ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRƯƠNG ĐÌNH TRỌNG THANH
TRẦN THÚY ANH

ĐỒ ÁN CHUYÊN NGÀNH

NGHIÊN CỨU CHIẾN THUẬT GIĂNG BẪY THÍCH ỨNG CHO MẠNG KHẢ LẬP TRÌNH

A STUDY ON ADAPTIVE AND AGILE CYBER DECEPTION STRATEGY FOR SDN-ENABLED NETWORKS

KỸ SƯ NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh, 2023

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRƯƠNG ĐÌNH TRỌNG THANH - 20520766
TRẦN THÚY ANH - 20521085

ĐỒ ÁN CHUYÊN NGÀNH

NGHIÊN CỨU CHIẾN THUẬT GIĂNG BẪY THÍCH ỨNG CHO MẠNG KHẢ LẬP TRÌNH

A STUDY ON ADAPTIVE AND AGILE CYBER DECEPTION STRATEGY FOR SDN-ENABLED NETWORKS

KỸ SƯ NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
ThS. Đỗ Hoàng Hiển
ThS. Phan Thế Duy

TP.Hồ Chí Minh - 2023

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thành khóa luận, nhóm đã nhận được sự định hướng, giúp đỡ, các ý kiến đóng góp quý báu và những lời động viên của các giáo viên hướng dẫn và giáo viên bộ môn. Nhóm xin bày tỏ lời cảm ơn tới thầy Đỗ Hoàng Hiển, thầy Phan Thế Duy đã tận tình trực tiếp hướng dẫn, giúp đỡ trong quá trình nghiên cứu.

Nhóm xin gửi lời cảm ơn đến gia đình và bạn bè đã động viên, đóng góp ý kiến trong quá trình làm khóa luận.

Nhóm cũng chân thành cảm ơn các quý thầy cô trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM, đặc biệt là các thầy cô khoa Mạng máy tính và Truyền thông, các thầy cô thuộc bộ môn An toàn Thông tin đã giúp đỡ nhóm.

Trương Đình Trọng Thanh **Trần Thúy Anh**

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

S	Không gian các trạng thái tác tử
s_t	Trạng thái môi trường tại lượt t
A	Tập các hành động tác tử
a_t	Hành động tại lượt thứ t
P	Hàm chuyển
R	Hàm phần thưởng
π	Chỉ số chính sách của Mô hình học tăng cường
ω	Trọng số cho phần thưởng
γ	Chiết khấu cho phần thưởng
V^π	Chính sách tối đa hóa kì vọng
Q	Hàm giá trị cho trạng thái - hành động
θ	Trọng số cho mạng nơ-ron
$L(\theta)$	Hàm mất mát
π^*	Chỉ số chính sách tối ưu của Mô hình học tăng cường
α	Tham số tốc độ học
L	Các lệnh Linux đã biết
C	Các lệnh tấn công tùy chỉnh
MDP	Markov Decision Process
RL	Reinforcement learning

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

TÓM TẮT ĐỀ TÀI

Tính cấp thiết của đề tài nghiên cứu:

Đối với lĩnh vực an ninh mạng, bên cạnh các hệ thống phát hiện và chống xâm nhập (IDPS) có nhiệm vụ giám sát lưu lượng mạng, các hành vi đáng ngờ và cảnh báo cho người quản trị hệ thống, giúp phát hiện và ngăn ngừa các hành động phá hoại bảo mật hệ thống hoặc những hành động trong tiến trình tấn công và được thiết kế làm việc thụ động trong việc phát hiện, ngăn chặn sự tấn công của tin tặc thì các trình giăng bẫy, cụ thể là Honeypot là một hệ thống tài nguyên thông tin được xây dựng với mục đích giả dạng, đánh lừa những kẻ sử dụng và xâm nhập không hợp pháp, thu hút sự chú ý, ngăn không cho chúng tiếp xúc với hệ thống thật. Qua nhiều năm, các hệ thống Honeypot đã được phát triển và thành công tích hợp mô hình học máy vào việc phát hiện và phân loại các dạng xâm nhập. Tuy nhiên, đi kèm với những cải tiến này, các kỹ thuật tấn công ngày càng trở nên tinh vi và phức tạp, đặc biệt khi có sự xuất hiện của các botnet tự động càng khiến việc thu thập thông tin trở nên rắc rối.

