.

$$L(\theta) = E[(\text{TargetQ} \quad Q(s, a; \theta))2 \,] \tag{1}$$

Ở đâu

$$\text{TargetQ} = r_{t+1} + \gamma \cdot \underset{\text{Mọi}}{\text{tối đa}} \quad Q(s_{t+1}, a; \theta)$$

trong đó γ là hệ số giảm giá, s và a lần lượt biểu thị trạng thái và hành động hiện tại, st+1 biểu thị trạng thái tiếp theo và rt+1 là phần thưởng của hành động a trong trạng thái s (Những điều này giống nhau trong phương trình sau).

dịch SARS. Trạng thái-hành động-phần thưởng-trạng thái-hành động (SARSA) là một thuật toán chính sách cho việc học TD. Giá trị Q tối ưu, được ký hiệu là Q(st , at), có thể được biểu thị như sau:

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

Có nhiều điểm khác biệt giữa Q-learning và SARSA.
Đầu tiên, SARSA theo chính sách học giá trị Q dựa trên hành động được thực hiện bởi chính sách hiện tại, trong khi Q-learning ngoài chính sách thực hiện điều này so với chính sách tham lam. Lấy bài toán đi bộ trên vách đá làm ví dụ, DQN có xu hướng chọn phần thưởng âm lớn khi khám phá, trong khi SARSA có xu hướng chọn con đường an toàn và tránh con đường tối ưu nguy hiểm. Điều này có nghĩa là một tác nhân Q-learning có thể rơi khỏi vách đá tại một điểm an toàn do chọn thăm dò.
DQN táo bạo hơn, trong khi SARSA bảo thủ hơn.

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

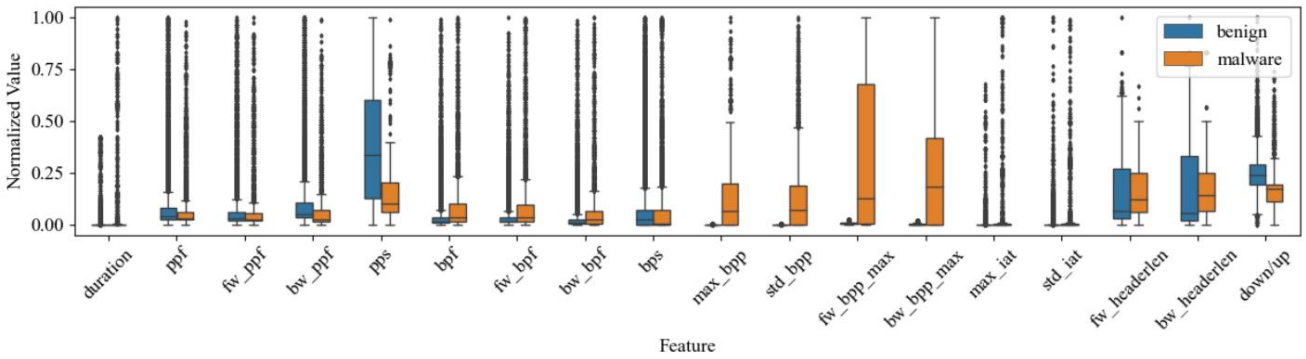
Vì những lý do trên, chúng tôi đã triển khai hai thuật toán RL này và đánh giá hiệu suất của chúng thông qua một loạt thử nghiệm.
Kết quả được trình bày trong Phần 5.

3.4 Không gian hành động

Không gian hành động bao gồm một loạt các hành động sửa đổi cho luồng botnet.

Phát hiện cấp độ luồng botnet dựa trên ML là dựa trên sự bất thường để xác định mạng botnet dựa trên sự khác biệt về một số tính năng giữa luồng botnet và luồng bình thường. Các mô hình ML thực hiện trích xuất tính năng trước khi phân loại, điều này sẽ gây ra một số nén thông tin và mất thông tin, bất kể nó được thực hiện thủ công hay tự động dựa trên mạng thần kinh. Chúng tôi hy vọng rằng sự nhiễu loạn được thêm vào mẫu ban đầu có thể gây nhầm lẫn một số tính năng nhất định của mẫu độc hại với mẫu bình thường sau khi trích xuất tính năng, mà bộ phát hiện botnet coi là dấu hiệu của các mẫu lành tính.

Nhưng như đã nói ở trên, điều quan trọng nhất để sửa đổi luồng botnet là không ảnh hưởng đến các chức năng độc hại của nó. Vì vậy, chúng tôi



Hình 3: Boxplot của 18 tính năng luồng chuẩn hóa cho mạng botnet và luồng thông thường

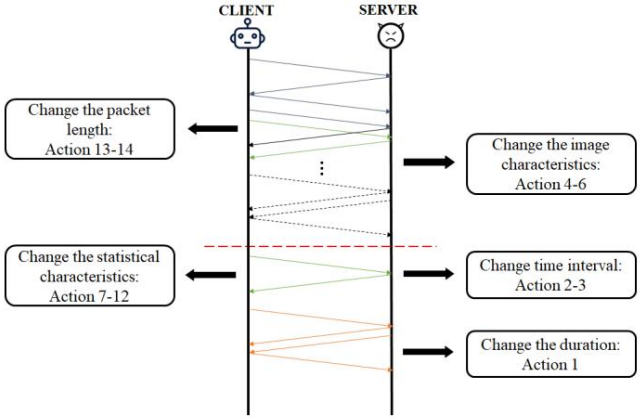
không thể xóa các gói lưu lượng theo ý muốn. Lựa chọn duy nhất là áp dụng các sửa đổi cho các khu vực không ảnh hưởng đến việc thực hiện các chức năng độc hại hoặc thêm các gói dữ liệu mới. Thông qua phân tích luồng botnet, chúng tôi thấy rằng nội dung độc hại của botnet được gói gọn trong lớp ứng dụng, do đó, các hoạt động gia tăng ở lớp vận chuyển sẽ không ảnh hưởng đến các chức năng độc hại ban đầu.

Để xác định tính năng nào sẽ bị xáo trộn, chúng tôi tham khảo các công trình phát hiện botnet dựa trên ML, chẳng hạn như [23] [10], [16] [20]. Chúng tôi nhận thấy các nhà nghiên cứu có xu hướng trích xuất một số đặc điểm phân biệt dựa trên cơ chế hoạt động của botnet và các đặc điểm này thường có mức độ chồng chéo cao trong các công việc khác nhau.

Tính đến độ khó của việc thiết kế hành động, chúng tôi chọn 18 tính năng từ tập hợp các tính năng thường được sử dụng trong phát hiện mạng bot, bao gồm thời lượng, gói trên mỗi luồng (ppf), gói trên giây (pps), luồng trước byte (bpf), byte trên giây (bps), thời gian giữa các lần đến (iat), tỷ lệ tăng/giảm, v.v. (fw:forward, bw:backward). Những đặc điểm này có thể dễ dàng bị ảnh hưởng bởi việc thiết kế cẩn thận thời gian, kích thước và hướng của gói được thêm vào. Hình 3 cho thấy sơ đồ hộp của 18 giá trị tính năng được chuẩn hóa trong bộ dữ liệu đào tạo của chúng tôi cho mạng botnet và các luồng lành tính.