Từ những khó khăn trên, chúng tôi đã nghiên cứu và trình bày một Honeypot thông minh sử dụng học tăng cường để chủ động tham gia và học hỏi từ các tương tác của kẻ tấn công. Nó điều chỉnh hành vi của mình đối với phần mềm độc hại tự động để tối ưu hóa khối lượng dữ liệu được thu thập. Phần mềm độc hại sử dụng các phương pháp tự động hóa cao để tạo mạng botnet toàn cầu. Các phương thức tự động này được sử dụng để tự lan truyền và thỏa hiệp các máy chủ. Honeypot đã được triển khai để nắm bắt các hành động tương tác tự động này. Các kỹ thuật học máy trước đây đã được sử dụng để mô hình hóa các tương tác mạng botnet hồi cứu. Chúng tôi phát triển một Honeypot sử dụng phương pháp học tăng cường với một hình thức không gian hành động trạng thái cụ thể để tương tác với phần mềm độc hại tự động. Nó so sánh chức năng với các

Honeypot tương tự nhắm đến sự tương tác của con người. Nó cũng chứng minh rằng các bộ dữ liệu được thu thập từ triển khai Honeypot thông minh lớn hơn đáng kể so với triển khai tương tác cao tiêu chuẩn và các Honeypot thích ứng hiện có.

CHƯƠNG 1. TỔNG QUAN

Chương này giới thiệu về vấn đề và các nghiên cứu liên quan. Đồng thời, trong chương này chúng tôi cũng trình bày phạm vi và cấu trúc của đề tài.

1.1. Giới thiệu vấn đề

Trong suốt thập kỷ vừa qua, trên thế giới người ta đã phát triển một số công cụ nhằm phát hiện, phòng chống các cuộc tấn công vào hệ thống công nghệ thông tin mà các công ty, tổ chức đang phải đối mặt. Đó là tường lửa với chức năng bảo vệ, ngăn chặn các hoạt động của kẻ tấn công, hệ thống phát hiện truy cập trái phép (Intrustion Detection Systems - IDS) nhằm phát hiện các xâm nhập, cũng như có các hành động phòng chống lại các xâm nhập đó, hoặc giảm thiểu ảnh hưởng của nó. Tuy nhiên, với sự phát triển của tội phạm công nghệ ngày càng đa dạng về cả số lượng và phương thức thì hệ thống phát hiện xâm nhập dựa trên các dấu hiệu được định nghĩa trước sẽ không thể phát hiện ra vì trong cơ sở dữ liệu của IDS, Firewall không có dấu hiệu để nhận ra các tấn công này. Chính vì vậy, chúng chỉ có khả năng phát hiện các xâm nhập đã được định nghĩa trước. Do đó, để bảo vệ hệ thống mạng, người quản trị cần liên tục cập nhật thông tin về các lỗ hổng, các phương thức của kẻ tấn công. Ngày nay, nhiều tổ chức phi lợi nhuận, giáo dục sử dụng hệ thống gọi là ot để nghiên cứu về chiến lược, cách thức của các nhóm tin tặc (blackhat) để chống lại họ. Với Honeypot thì các lỗ hổng bảo mật được mở ra hoàn toàn. Mục đích của Honeypot là phát hiện và học hỏi từ các cuộc tấn công, sau đó sử dụng những thông tin đó để nâng cao an toàn thông tin. Người quản trị mạng trực tiếp thu được thông tin về các nguy hiểm trong hệ thống. Các lỗ hổng bảo mật chưa được phát hiện có thể được nhận ra và nghiên cứu để khắc phục từ thông

tin do Honeypot thu thập được. Honeypot đóng vai trò quan trọng trong việc phát hiện và ngăn chặn các cuộc tấn công mạng. Thiết kế Honeypot đã thay đổi theo thời gian khi các phương pháp và mục tiêu tấn công mạng đã phát triển. Không giống như hệ thống phát hiện xâm nhập (IDS) và hệ thống ngăn chặn xâm nhập (IPS), Honeypot được triển khai để thu thập dữ liệu nhằm phân tích hồi cứu. Tuy nhiên, Honeyot không đưa ra các biện pháp bảo mật chủ động và chỉ tập trung vào việc thu thập dữ liệu.

Qua nhiều công trình nguyên cứu, cuối cùng Honeypot đã phát triển để đáp ứng các lệnh của kẻ tấn công với mục tiêu kéo dài thời gian tương tác [9]. Mục tiêu này dẫn đến các bộ dữ liệu lớn hơn cung cấp tài liệu tốt hơn cho phân tích hành vi. Các kỹ thuật học máy đã được tích hợp vào chức năng của Honeypot cho phép chúng học hỏi từ các cuộc tấn công. Các phản hồi được trình bày cho những kẻ tấn công được tối ưu hóa để kéo dài thời gian tương tác. Các kỹ thuật học tập Có giám sát, Không giám sát và Tăng cường (RL) đã được triển khai để mô hình hóa hành vi hồi cứu [3] và cũng để chủ động tương tác với kẻ tấn công [6]. Trong giai đoạn lây nhiễm, các botnet có thể lan truyền và xâm phạm các máy chủ. Các quy trình tự động này tìm kiếm trên web các máy chủ dễ bị tổn thương để thỏa hiệp và sau đó báo cáo lại cho bộ phận chỉ huy và kiểm soát (CC). Trong giai đoạn tiếp theo, quản trị viên bot sẽ hướng dẫn CC giao tiếp với các máy chủ, hướng dẫn họ thực hiện các lệnh tự động tiếp theo. Honeypot nắm bắt tất cả hoạt động tự động này từ các nỗ lực vũ phu ban đầu, đến các thỏa hiệp và hướng dẫn tấn công tiếp theo.

Trong bài nghiên cứu này, chúng tôi sẽ mô tả cách tiếp cận RL để tạo ra một Honeypot thích ứng phù hợp để kéo dài thời gian tương tác với các bot tự động. Các cách tiếp cận trước đây như RASSH [6] cũng đã sử dụng RL để kéo dài thời gian tấn công. Tuy nhiên điều này phần lớn là dành cho những kẻ tấn công con người. Kết quả của chúng tôi cho thấy rằng sự gia tăng gần đây của các bot tự động khiến các kỹ thuật này trở nên không phù hợp và chúng tôi đề xuất một hình thức không gian hành động trạng thái mới được thiết kế để kéo

dài thời gian tương tác cho phần mềm độc hại tự động không gian hành động trạng thái mới được thiết kế để kéo dài thời gian tương tác cho phần mềm độc hại tự động, đồng thời so sánh hiệu suất của phương pháp của chúng tôi với RASSH và chứng minh bằng thực nghiệm cách không gian trạng thái đã sửa đổi của chúng tôi cải thiện tương tác bot.

1.2. Giới thiệu những nghiên cứu liên quan

1.2.1. Trình giăng bẫy

Trình giăng bẫy của nhóm chúng tôi hay còn gọi là Honeypot, hệ thống sẽ phát hiện và thu thập, học hỏi từ các cuộc tấn công, sau đó sử dụng những thông tin đó để nâng cao an toàn thông tin và nghiên cứu để khắc phục. Honeypot thường được xây dựng với các lỗ hổng bảo mật đã biết hoặc thiết lập dựa trên các thành phần mạng và ứng dụng mục tiêu tiềm năng. Đóng vai trò như một "bẫy", Honeypot tạo ra các cấu trúc dữ liệu và ghi lại các hoạt động lạ, để phân tích và tìm hiểu về các phương thức tấn công mới hoặc tiên tiến.

1.2.2. Mô hình học tăng cường

Học tăng cường là một kỹ thuật học máy trong đó một tác tử học tập học trực tiếp từ môi trường của nó, thông qua các tương tác thử và sai mà không có bất kỳ kiến thức nào trước đó. Thay vì được hướng dẫn về hành động cần thực hiện với một tập hợp đầu vào cụ thể, thay vào đó, nó học hỏi dựa trên kinh nghiệm trước đó về hành động mà nó nên thực hiện trong hoàn cảnh hiện tại.

1.3. Tính ứng dụng

Đề tài này đưa hệ thống giăng bẫy áp dụng học tăng cường vào ngữ cảnh thực tế hơn. Các mẫu xâm nhập sẽ qua một bước kiểm tra tính năng và đưa ra phản hồi. Vì vậy kết quả sau khi thực nghiệm sẽ cải thiện hiệu suất thu thập,

đôi khi có thêm nhiều đề xuất phòng tránh trong tương lai.

1.4. Những thách thức

Trong quá trình lập trình Honeypot để tái hiện cách thức hoạt động của hệ thống, chúng tôi phát hiện rằng hệ thống trong bài báo như RASSH và Kippo đều thực thi bằng Python 2, mặc dù đã cố gắng cải tiến Honeypot chạy trên nền Python 3 nhưng đã thất bại, vì thế nhóm quyết định giữ nguyên phiên bản cũ. Ngoài ra, tập dữ liệu thử nghiệm tương đối ít và cũng giới hạn các dạng câu lệnh, nên đôi khi sẽ khó phân tích các trường hợp phức tạp hơn.

1.5. Mục tiêu, đối tượng, và phạm vi nghiên cứu

1.5.1. Mục tiêu nghiên cứu

Ứng dụng học tăng cường để thiết lập, cải tiến Honeypot giúp cải thiện hiệu suất và mở rộng tầm thu thập các loại tấn công.

1.5.2. Đối tượng nghiên cứu

Đối tượng nghiên cứu:

- Các cuộc xâm nhập Shell thông qua kết nối SSH
- Trình giăng bẫy kết hợp học máy
- Mô hình học tăng cường
- Các phương pháp đánh giá tính hiệu quả.

1.5.3. Phạm vi nghiên cứu

Tìm hiểu cách thực thi của một số tập lệnh thường thấy trong các cuộc xâm nhập, dựa vào các đặc tính và hành vi của tập lệnh mà trả về các trạng thái

khác nhau, đồng thời kéo dài thời gian tương tác của kẻ tấn công.