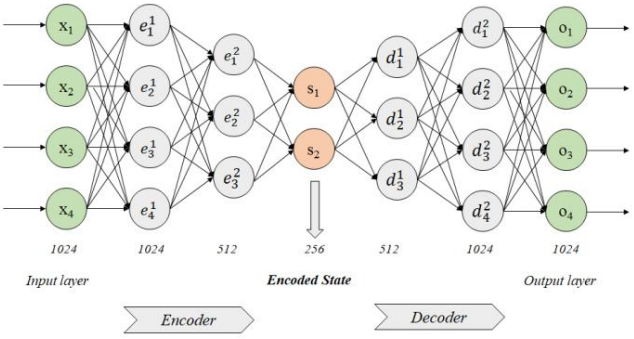
Chúng ta có thể quan sát thấy rằng do cơ chế hoạt động độc đáo của botnet, có một số khác biệt giữa các phạm vi giá trị cho một số tính năng trong mạng botnet và các luồng thông thường. Ví dụ: botnet sẽ gửi một số lượng lớn các gói tin nhịp tim ngắn để xác nhận xem kết nối có được duy trì hay không, do đó, nó có ppf nhỏ hơn; tải xuống các ứng dụng độc hại và truyền thông tin cá nhân ở phía bot sẽ dẫn đến bpp lớn hơn; các bot cần gửi một lượng lớn thông tin bí mật để đáp ứng các lệnh ngắn từ quản trị viên bot, vì vậy lưu lượng botnet có xu hướng có tỷ lệ tăng/giảm nhỏ.

Dựa trên khám phá này, không gian hành động của chúng tôi bao gồm 14 hành động, có thể ảnh hưởng đến các đặc điểm thống kê nêu trên của lớp vận chuyển bằng cách sửa đổi dấu thời gian của gói dữ liệu hoặc thêm các gói được xây dựng cẩn thận. Khi xây dựng các gói mới, chúng tôi chủ yếu xem xét ba thuộc tính: dấu thời gian, hướng và kích thước gói. 14 hành động này được chia thành 5 loại tùy theo đối tượng mà chúng dự định tác động, như được tóm tắt trong Hình 4:



Hình 4: Không gian hành động

- Thay đổi thời lượng
  1) Giữ lại gói TCP FIN cuối cùng trong 1-3 giây ngẫu nhiên
- Thay đổi khoảng thời gian
  2) Thêm gói chuyển tiếp với khoảng thời gian 20 giây ở cuối
  3) Thêm một gói tin quay lại với khoảng cách 20 giây ở cuối
- Thay đổi đặc điểm hình ảnh: (đối với mô hình DNN) chú ý đến vị trí và nội dung của gói tin
  4) Thêm một gói rỗng tại vị trí ngẫu nhiên (0 < loc < 8)
  5) Thêm một gói ngẫu nhiên tại vị trí ngẫu nhiên (0 < loc < 8)
  6) Thêm gói 0 đầy đủ tại vị trí ngẫu nhiên (0 < loc < 8)
- Thay đổi các đặc tính thống kê: (đối với mô hình không phải DNN) chú ý đến hướng và kích thước của gói
  7) Thêm gói 0 kích thước lớn chuyển tiếp
  8) Thêm gói 0 kích thước lớn ngược
  9) Thêm gói kích thước trung bình chuyển tiếp không có nội dung ở đầu
  10) Thêm gói kích thước trung bình ngược không có nội dung tại cái đầu
  11) Thêm gói trống về phía trước 12) Thêm gói trống về phía sau
- Thay đổi độ dài gói 13) Thêm độ dài ngẫu nhiên bằng 0 vào cuối gói với xác suất 0,2 14) Chọn hai gói không có trọng tải, thêm ký tự ' 0' đến kích thước trung bình



Hình 5: Chi tiết Trình tạo trạng thái

Mục đích của danh mục "Thay đổi thời lượng" là ảnh hưởng đến thời lượng; "Thay đổi khoảng thời gian" là thay đổi IAT; "Thay đổi độ dài gói" được sử dụng để làm phiền bpp; "Thay đổi đặc điểm hình ảnh " chứa các hành động để thay đổi đầu vào của bộ phát hiện botnet dựa trên mạng thần kinh sâu, để chúng tập trung vào vị trí và tải trọng của gói dữ liệu mới được chèn vào; Các hành động trong "Thay đổi các đặc điểm thống kê" là làm xáo trộn toàn diện các đặc điểm thống kê ở trên, do đó hướng và kích thước của gói dữ liệu chủ yếu được xem xét.

3.5 Không gian trạng

thái Xem xét rằng phản hồi nhị phân của bộ phát hiện chứa quá ít thông tin để tác nhân sử dụng, chúng ta cần một bộ tạo trạng thái để chuyển trạng thái của luồng botnet tới tác nhân. Để mô tả chính xác trạng thái của các mẫu luồng botnet hiện tại, chúng tôi sử dụng bộ mã hóa tính năng có cấu trúc sâu – bộ mã hóa tự động xếp chồng (SAE) – để tự động trích xuất các tính năng luồng botnet và đưa chúng trở lại tác nhân dưới dạng trạng thái.

SAE là một mạng thần kinh bao gồm một số lớp mã hóa tự động thưa thớt, trong đó đầu ra của mỗi lớp ẩn được kết nối với đầu vào của lớp ẩn kế tiếp. SAE lần đầu tiên được đề xuất bởi Bengio et al. [số 8]. Để tránh vấn đề độ dốc biến mất tiềm ẩn của mạng sâu, đào tạo SAE được thực hiện bằng cách sử dụng tinh chỉnh trước đào tạo không giám sát và tinh chỉnh có giám sát. Ở một mức độ nào đó, các mạng được đào tạo trước tạo điều kiện hội tụ lặp lại trong giai đoạn được giám sát vì chúng phù hợp với cấu trúc của dữ liệu huấn luyện, làm cho giá trị ban đầu của toàn bộ mạng có trạng thái phù hợp. Bởi vì mỗi lớp dựa trên các tính năng được trích xuất bởi lớp trước, SAE có thể trích xuất các tính năng phức tạp và trừu tượng cao. SAE đã đạt được hiệu suất đáng chú ý trên nhiều tác vụ tiền xử lý tính năng và giảm kích thước.

Cụ thể, chúng tôi lấy 1024 byte đầu tiên của mỗi tệp luồng botnet (vì một vài gói đầu tiên, tối đa 20 gói đầu tiên, đã được chứng minh là đủ để có độ chính xác chính xác, ngay cả đối với lưu lượng được mã hóa [42]) làm đầu vào cho Mô hình SAI. Sau một số giai đoạn đào tạo, mô hình SAE có thể tự động tìm hiểu một vectơ trạng thái 256 chiều của luồng botnet.

Khi xác định thứ nguyên trạng thái, chúng tôi kiểm tra thứ nguyên 128 và 256. Dưới sự đánh đổi giữa chi phí thời gian đào tạo và hiệu ứng trốn tránh, cuối cùng chúng tôi đã đặt số thứ nguyên đối tượng thành 256.

4 THIẾT LẬP THỬ NGHIỆM

4.1 Triển khai Bằng cách đề

cập đến việc triển khai Tor, chúng tôi triển khai hệ thống của mình như một proxy đối nghịch trong môi trường mạng, như thể hiện trong Hình 6. Kẻ tấn công có thể dễ dàng triển khai proxy đối nghịch ở phía botmaster, trong khi ở phía bot, kẻ tấn công có thể đạt được điều này bằng cách cập nhật bot thông qua kênh C&C ban đầu. Do đó, phương pháp của chúng tôi có thể được triển khai mà không cần sửa đổi phức tạp đối với phần mềm độc hại ban đầu.

Theo kịch bản triển khai này, tất cả lưu lượng giao tiếp giữa kẻ tấn công và bot sẽ đến proxy trước. Do đó, kẻ tấn công có thể giám sát lưu lượng truyền thông botnet và proxy đối thủ được trang bị tác nhân RL được đào tạo có thể thực hiện các hành động gia tăng đối với luồng botnet cho đến khi nó vượt qua bộ phát hiện thành công. Theo cách này, thứ mà máy dò thu được là lưu lượng giao tiếp botnet đã được xử lý bởi tác nhân đối nghịch và rất có khả năng bỏ qua nó.

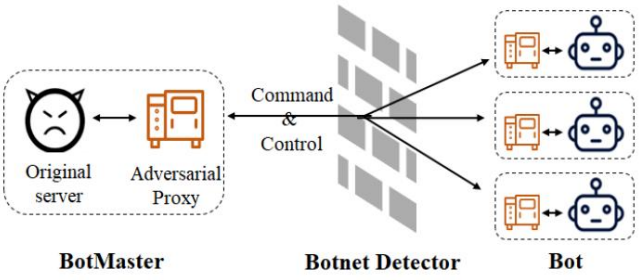
Trong một kiến trúc tấn công và phòng thủ như vậy, kẻ tấn công có thể tránh được sự phát hiện botnet dựa trên tấn công đối thủ trong một mạng xã hội. kịch bản hộp đen hoàn toàn.

Với mục đích thu hút cộng đồng, chúng tôi triển khai khung RL của mình với OpenAI gym [12]. Cụ thể, chúng tôi triển khai các tác nhân SARSA và DQN bằng keras-rl [33].

4.2 Bộ dữ liệu

Việc đánh giá hiệu suất của bất kỳ phương pháp phát hiện nào đều yêu cầu thử nghiệm với dữ liệu đủ không đồng nhất để mô phỏng lưu lượng truy cập thực ở mức chấp nhận được. Chúng tôi chọn hai bộ dữ liệu công khai: CTU [17], được nắm bắt bởi Dự án Cơ sở Bắt giữ Phần mềm độc hại, đây là một dự án nghiên cứu chịu trách nhiệm giám sát liên tục bối cảnh mối đe dọa đối với các mối đe dọa mới nổi, truy xuất các mẫu độc hại và chạy chúng trong các cơ sở để nắm bắt lưu lượng; và ISOT [37], được tạo bằng cách hợp nhất các bộ dữ liệu có sẵn khác nhau. Nó chứa cả lưu lượng độc hại (dấu vết của botnet Storm và Zeus) và lưu lượng không độc hại (gói trò chơi, lưu lượng HTTP và ứng dụng P2P). Bộ dữ liệu chứa dữ liệu theo dõi cho nhiều hoạt động mạng trải dài từ web và email đến phương tiện truyền thông sao lưu và truyền trực tuyến. Sự đa dạng về lưu lượng này làm cho nó giống với lưu lượng mạng trong thế giới thực.

Chúng tôi chọn 10 họ botnet từ các bộ dữ liệu công khai này để tạo thành một bộ dữ liệu mới với các cân nhắc sau:



Hình 6: Triển khai hệ thống của chúng tôi

Bảng 2: Chi tiết bộ dữ liệu

Được sử dụng cho	Số sê-ri tệp gia đình	botnet	kênh.	phiên ban đầu	Đã sử dụng phiên	bị bắt năm	ghi chú
Xe lửa & Bài kiểm tra	IRC	tinh thần	CTU-47	4,507	4.000	2011	
		người máy	ĐHCT-44/45/52	31,736	10.000	2011	Một mạng botnet dựa trên IRC với 40.000 thành viên tích cực
		Murio	ĐHCT-49	8,381	8.000	2011	
	HTTP	virut	ĐHCT-46/54	31,173	30.000	2011	Chịu trách nhiệm cho 5,5% số vụ lây nhiễm phần mềm độc hại trong quý 3 năm 2012. Nó đã
		Miuref(3ve)	CTU-127	13,426	10.000	2015	lây nhiễm khoảng 1,7 triệu máy tính
		Neris	ĐHCT-42/43	31,133	30.000	2011	
		HTbot	CTU-348	36,855	30.000	2018	Một trong những botnet ngân hàng Android được báo cáo lớn nhất được biết đến cho đến nay
		Dridex/Necurs	CTU-346	10.029	10.000	2018	Một trong những botnet thư rác lớn nhất thế giới
		kẻ lừa bịp	CTU-327	30.052	30.000	2018	Mối đe dọa kinh doanh hàng đầu năm 2018
	P2P	Bão	ISOT-Bão	4.672	4.672	2012	Đã lây nhiễm hơn 1 triệu hệ thống
đào tạo	bình thường	–	ISOT-bình thường	511,322	80.000	2010	

• Tính đa dạng, bộ dữ liệu bao gồm hầu hết các loại kênh truyền thông botnet chính thống (IRC, HTTP, P2P) và các đặc điểm lưu lượng khác nhau đáng kể. • Thông thường, bộ dữ liệu bao gồm lưu lượng truy cập của các họ botnet điển hình, chúng gây ra các cuộc tấn công lớn, có số lượng lớn máy chủ được kiểm soát hoặc áp dụng các phương pháp ẩn giấu nâng cao. • Khoảng thời gian lớn, đối với một nhóm duy nhất, mỗi tệp lưu lượng truy cập mất nhiều thời gian để nắm bắt. Nhìn chung, bộ dữ liệu đó bao gồm lưu lượng botnet từ 2011-2018. Điều này làm cho bộ dữ liệu linh hoạt hơn và mới lạ hơn các bộ dữ liệu hiện có khác.

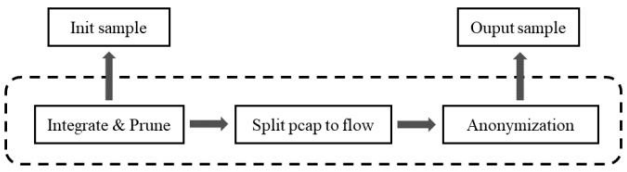
Chúng tôi tóm tắt các họ botnet và thời gian bắt giữ chúng, giới thiệu ngắn gọn và số lượng mẫu phiên được sử dụng cho thử nghiệm trong Bảng 2.