1.5.4. Cấu trúc Đề tài

Chúng tôi xin trình bày nội dung của Đồ án theo cấu trúc như sau:

- Chương 1: Giới thiệu tổng quan về đề tài của Đồ án và những nghiên cứu liên quan.
- Chương 2: Trình bày cơ sở lý thuyết và kiến thức nền tảng liên quan đến đề tài.
- Chương 3: Trình bày mô hình hệ thống giăng bẫy.
- Chương 4: Trình bày thực nghiệm và đánh giá.
- Chương 5: Kết luận và hướng phát triển của đề tài.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này trình bày cơ sở lý thuyết của nghiên cứu: Bao gồm mã độc, các kỹ thuật phân tích mã độc, mô hình giăng bẫy, và mô hình học tăng cường.

2.1. Trình giăng bẫy

Trình giăng bẫy (hay còn gọi là Honeypot) là một máy chủ hoặc một hệ thống mạng giả mạo giống như một hệ thống máy tính với các ứng dụng và dữ liệu, được thiết kế để thu hút và ghi lại các hoạt động tấn công từ các tác nhân độc hại. Mục tiêu chính của Honeypot là thu thập thông tin về các phương thức tấn công, chiến thuật và các mẫu tấn công mới nhằm nâng cao hiểu biết và bảo mật của hệ thống thật.

Honeypot có thể mô phỏng các máy chủ, ứng dụng, hệ điều hành, dịch vụ mạng và các hệ thống mạng khác. Một Honeypot thường được cấu hình để tạo ra các mục tiêu hấp dẫn cho tấn công viên, như các cổng mạng mở, dịch vụ không được bảo vệ hoặc các lỗ hổng bảo mật tiềm ẩn.

Khi kẻ tấn công tìm cách xâm nhập vào trình giăng bẫy, các hoạt động và thông tin quan trọng về tấn công sẽ được ghi lại và phân tích. Các thông tin thu thập được từ Honeypot có thể bao gồm địa chỉ IP, phương thức tấn công, gói tin mạng, thông tin xác thực và các hành vi xâm nhập nhằm tìm hiểu về kỹ thuật tấn công và từ đó cải thiện bảo mật hệ thống thật.

Trình giăng bẫy rất hữu ích trong việc phát hiện tấn công, tìm hiểu các mẫu tấn công mới, xác định các lỗ hổng bảo mật và nghiên cứu để nâng cao hiểu biết về bảo mật mạng. Tuy nhiên, cần đảm bảo rằng Honeypot không gây nguy hiểm cho hệ thống thật và được quản lý an toàn để ngăn chặn tấn công từ Honeypot lan rộng vào hệ thống chính.

Các Honeypot có thể được phân loại dựa trên việc chúng là vật lý hay ảo:

- **Honeypot vật lý:** là máy thực có địa chỉ IP riêng, nó mô phỏng các hành vi được mô hình hóa bởi hệ thống. Tuy nhiên nó không được sử dụng nhiều do giá thành cao, việc cấu hình phần cứng chuyên dụng làm việc bảo trì trở nên phức tạp hơn
- **Honeypot ảo:** Honeypot này cho phép một người cài đặt và mô phỏng các máy chủ trên mạng từ các hệ điều hành khác nhau. Phương thức này được sử dụng nhiều hơn.

Các Honeypot có thể được phân loại dựa trên việc triển khai:

- **Honeypot sản xuất:** dễ sử dụng, chỉ nắm bắt thông tin giới hạn và được sử dụng chủ yếu bởi các tập đoàn. Honeypot sản xuất được đặt bên trong mạng sản xuất cùng với các máy chủ sản xuất khác để cải thiện trạng thái bảo mật chung. Thông thường, các Honeypot sản xuất là các Honeypot tương tác thấp, dễ triển khai hơn. Chúng cung cấp ít thông tin về các cuộc tấn công hoặc kẻ tấn công hơn so với Honeypot nghiên cứu.
- **Honeypot nghiên cứu:** được thực hiện để thu thập thông tin về động cơ và chiến thuật của cộng đồng "black hat" nhắm vào các mạng khác nhau. Chúng được sử dụng để nghiên cứu các mối đe dọa mà các tổ chức phải đối mặt và tìm hiểu cách bảo vệ tốt hơn trước những mối đe dọa đó. Honeypot nghiên cứu rất phức tạp để triển khai và bảo trì, nắm bắt nhiều thông tin và được sử dụng chủ yếu bởi các tổ chức nghiên cứu, quân đội hoặc chính phủ.

Các Honeypot có thể được phân loại dựa trên các tiêu chí thiết kế:

- **Honeypot thuần túy (Pure Honeypot):** bắt chước hệ thống sản xuất. Dữ liệu được tạo ra để bảo mật, cũng như thông tin người dùng “nhạy cảm”, có một số cảm biến (sensor) được sử dụng để theo dõi và quan sát hoạt động của kẻ tấn công.