Sau khi có được tập dữ liệu, chúng tôi tiến hành tiền xử lý dữ liệu tiến trình, như thể hiện trong Hình 7.

Bước 1: Tích hợp & cắt tỉa. Chúng tôi 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.

Bước 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 quốc gia đích có thể hoán đổi cho nhau trong một bộ 5 (SIP, SPort, DIP, DPort, Giao thức). Cụ thể, chúng tôi sử dụng SplitCap [3] để chia từng tệp pcap thành các phiên.

Bước 3: Ẩn danh & làm sạch. Tệp lưu lượng truy cập của chúng tôi được tạo từ các môi trường mạng khác nhau. Để loại bỏ IP và MAC địa chỉ' trên máy dò, thông tin duy nhất của lưu lượng truy cập dữ liệu cần phải được ngẫu nhiên hóa. Cụ thể, chúng tôi thay thế chúng bằng một địa chỉ mới được tạo ngẫu nhiên.



Hình 7: Tiền xử lý dữ liệu

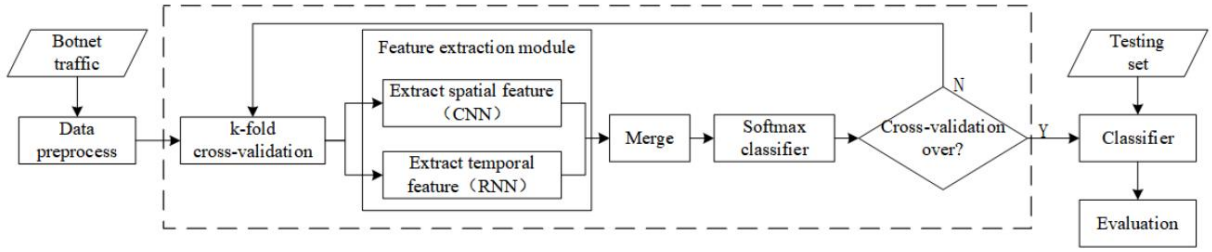
4.3 Máy dò

Trong hệ thống của chúng tôi, chức năng của trình phát hiện là dự đoán luồng botnet đã sửa đổi và cung cấp kết quả nhị phân trở lại cho tác nhân. Để so sánh, chúng tôi chọn hai mô hình phát hiện tiên tiến nhất trong các thử nghiệm của mình: mô hình phát hiện DL tổng hợp kết hợp CNN với LSTM và mô hình phát hiện ML không phân biệt dựa trên trên XGBoost làm công cụ phát hiện botnet của chúng tôi.

Mô hình phát hiện BotCatcher. Chế độ giao tiếp giữa máy chủ C&C và bot khác biệt đáng kể so với chế độ giao tiếp giữa những người dùng bình thường, do đó, lưu lượng truy cập bất thường do botnet tạo ra có thể được phát hiện thông qua phân tích lưu lượng. BotCatcher [43] sử dụng thuật toán học sâu để tự động trích xuất các đặc điểm không gian và thời gian từ lưu lượng mạng và huấn luyện bộ phân loại softmax tương ứng. Chúng tôi hiển thị khung hệ thống trong Hình 8.

Xem xét rằng CNN có khả năng trích xuất các tính năng cục bộ và nhận ra sự tương đồng về không gian, RNN có khả năng xử lý dữ liệu chuỗi và nhận ra sự tương đồng về thời gian. Trong mô-đun học tính năng, BotCatcher sử dụng CNN để trích xuất các tính năng không gian bằng cách chuyển đổi dữ liệu phiên botnet thành hình ảnh màu xám và tận dụng LSTM để tìm hiểu các tính năng tạm thời của chuỗi gói. Sau khi xử lý hai loại tính năng này thông qua mạng thần kinh đa lớp, BotCatcher đặt chúng vào một lớp softmax để xác định các mẫu lưu lượng truy cập bất thường. Tác giả đã sử dụng bộ dữ liệu CTU để xác minh tính hiệu quả của mô hình được đề xuất, trong khi chúng tôi đạt được tỷ lệ phát hiện 99,6% trên bộ dữ liệu của mình.

Mô hình phát hiện XGBoost. XGBoost là viết tắt của eXtreme Gradient Boosting, đây là một triển khai hiệu quả của các máy tăng cường độ dốc do Tianqi Chen tạo ra. Nó được thiết kế cho tốc độ và hiệu suất, đó là lý do tại sao gần đây nó đã thống trị máy học ứng dụng và các cuộc thi Kaggle về dữ liệu có cấu trúc hoặc dạng bảng.



Hình 8: Khung hệ thống BotCatcher

Bảng 3: Bộ tính năng của máy dò XGBoost.

Tính năng	Tính năng quan trọng	Tầm quan
max_fiat	0,188713 max_forward_pkt_len 0,112026	trọng 0,003005
std_thời	std_backward_pkt_len 0,110066	0,002809
lượng	total_pkt_len std_pkt_len	0,002221
fiat forward_header_len	0,074335 0,103142	0,001829
total_fiat	mean_forward_pkt_len 0,071723 mean_pkt_len	0,001568
mean_fiat	0,066497 min_pkt_len 0,06Ex57	0,001502
flowPktsPerSecond	0,06Ex571	0,001372
fPktsPerSecond		0,000719
bPktsPerSecond	max_backward_pkt_len	0,000523
	0.028872 mean_backward_pkt_len 0.000457 total_forward_pkt_len	
IOPRbackward_header_len	0,02652 0.021164 flowBytesPerSecond	0,000392
SameLenPktRatio	min_forward_pkt_len 0.01633 0.015546	0,000327
max_biat	0.013848 std_forward_pkt_len	0,000261
total_biat	total_backward_pkt_len 0.012542	0,000261
total_packets	fBytesPerSecond 0.011431 min_backward_pkt_len	0,000196
mean_biat	min_fiat 0.007773 min_biat 0.004377	0,000131
std_biat	bBytesPerSecond	6.53E-05
max_pkt_len	0,010255	0
total_fpackets		0
total_bpackets		0

Dhaliwalet và cộng sự. [15] đã chọn XGBoost để xây dựng hệ thống IDS. Hiệu suất của XGBoost tốt hơn hiệu suất của nhiều mô hình khác vì nó có thể giải quyết hiệu quả vấn đề dư thừa dữ liệu và có thể được xử lý song song. Do đó, XGBoost rất phù hợp để xử lý các mạng trong thế giới thực.

Trong công việc của mình, các tác giả đã thực hiện thử nghiệm xác thực chéo bằng cách sử dụng tập dữ liệu NSL-KDD (tệp csv chứa các đặc điểm 41 chiều của lưu lượng truy cập độc hại), đạt được kết quả (98,7%) tốt hơn so với kết quả của các thuật toán học máy khác. Cấu trúc mô hình và cài đặt siêu tham số được sử dụng trong bài báo này hoàn toàn phù hợp với chúng đã sử dụng (họ đã điều chỉnh cài đặt siêu tham số) và chúng tôi thu được kết quả phát hiện (98,9%) tốt như công việc ban đầu trên tập dữ liệu của chúng tôi. Chúng tôi liệt kê các tính năng được trích xuất và tầm quan trọng của chúng trong Bảng 3.

5 KẾT QUẢ

Để đánh giá hệ thống của mình, chúng tôi chia tập dữ liệu thành bốn tập con riêng biệt: tập huấn luyện trình phát hiện, tập huấn luyện tác nhân, tập thử nghiệm trình phát hiện và tập thử nghiệm tác nhân, theo tỷ lệ 4:4:1:1. Sự khác biệt là để kiểm tra khả năng tổng quát hóa của mô hình tấn công của chúng tôi và để mô phỏng tốt hơn kịch bản tấn công thực sự, trong đó kẻ tấn công có thể không lấy được dữ liệu huấn luyện của máy dò mục tiêu.

Để so sánh hiệu suất của các thuật toán và trình phát hiện RL khác nhau , chúng tôi triển khai bốn phiên bản hệ thống sau: tác nhân SARSA–trình phát hiện BotCatcher, tác nhân SARSA–trình phát hiện XGBoost, DQN tác nhân–máy dò BotCatcher và tác nhân DQN–máy dò XGBoost. Mỗi tác nhân được đào tạo cho các vòng action_num sample_num.

5.1 Hiệu suất trốn tránh Tỷ lệ trốn

tránh minh họa xác suất mà ví dụ về đối thủ của luồng botnet có thể trốn tránh thành công máy dò. Bảng 4 so sánh tỷ lệ trốn tránh thử nghiệm của bốn ví dụ hệ thống sau khi đào tạo trên 10 họ botnet. Trong số đó, XGBoost-random và BotCatcher-random đại diện cho các tình huống trong đó chúng tôi sử dụng chiến lược ran dom để chọn các hành động từ không gian hành động nhằm sửa đổi luồng botnet. Một mặt, chúng là cơ sở để đo lường xem tác nhân đó có hiệu quả hay không. Nói cách khác, nếu hiệu suất của tác nhân không tốt bằng chiến lược ngẫu nhiên, điều đó có nghĩa là tác nhân đã không học được điều gì hữu ích. Từ Hình 1, chúng ta có thể thấy rằng kết quả của thuật toán RL tốt hơn kết quả của chiến lược ngẫu nhiên trong mọi trường hợp. Mặt khác, điều này cũng có thể chứng minh rằng không gian hành động của chúng tôi thực sự hiệu quả bởi vì ngay cả sự lựa chọn ngẫu nhiên cũng có thể bỏ qua máy dò với một xác suất nhất định.

Sự tương phản giữa các gia đình. Từ Bảng 4, chúng ta có thể thấy rằng tỷ lệ trốn tránh khác nhau giữa các họ botnet khác nhau; Storm thậm chí có tỷ lệ trốn tránh bằng 0 với XGBoost-SARSA. Với một không gian hành động không thay đổi, ảnh hưởng của các hành động hiện có trên các mẫu gia đình khác nhau sẽ khác nhau. Thông qua phân tích thống kê của từng mẫu gia đình, chúng tôi thấy rằng tỷ lệ trốn tránh thấp hơn có thể là do các phiên chứa số lượng gói lớn nhất (chẳng hạn như Storm, một mạng botnet dựa trên P2P có kích thước phiên lớn), do đó ảnh hưởng của việc thêm gói hoặc thay đổi dấu thời gian trên giá trị riêng là tương đối nhỏ. Ngoài ra, các đặc điểm của các mẫu botnet rất đa dạng so với các mẫu lành tính, khiến tác nhân không thể chuyển đổi nó thành một mẫu lành tính trong một số bước hành động giới hạn . Trong các ứng dụng thực tế, kẻ tấn công có thể đánh đổi giữa tỷ lệ trốn tránh và quy mô nhiễu loạn đối với mẫu lưu lượng và tăng action_num một cách thích hợp.

Tương phản giữa các đại lý. Bằng cách so sánh tỷ lệ trốn tránh của các phiên bản hệ thống với cùng một trình phát hiện nhưng các tác nhân khác nhau trong Hình 1, chúng tôi thấy rằng trong hầu hết các trường hợp, tác nhân SARSA hoạt động tốt hơn tác nhân DQN. Chúng tôi nghĩ rằng điều này là do sự khác biệt nội tại giữa hai thuật toán RL. SARSA là một thuật toán chính sách thận trọng hơn so với Q-learning. Q-learning luôn nghĩ đến việc tối đa hóa các hàm Q, bất kể các kết quả không phải maxQ khác. SARSA là một thuật toán thận trọng, quan tâm đến từng bước của quyết định và nhạy cảm với lỗi và cái chết. Do đó, khi chúng tôi nhắm đến việc tạo ra các ví dụ đối lập với

Bảng 4: Hiệu suất trốn tránh của các phiên bản hệ thống.

	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-Ngẫu nhiên	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-Ngẫu nhiên	17%	19%	20%	37%	37%	27%	48%	37%	42%	59%

Bảng 5: Hiệu suất thời gian của các phiên bản hệ thống.

	Menti	Rbot	Murio	Virut	Miuref	Neris	HTBot	Dridex	Trickbot	Storm
XGBoost-SARSA	1,35	3,21	1.14	2,27	1,42	2,93	2,56	1,81	2,33	-
XGBoost-DQN	2.14	5,26	1,87	3,76	2,34	4,45	4.11	3.01	4	1,61
XGBoost-Random	6.42	10.26	5.14	6,05	6.17	8,05	7.12	8,56	7.12	-
BotCatcher-SARSA	6.54	4,24	7.11	3,05	4,73	6.04	3.06	2.19	2,57	4,93
BotCatcher-DQN	8.3	6,36	7,48	3,33	5,82	4,44	3,33	2,38	3,54	4.02
BotCatcher-Ngẫu nhiên	11,43	11,91	11,14	9.03	9.18	10.11	8,05	7.13	9.06	10.16

Chúng tôi sử dụng số truy vấn trung bình cần thiết để tác nhân tạo các mẫu đối thủ hiệu quả nhằm đánh giá hiệu suất thời gian của hệ thống.

ít bước hơn và nhiễu loạn nhỏ, tác nhân SARSA có thể phù hợp hơn tác nhân DQN.

Sự tương phản giữa các máy dò. Như được hiển thị trong Bảng 4, tỷ lệ trốn tránh với XGBoost làm công cụ phát hiện cao hơn so với BotCatcher trong hầu hết các kết quả của họ botnet và chúng tôi tin rằng điều này là do các hành động có tác động lớn hơn đến các đặc điểm thống kê so với các đặc điểm hình ảnh. Bằng cách phân tích các tính năng thống kê của mô hình XGBoost được hiển thị trong Bảng 3 và không gian hành động được mô tả trong phần 3, chúng ta có thể thấy rằng (i) các hành động trong không gian hành động đều ảnh hưởng trực tiếp hoặc gián tiếp đến các tính năng thống kê, cho dù chúng được thiết kế cho các tính năng thống kê hoặc các tính năng hình ảnh; (ii) nhiều hành động trong không gian hành động thậm chí còn thay đổi các tính năng phụ thuộc nhiều nhất của XGBoost (hành động 1 thời lượng, hành động 2 hướng iat, add-pp hành động, hành động 9&10-header_len, v.v.). Do đó, chúng tôi kết luận rằng việc sửa đổi được nhắm mục tiêu và tính nhất quán cao giữa không gian hành động và không gian tính năng của máy dò chịu trách nhiệm chính cho tính dễ bị tổn thương của máy dò.