- **Honeypot thích ứng (Adaptive Honeypot):** là các Honeypot được thiết kế để tự động thích ứng và điều chỉnh cấu hình của chúng để tạo ra một môi trường Honeypot linh hoạt và khó bị phát hiện đối với kẻ tấn công. Các Honeypot thích ứng có khả năng theo dõi và phân tích hành vi của kẻ tấn công để hiểu và tương tác với họ. Dựa trên thông tin thu thập được từ hành vi tấn công, Honeypot này có thể thay đổi các đặc điểm và cấu hình của mình như các dịch vụ mạng, cổng kết nối, lỗi phần mềm để đáp ứng các hành vi cụ thể của kẻ tấn công.

- **Honeypot tương tác cao (High-interaction Honeypot):** được thiết kế để khiến những kẻ tấn công đầu tư nhiều thời gian nhất có thể vào bên trong Honeypot. Mang lại cho nhóm bảo mật nhiều cơ hội hơn để quan sát các mục tiêu và ý định của kẻ tấn công cũng như nhiều cơ hội hơn để khám phá các lỗ hổng trong hệ thống. Honeypot tương tác cao có thể có thêm hệ thống, cơ sở dữ liệu và quy trình mà kẻ tấn công sẽ muốn cố gắng xâm nhập. Các nhà nghiên cứu có thể quan sát cách kẻ tấn công tìm kiếm thông tin, cũng như thông tin nào họ thích và cách họ cố gắng leo thang đặc quyền truy cập. Vd: Honeynet

- **Honeypot tương tác thấp (Low-interaction Honeypot):** chỉ mô phỏng các dịch vụ mà kẻ tấn công thường xuyên yêu cầu. Vì Honeypot tiêu thụ tương đối ít tài nguyên nên nhiều máy ảo có thể dễ dàng được lưu trữ trên một hệ thống vật lý, hệ thống ảo có thời gian phản hồi ngắn và cần ít mã hơn, giảm độ phức tạp trong bảo mật của hệ thống ảo. Vd: Honeyd

- **Sugarcane:** là một loại Honeypot giả dạng một proxy mở. Nó thường có thể ở dạng một máy chủ được thiết kế trông giống như một proxy HTTP bị định cấu hình sai.

2.2. Tổng quan mô hình học máy

Mô hình học máy là một khái niệm trong lĩnh vực trí tuệ nhân tạo (AI) và học máy (machine learning). Đây là một hệ thống hoặc mô hình tính toán được xây dựng để học từ dữ liệu và kinh nghiệm để tạo ra dự đoán hoặc thực hiện các nhiệm vụ mà không cần được lập trình cụ thể.

Mô hình học máy sử dụng các thuật toán và phương pháp để xây dựng một mô hình dự đoán hoặc học hành vi từ dữ liệu huấn luyện có sẵn. Mô hình có thể được sử dụng để phân loại dữ liệu, dự đoán kết quả, điều khiển hệ thống hoặc thực hiện các tác vụ khác dựa trên dữ liệu đầu vào.

Các mô hình học máy thường được huấn luyện bằng cách sử dụng một tập dữ liệu đào tạo, trong đó mô hình học và điều chỉnh các thông số của nó để tối ưu hóa kết quả. Các mô hình học máy phổ biến bao gồm học có giám sát (supervised learning), học không giám sát (unsupervised learning), học tăng cường (reinforcement learning) và chi tiết như sau:

- Học có giám sát (Supervised Learning): Trong học có giám sát, mô hình học từ các cặp dữ liệu huấn luyện gồm dữ liệu đầu vào và đầu ra tương ứng. Mục tiêu của mô hình là học một hàm ánh xạ từ dữ liệu đầu vào tới đầu ra tương ứng. Khi được huấn luyện, mô hình sẽ dự đoán đầu ra cho các dữ liệu mới. Ví dụ: phân loại hình ảnh, dự đoán giá nhà.

- Học không giám sát (Unsupervised Learning): Trong học không giám sát, mô hình được huấn luyện từ dữ liệu đầu vào mà không có nhãn hoặc đầu ra tương ứng. Mục tiêu của mô hình là tìm ra cấu trúc, mẫu hoặc nhóm trong dữ liệu đầu vào. Các phương pháp phổ biến trong học không giám sát bao gồm phân cụm (clustering), giảm chiều dữ liệu (dimensionality reduction), và khai phá luật kết hợp (association rule mining).

- Học tăng cường (Reinforcement Learning): Trong học tăng cường, mô hình học bằng cách tương tác với môi trường và nhận phản hồi (reward) từ môi

trường. Mục tiêu của mô hình là tìm ra chuỗi hành động sẽ tối đa hóa tổng phần thưởng trong thời gian dài. Mô hình học tăng cường được sử dụng trong các bài toán như tự hành, chơi game, và quyết định tối ưu trong môi trường động.