5.2 Hiệu suất thời gian Một điểm chính

trong các cuộc tấn công đối nghịch hộp đen là đưa ra số lượng truy vấn ít nhất cho mô hình mục tiêu: nếu phương pháp được đề xuất yêu cầu số lượng truy vấn rất cao, thì tính khả thi của nó trong bối cảnh thế giới thực sẽ bị ảnh hưởng. Chúng tôi sử dụng số truy vấn trung bình cần thiết để tác nhân tạo các mẫu đối thủ hiệu quả nhằm đánh giá hiệu suất thời gian của hệ thống. Các kết quả được thể hiện trong Bảng 5.

Từ quan điểm của các họ botnet. , thời gian thực hiện giữa các gia đình khác nhau không thay đổi đáng kể, có nghĩa là miễn là đại lý của chúng tôi được đào tạo tốt, nó sẽ có hiệu suất tốt hơn so với cơ sở bất kể điều gì.

họ botnet mà nó xử lý. Kết quả này cũng cho thấy thêm rằng hệ thống của chúng tôi có tính sẵn sàng cao. Nếu hệ thống của chúng tôi có khoảng cách lớn về hiệu suất thời gian khi đối mặt với các họ botnet khác nhau, thì kẻ tấn công có thể cân nhắc cẩn thận liệu hệ thống của chúng tôi có thể thích ứng với mạng botnet của hắn hay không.

Từ quan điểm của các thuật toán RL. , tác nhân được trang bị thuật toán DQN thường yêu cầu nhiều bước hơn để vượt qua trình phát hiện so với tác nhân được trang bị thuật toán SARSA. Điều này có nghĩa là hành động được DQN lựa chọn theo chính sách tham lam có thể không phải là hành động tối ưu ở trạng thái hiện tại, vì vậy tác nhân cần thực hiện các hành động bổ sung để tích lũy nhiễu nhằm bỏ qua bộ dò.

Từ quan điểm của các máy dò. ,số bước cần thiết để tác nhân bỏ qua BotCatcher nhiều hơn so với XGBoost, nghĩa là tác nhân cần nhiều bước lặp lại để đánh lừa BotCatcher hơn là đánh lừa XGBoost. Nói cách khác, XGBoost không chỉ có tỷ lệ trốn tránh cao hơn mà các mẫu trốn tránh cũng yêu cầu các bước tương đối ít hơn so với BotCatcher. Do đó, chúng ta có thể kết luận rằng XGBoost dễ bị tổn thương hơn BotCatcher theo các thử nghiệm của công việc này.

5.3 Hành động chiếm ưu thế

Các đột biến chiếm ưu thế đề cập đến các hành động thường xuyên nhất được thực hiện bởi tác nhân khi trốn tránh thành công máy dò. Mỗi họ botnet có những tính năng độc đáo riêng, do đó, dòng chảy của các họ khác nhau cũng khác nhau về một số đặc điểm nhất định. Ví dụ: ppf của Trickbot là 8,66, trong khi của HTBot là 28,34. Do đó, chúng tôi đoán rằng những ảnh hưởng được tạo ra bởi các hành động khác nhau sẽ khác nhau đối với mỗi họ, vì vậy các hành động chi phối của mỗi họ botnet nên

Bảng 6: Hành động chiếm ưu thế của tác nhân BotCatcher-SARSA

Số hành động	Menti	Rbot	Murio	Virut	Miuref	Neris	HTBot	Dridex	Trickbot	Geodo	Storm	Waledac
1	0,51	0,02	0,49		0,00	0,03	0,29	0,06	0,00	0,00	0,06	0,01	0,06
2	0,01	0,00	0,01	0,01	0,00	0,03	0,01	0,06	0,42	0,05	0,01	0,06
3	0,01	0,02	0,00	0,32	0,03	0,01	0,02	0,07	0,00	0,01	0,06	0,28
4	0,00	0,14	0,02	0,01	0,09	0,01	0,00	0,06	0,00	0,07	0,01	0,04
5	0,00	0,04	0,02	0,19	0,02	0,03	0,00	0,00	0,00	0,08	0,01	0,02
6	0,09	0,08	0,04	0,01	0,02	0,01	0,31	0,00	0,00	0,03	0,82	0,02
7	0,01	0,01	0,00	0,00	0,26	0,00	0,00	0,21	0,02	0,01	0,01	0,02
...	0,02	0,01	0,01	0,12	0,00	0,13	0,34	0,19	0,32	0,27	0,03	0,01
9	0,00	0,00	0,00	0,00	0,01	0,02	0,00	0,12	0,02	0,06	0,01	0,01
10	0,00	0,55	0,10	0,00	0,35	0,01	0,00	0,21	0,07	0,02	0,01	0,01
11	0,00	0,05	0,20	0,00	0,01	0,04	0,06	0,00	0,15	0,15	0,01	0,24
12	0,20	0,01	0,01	0,01	0,09	0,01	0,00	0,00	0,00	0,01	0,01	0,08
13	0,07	0,08	0,07	0,03	0,03	0,12	0,19	0,08	0,02	0,16	0,01	0,03
14	0,07	0,00	0,02	0,29	0,04	0,29	0,00	0,00	0,00	0,04	0,01	0,12

trở nên khác biệt. Để kiểm tra giả thuyết này, trong quá trình thử nghiệm, chúng tôi ghi lại danh sách hành động được thực hiện bởi tác nhân và đếm tần suất của từng hành động. Bảng 6 hiển thị kết quả trong phiên bản BotCatcher-SARSA, trong đó số hành động tương ứng với số được mô tả trong phần 3.

Từ Bảng 6, chúng ta có thể thấy rằng có sự khác biệt lớn trong phân bố và hành động chi phối của các họ khác nhau và những điều này thường liên quan đến đặc điểm và chức năng chính của dòng chảy gia đình khác nhau.

Lấy họ Rbot và Menti làm ví dụ, theo thống kê, chúng tôi nhận được rằng thời lượng trung bình của họ Rbot là 9,02 giây, trong khi của họ Menti là 2,91 giây. Đồng thời, chúng ta có thể thấy từ Bảng 6 rằng tác động của chanдetimestamp đối với các mẫu Rbot (0,02) ít hơn đáng kể so với các mẫu Menti (0,51). Nghĩa là, dòng họ có thời lượng ngắn có thể bị ảnh hưởng nhiều hơn bởi hành động dấu thời gian so với dòng họ có thời lượng dài. Do đó, chúng tôi xác định rằng ảnh hưởng của các hành động đối với các gia đình khác nhau có liên quan chặt chẽ đến các đặc điểm thống kê hoặc đặc điểm hình ảnh của mỗi gia đình.