Ngoài ba loại chính này, còn có các phương pháp và kỹ thuật khác trong lĩnh vực học máy như học bán giám sát (semi-supervised learning), học sâu (deep learning), và học truyền thống (traditional machine learning). Tùy thuộc vào bài toán cụ thể, các phương pháp và kỹ thuật này có thể được sử dụng để giải quyết các vấn đề khác nhau. Mô hình học máy có thể áp dụng vào nhiều lĩnh vực, bao gồm nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự đoán tín dụng, quảng cáo trực tuyến, phân tích dữ liệu và nhiều lĩnh vực khác.

2.3. Tổng quan mô hình học tăng cường

Học tăng cường là một nhánh nghiên cứu của học máy, trong đó sẽ có một tác tử đưa ra các hành động để tương tác với một môi trường. Mục tiêu cuối là tối ưu hóa phần thưởng tích lũy đạt được.

Dựa trên MDP, một hệ thống học tăng cường cơ bản sẽ có hai thành phần chính là Tác tử (Agent) và Môi trường (Environment). Kênh giao tiếp của hai thành phần này bao gồm: Hành động (Action), Phần thưởng (Reward), và Trạng thái (State).

Tác tử sẽ tương tác với môi trường qua mội chuỗi các lượt đi. Với mỗi lượt t, tác tử sẽ chọn một hành động $a_t \in A$, dựa vào một chính sách $\pi(a|s_t)$ và thu về một vectơ trạng thái có thể quan sát được s_t từ môi trường. Môi trường cũng sẽ xuất ra một phần thưởng $r_t \in R$ tương ứng với các hành động nhận vào, đồng thời môi trường cũng thay đổi trạng thái của chính nó, tạo ra một trạng thái mới s_{t+1}. Phần thưởng r_t và trạng thái s_{t+1} sẽ là đầu vào kế tiếp của tác tử. Từ đó, tác tử sẽ xây dựng tiếp chính sách $\pi(a|s_{t+1})$. Tác tử sẽ học dần và tuần tự qua việc khám phá (exploration) và khai thác (exploitation), dần dần chọn ra

Hình 2.1: *Mô hình học tăng cường*

được hành động nhằm tạo ra trạng thái môi trường mong muốn.

Phần thưởng là yếu tố quan trọng trong việc học và mục tiêu cuối cùng của mô hình học tăng cường là xây dựng được một chính sách để tối đa hóa kì vọng:

$$V^{\pi}(s_t) = E_{a_t}[Q^{\pi}(s_t, a_t)|s_t]$$

với

$$Q^{\pi}(s_t, a_t) = E_{s_{t+1:\infty}, a_{t+1:\infty}}[R_t|s_t, a_t]$$

và

$$R_t = \sum_{i \geq 1} \gamma^i r_t + i, \gamma \in [0, 1]$$

γ là giá tri chiết khấu cho phần thưởng. Giá trị γ kiểm soát hành động mới sao cho chúng không được phép tác động nhiều đến phần thưởng nhưng lại quan trọng đối với kết quả V^{π}. Hàm đánh giá mức độ hữu ích của một hành động cho một trạng thái được gọi là hàm Q.

2.4. Quy trình quyết định Markov (Markov Decision Process - MDP)

Các vấn đề về RL nói chung có thể được mô hình hóa bằng các quy trình quyết định Markov (MDPs). Trên thực tế, các phương pháp RL tạo điều kiện thuận lợi cho các giải pháp cho MDP khi không có mô hình môi trường hoàn chỉnh. Điều này đặc biệt hữu ích khi giải quyết các vấn đề trong thế giới thực, chẳng hạn như Honeypot, vì mô hình thường có thể không xác định hoặc khó ước lượng. MDP là một khung toán học cụ thể phù hợp để lập mô hình ra quyết định trong điều kiện không chắc chắn.

- S: Không gian các trạng thái mà tác tử quan sát được khi tương tác với môi trường.
- A: Một tập các hành động mà tác tử có thể thực hiện với môi trường.
- $p(.|s, a)$: Xác định phân phối xác suất chi phối chuyển đổi trạng thái
- $s_{t+1} \sim p(.|s, a)$;
- $q(.|s, a)$: Xác định phân phối xác suất chi phối phần thưởng nhận được
- $R(s_t, a_t) \sim q(.|s_t, a_t)$: Một hàm đưa ra phần thưởng. Hàm này sẽ xác định phần thưởng khi môi trường chuyển từ một trạng thái sang trạng thái kế tiếp, dưới tác động của một hành động.

S là tập hợp tất cả các trạng thái có thể đại diện cho thế giới quan sát được của tác tử. Vào cuối mỗi khoảng thời gian, t là tác tử chiếm trạng thái $s_t \in S$. Sau đó, tác tử phải chọn một hành động tại $a_t \in A_{(s_t)}$, trong đó $A_{(s_t)}$ là tập hợp tất cả các hành động có thể xảy ra trong trạng thái s_t. Việc thực hiện hành động đã chọn dẫn đến chuyển trạng thái sang s_{t+1} và phần thưởng bằng số ngay lập tức $R(s_t, a_t)$. Phương trình sau đây biểu thị hàm phần thưởng, xác định phân

phối phần thưởng theo môi trường:

$$R_a s, s' = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\}(1)$$

Mục tiêu của tác tử học tập là tối ưu hóa phần thưởng chiết khấu dài hạn dự kiến. Xác suất chuyển trạng thái $p(s_{t+1}|s_t, a_t)$ chi phối khả năng tác tử sẽ chuyển sang trạng thái s_{t+1} do chọn a_t trong s_t:

$$P_a s, s' = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}(2)$$

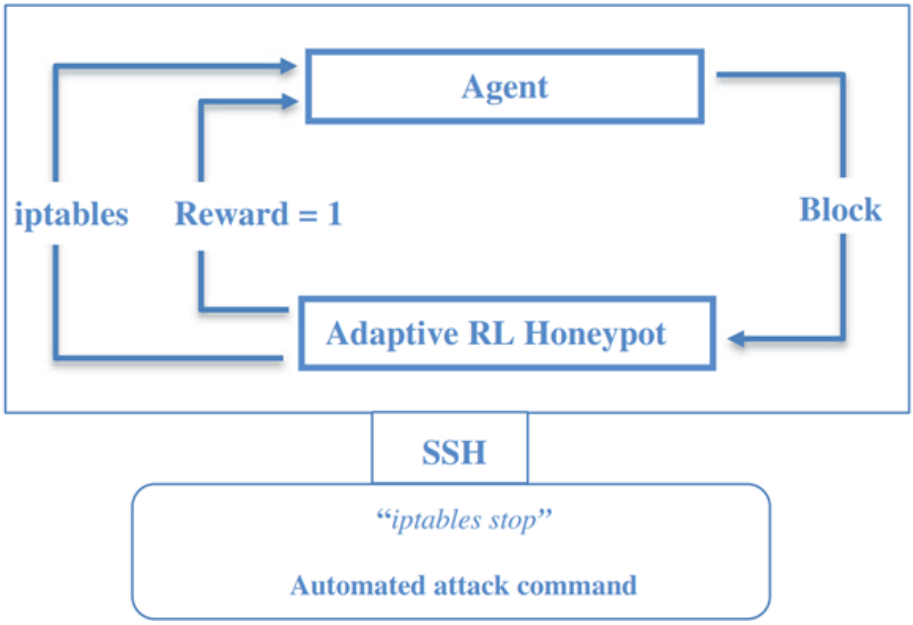
Phần thưởng bằng số nhận được khi đến trạng thái tiếp theo được điều chỉnh bởi phân phối xác suất $q(s_{t+1}|s_t, a_t)$ và cho thấy lợi ích của việc lựa chọn a_t trong khi ở s_t. Trong trường hợp cụ thể, khi đã biết mô hình môi trường hoàn chỉnh, tức là (S, A, p, q) hoàn toàn có thể quan sát được, bài toán giảm xuống thành bài toán lập kế hoạch và có thể được giải bằng các kỹ thuật lập trình động truyền thống như phép lặp giá trị. Tuy nhiên, nếu không có sẵn mô hình hoàn chỉnh, thì các phương pháp RL đã được chứng minh hiệu quả trong việc giải quyết MDP.

2.5. SARSA Learning

Trong quá trình RL, tác tử chọn một hành động bằng cách sử dụng chính sách (π) được cung cấp tín hiệu trạng thái tinh thần môi trường. Mục tiêu của tác tử RL là tìm hiểu chính sách tối ưu (π^*), ánh xạ tối ưu từ trạng thái sang hành động.

Chúng tôi phải xem xét cách áp dụng mô hình RL cho sự phát triển của Honeypot thích ứng. Trong suốt quá trình triển khai, Honeypot được coi là một môi trường với RL được tích hợp vào các thiết kế của nó. Chúng tôi đang sử dụng SSH làm điểm truy cập và máy chủ Linux mô phỏng làm môi trường dễ bị tấn công. Trong môi trường này, máy chủ có các trạng thái được kiểm tra và thay đổi bằng các tập lệnh bash. Ví dụ là iptables, wget, sudo, v.v. tác tử RL có

thể thực hiện các hành động trên các trạng thái này, chẳng hạn như cho phép hoặc chặn việc thực thi tập lệnh. Môi trường đưa ra phần thưởng cho tác tử để thực hiện hành động đó. tác tử học hỏi từ quá trình này và theo thời gian, học chính sách tối ưu π^*, biểu thị hành động tốt nhất để thực hiện cho một trạng thái nhất định. Một ví dụ đơn giản về quy trình Honeypot thích ứng được hiển thị trong Hình 2.2.