6 KẾT LUẬN

Trong bài báo này, chúng tôi đề xuất một khung dựa trên RL chung để tạo các ví dụ về luồng botnet quảng cáo ngược, để khởi chạy các cuộc tấn công đối nghịch hộp đen chống lại các công cụ phát hiện luồng botnet dựa trên ML.

Để đảm bảo rằng các chức năng độc hại ban đầu của luồng botnet sẽ không bị ảnh hưởng khi sửa đổi luồng botnet, chúng tôi thiết kế một không gian hành động với 14 hành động bảo toàn chức năng. Những hành động này có thể thay đổi một số đặc điểm quan trọng của lớp vận chuyển nhưng sẽ không ảnh hưởng đến thông tin lớp ứng dụng có chứa các chức năng độc hại. Chúng tôi chọn 14 họ botnet để xây dựng một bộ dữ liệu botnet để đánh giá phương pháp của chúng tôi. Thông qua các thử nghiệm, chúng tôi chứng minh rằng các công cụ phát hiện botnet dựa trên ML thực sự dễ bị tấn công bởi các đối thủ và hệ thống của chúng tôi có thể đạt được tỷ lệ trốn tránh đáng kể cho các mô hình phát hiện botnet khác nhau với ít truy vấn hơn.

Mặc dù chúng tôi đạt được hiệu suất đáng kể, nhưng các phương pháp của chúng tôi có thể được cải thiện theo một số cách. Đầu tiên, chúng ta có thể khám phá các hành động bổ sung có thể đánh lừa máy dò bằng cách khám phá bộ tính năng mà máy dò phụ thuộc vào để thêm nhiễu loạn cho các tính năng này theo cách được nhắm mục tiêu. Thứ hai, các hành động hiện tại có tác động lớn đến luồng botnet và chúng tôi có thể giảm nhiễu loạn thông qua một phương pháp lặp đi lặp lại.

Chúng tôi tin rằng khuôn khổ được đề xuất trong bài báo này có thể thúc đẩy nghiên cứu các ví dụ về luồng botnet đối địch và có tác động tích cực đến lĩnh vực phát hiện botnet.

SỰ NHÌN NHẬN

Công trình này được hỗ trợ bởi Hiệp hội Xúc tiến Đổi mới Thanh niên CAS (Số 2019163), Quỹ Khoa học Tự nhiên Quốc gia Trung Quốc (Số 61902396), Chương trình Nghiên cứu Ưu tiên Chiến lược của Viện Hàn lâm Khoa học Trung Quốc (Số XDC02040100), Phòng thí nghiệm Trọng điểm phòng thí nghiệm về Công nghệ đánh giá mạng tại Học viện Khoa học Trung Quốc và Phòng thí nghiệm trọng điểm về Công nghệ bảo vệ và an ninh mạng Bắc Kinh.

NGƯỜI GIỚI THIỆU

[1] 2008. https://en.wikipedia.org/wiki/Conficker [2] 2008. https://en.wikipedia.org/wiki/Gh0st_RAT [3] 2011. SplitCap. https://www.netresec.com/?page=SplitCap.

[4] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, và Una-May O'Reilly. 2018. Học sâu đối thủ để phát hiện mạnh mẽ phần mềm độc hại được mã hóa nhị phân. Vào năm 2018, Hội thảo về Bảo mật và Quyền riêng tư của IEEE (SPW). IEEE, 76–82.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas và Yi Zhou. 2017. Hiểu về Mirai Botnet. Trong Hội nghị chuyên đề về bảo mật USENIX lần thứ 26 (USENIX Security 17). Hiệp hội USENIX , Vancouver, BC, 1093–1110. https://www.usenix.org/conference/ usenixsecurity17/technical-sessions/presentation/ antonakakis [6] Giovanni Apruzzese và Michele Colajanni. 2018. Trốn tránh các công cụ phát hiện botnet dựa trên các luồng và rừng ngẫu nhiên với các mẫu đối nghịch. Năm 2018, Hội nghị chuyên đề quốc tế lần thứ 17 của IEEE về ứng dụng và điện toán mạng (NCA). IEEE, 1–8.

[7] Shumeet Baluja và Ian Fischer. 2018. Học cách tấn công: Mạng chuyển đổi đối thủ.. Trong AAAI, Tập. 1. 3.

[8] Yoshua Bengio, Pascal Lamblin, Dan Popovici, và Hugo Larochelle. 2007. Đào tạo theo lớp tham lam của các mạng sâu. Trong Những tiến bộ trong xử lý thông tin thần kinh

các hệ thống. 153–160.

[9] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto và Fabio Roli. 2013. Các cuộc tấn công trốn tránh máy học tại thời điểm thử nghiệm. Trong hội nghị chung châu Âu về học máy và khám phá tri thức trong cơ sở dữ liệu. Springer, 387–402.

[10] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, và Christopher Kruegel. 2012. Tiết lộ: phát hiện các máy chủ điều khiển và chỉ huy mạng botnet thông qua phân tích luồng mạng quy mô lớn. Trong Kỷ yếu của Hội nghị Ứng dụng Bảo mật Máy tính Thường niên lần thứ 28. 129–138.

[11] Wieland Brendel, Jonas Rauber, và Matthias Bethge. 2017. Các cuộc tấn công đối nghịch dựa trên quyết định : Các cuộc tấn công đáng tin cậy chống lại các mô hình máy học hộp đen. bản in trước arXiv arXiv:1712.04248 (2017).

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schul man, Jie Tang, và Wojciech Zaremba. 2016. Mở phòng tập thể dục. bản in trước arXiv arXiv:1606.01540 (2016).

[13] Nicholas Carlini và David Wagner. 2017. Hướng tới đánh giá sự mạnh mẽ của mạng lưới thần kinh. Trong hội nghị chuyên đề về bảo mật và quyền riêng tư của IEEE năm 2017 (sp). IEEE, 39–57.

[14] Hung Dang, Yue Huang, và Ee-Chien Chang. 2017. Trốn tránh phân loại bằng cách biến hình trong bóng tối. Trong Kỷ yếu của Hội nghị ACM SIGSAC 2017 về Bảo mật Máy tính và Truyền thông. 119–133.

[15] Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid, và Robert Abbas. 2018. Hệ thống phát hiện xâm nhập hiệu quả sử dụng XGBoost. Thông tin 9, 7 (2018), 149.

[16] Jérôme François, Shaonan Wang, Thomas Engel, et al. 2011. BotTrack: theo dõi botnet sử dụng NetFlow và PageRank. Trong Hội nghị Quốc tế về Nghiên cứu Mạng. Springer, 1–14.

[17] Sebastian Garcia, Martin Grill, Jan Stiborek, và Alejandro Zunino. 2014. So sánh thực nghiệm các phương pháp phát hiện botnet. máy tính & bảo mật 45 (2014), 100–123.

[18] Ian J Goodfellow, Jonathon Shlens, và Christian Szegedy. 2014. Giải thích và khai thác các ví dụ đối lập. bản in trước arXiv arXiv:1412.6572 (2014).

[19] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, và Patrick McDaniel. 2017. Ví dụ đối thủ để phát hiện phần mềm độc hại. Trong Hội nghị chuyên đề châu Âu về nghiên cứu bảo mật máy tính. Springer, 62–79.

[20] Guofei Gu, Roberto Perdisci, Junjie Zhang, và Wenke Lee. 2008. Botminer: Phân tích phân cụm lưu lượng mạng để phát hiện botnet độc lập với giao thức và cấu trúc. (2008).

[21] Weiwei Hu và Ying Tan. 2017. Tạo các ví dụ về phần mềm độc hại đối nghịch cho các cuộc tấn công hộp đen dựa trên gan. bản in trước arXiv arXiv:1702.05983 (2017).

[22] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, và Fabio Roli. 2018. Các nhị phân phần mềm độc hại đối thủ: Né tránh việc học sâu để phát hiện phần mềm độc hại trong các tệp thực thi. Năm 2018, Hội nghị xử lý tín hiệu châu Âu lần thứ 26 (EUSIPCO). IEEE, 533–537.

[23] Satoshi Kondo và Naoshi Sato. 2007. Kỹ thuật phát hiện lưu lượng botnet bằng phân loại phiên C&C sử dụng SVM. Trong Hội thảo Quốc tế về An ninh. mùa xuân, 91–104.

[24] Alexey Kurakin, Ian Goodfellow, và Samy Bengio. 2016. Học máy đối thủ trên quy mô lớn. bản in trước arXiv arXiv:1611.01236 (2016).

[25] Zilong Lin, Yong Shi, và Zhi Xue. 2019. IDSGAN: Mạng quảng cáo đa dạng để tạo tấn công chống phát hiện xâm nhập. arXiv:1809.02077 [cs.CR]

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Kiểm soát cấp độ con người thông qua học tăng cường sâu. Thiên nhiên 518, 7540 (2015), 529–533.

[27] SEYed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, và Pascal Frossard. 2017. Những nhiễu loạn đối thủ phổ quát. Trong Kỷ yếu hội nghị IEEE về thị giác máy tính và nhận dạng mẫu. 1765–1773.

[28] SEYed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, và Pascal Frossard. 2016. Deepfool: một phương pháp đơn giản và chính xác để đánh lừa các mạng lưới thần kinh sâu. Trong Kỷ yếu của hội nghị IEEE về thị giác máy tính và nhận dạng mẫu. 2574–2582.

[29] Carlos Novo và Ricardo Morla. Năm 2020. Phát hiện dựa trên luồng và tránh lưu lượng truy cập C2 của phần mềm độc hại được mã hóa dựa trên proxy. Trong Kỷ yếu Hội thảo ACM lần thứ 13 về Trí tuệ nhân tạo và An ninh (Sự kiện ảo, Hoa Kỳ) (AISec'20). Hiệp hội Máy tính, New York, NY, USA, 83–91. https: //doi.org/10.1145/3411508.3421379 [30] Nick Pantic và Mohammad I. Husain.

2015. Ẩn lệnh và kiểm soát Botnet bằng Twitter (ACSAC 2015). Hiệp hội Máy tính, New York, NY, Hoa Kỳ, 10 trang. https://doi.org/10.1145/2818000.2818047 [31] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik và Ananthram Swami. 2017.

Các cuộc tấn công hộp đen thực tế chống lại máy học. Trong Kỷ yếu của hội nghị ACM Châu Á 2017 về bảo mật máy tính và truyền thông. 506–519.

[32] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, và Ananthram Swami. 2016. Những hạn chế của học sâu trong môi trường đối nghịch. Năm 2016, hội nghị chuyên đề châu Âu về bảo mật và quyền riêng tư của IEEE (EuroS&P). IEEE, 372–387.

[33] Matthias Plappert. 2016. máy ảnh-rl. https://github.com/keras-rl/keras-rl.

[34] Shahbaz Rezaei và Xin Liu. 2019. Học sâu để phân loại lưu lượng được mã hóa : Tổng quan. Tạp chí truyền thông IEEE 57, 5 (2019), 76–81.

[35] M. Rigaki và S. Garcia. 2018. Đưa GAN vào cuộc chiến dao kéo: Điều chỉnh giao tiếp của phần mềm độc hại để tránh bị phát hiện. Vào năm 2018, Hội thảo về Bảo mật và Quyền riêng tư của IEEE (SPW). 70–75. https://doi.org/10.1109/SPW.2018.00019

[36] Markus Ring, Daniel Schlör, Dieter Landes, và Andreas Hotho. 2018. Tạo lưu lượng truy cập mạng dựa trên luồng bằng cách sử dụng Mạng đối thủ chung. Máy tính & Bảo mật 82 (12 2018). https://doi.org/10.1016/j.cose.2018.12.012 [37] Sherif Saad, Issa Traore,

Ali Ghorbani, Bassam Sayed, David Zhao, Wei Lu, John Felix và Payman Hakimian. 2011. Phát hiện botnet P2P thông qua phân tích hành vi mạng và học máy. Năm 2011, hội nghị quốc tế thường niên lần thứ 9 về quyền riêng tư, bảo mật và tin cậy. IEEE, 174–180.

[38] Elizabeth Stinson và John C. Mitchell. 2008. Hướng tới đánh giá có hệ thống về tính khả dụng của các phương pháp phát hiện bot/botnet. WOOT 8 (2008), 1–9.

[39] Richard S Sutton và Andrew G Barto. 2018. Học tăng cường: Giới thiệu . nhà xuất bản MIT.

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, và Rob Fergus. 2013. Các tính chất hấp dẫn của mạng lưới thần kinh. bản in trước arXiv arXiv:1312.6199 (2013).

[41] Pablo Torres, Carlos Catania, Sebastian Garcia, và Carlos Garcia Garino. 2016. Một phân tích về các mạng thần kinh tái phát cho hành vi phát hiện botnet. Năm 2016, đại hội IEEE hai năm một lần của Argentina (ARGENCON). IEEE, 1–6.

[42] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, và Yiqiang Sheng. 2017. Phân loại lưu lượng phần mềm độc hại bằng cách sử dụng mạng thần kinh tích chập để học biểu diễn. Trong Hội nghị Quốc tế về Mạng Thông tin (ICOIN) năm 2017. IEEE, 712–717.

[43] Di Wu, Binxing Fang, Xiang Cui, và Qixu Liu. 2018. BotCatcher: Hệ thống phát hiện Botnet dựa trên deep learning. Infocomm-journal 39, 8 (2018), 18–28.

[44] Xiapu Luo, EWW Chan, và RKC Chang. 2008. Các kênh thời gian bí mật của TCP: Thiết kế và phát hiện. Năm 2008, Hội nghị quốc tế IEEE về các hệ thống và mạng đáng tin cậy với FTCS và DCC (DSN). 420–429. https://doi.org/10.1109/DSN. 2008.4630112