Hình 2.2: *Honeypot thích ứng với quá trình học tăng cường*

Chính sách tối ưu π^* được học theo thời gian khi Honeypot liên tục bị tấn công. Quá trình học tập cuối cùng sẽ hội tụ khi Honeypot được thưởng cho mỗi đợt tấn công. Phương pháp chênh lệch thời gian này để học theo chính sách sử dụng quá trình chuyển đổi từ một cặp trạng thái/hành động sang cặp trạng thái/hành động tiếp theo, để tìm hiểu các động lực của môi trường. Phần thưởng đạt được khi chuyển đổi được sử dụng để xác định giá trị của hành động trong khi ở trạng thái s. Trạng thái, Hành động, Phần thưởng, Trạng thái, Hành động còn được gọi là SARSA, là cách triển khai phổ biến của RL về chính sách (3). Chính sách này là sự kết hợp của các giá trị Q và chiến lược lựa chọn hành động. Chiến lược lựa chọn hành động mà chúng tôi sử dụng trong công việc

này là Epsilon-Greedy, thường chọn hành động mang lại phần thưởng lớn nhất tiết kiệm được một phần nhỏ thời gian khi nó chọn một hành động ngẫu nhiên. Chọn một hành động không tham lam được coi là một động thái thăm dò và người ta hy vọng rằng bằng cách chọn một hành động dưới mức tối ưu trong một thời gian, các chính sách tốt hơn có thể được tìm thấy trong thời gian dài. Phương trình sau mô tả quy tắc cập nhật cho SARSA

$$Q_(s_t, a_t) \leftarrow Q_(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)](3)$$

Tham số tốc độ học cũng được áp dụng (α) kiểm soát lượng kiến thức mới mà chúng tôi cập nhật cho mỗi lần thử.

2.6. Honeypot thích ứng

Các kỹ thuật học máy trước đây đã được áp dụng cho Honeypot. Những kỹ thuật này đã phân tích hồi cứu hành vi của kẻ tấn công trên tập dữ liệu đã thu thập. Các phương pháp giám sát và không giám sát được sử dụng để lập mô hình thích ứng với phần mềm độc hại và phân loại các cuộc tấn công [5][2]. Phân tích này là một nguồn thông tin có giá trị cho các nhóm bảo mật. Tuy nhiên, bằng cách chủ động thích ứng với kẻ tấn công, một Honeypot có thể kéo dài thời gian thích ứng và nắm bắt các bộ dữ liệu lớn hơn có khả năng dẫn đến phân tích tốt hơn. Để đạt được mục tiêu này, việc tạo ra các Honeypot thông minh, có khả năng thích ứng đã được khám phá. Wagener [9] sử dụng RL để trích xuất càng nhiều thông tin càng tốt từ kẻ xâm nhập, để kéo dài tương tác. Honeypot có tên là Heliza được phát triển để sử dụng RL khi giao chiến với kẻ tấn công. Honeypot đã thực hiện các chiến lược hành vi như chặn lệnh, trả lại thông báo lỗi và đưa ra lời lăng mạ. Trước đây, anh đã đề xuất sử dụng lý thuyết trò chơi [10] để xác định hành động phản ứng của một Honeypot đối với hành vi của kẻ tấn công. Một máy tự động xác suất phân cấp được trình bày với mục đích làm cho một Honeypot thích nghi và tự chủ. Lấy mô hình RL, Heliza

định nghĩa môi trường là các tương tác giữa thỏa hiệp SSH thành công và kẻ tấn công thoát khỏi Honeypot. Không gian trạng thái hữu hạn cơ bản là danh sách các lệnh được nhập trong quá trình tấn công vào môi trường. Các hành động của Honeypot như sau: allow, block, substitute và insult. Chức năng phần thưởng có hai mục tiêu. Đầu tiên là thu thập thông tin liên quan đến kẻ tấn công, bao gồm các chuyển đổi sang công cụ tấn công, công cụ hệ thống và các lệnh xúc phạm. Thứ hai là kéo dài thời lượng tương tác của kẻ tấn công được biểu thị bằng độ trễ giữa các lệnh của kẻ tấn công gợi ý chức năng nhận thức của con người. Wagener gửi trước kết quả cho hai chức năng phần thưởng. Đối với phần thưởng đầu tiên, anh ấy chọn hai trạng thái sudo và wget làm ví dụ về trạng thái và hiển thị các giá trị học tập được tính cho mỗi hành động. Phần thưởng thứ hai so sánh số lần chuyển đổi tích lũy được ghi lại trên Honeypot, với một Honeypot tương tác cao tiêu chuẩn. Honeypot được phát triển bằng cách sử dụng khung Chế độ người dùng Linux (UML).

Pauna [6] cũng trình bày một Honeypot thích ứng sử dụng các thuật toán và tham số học tăng cường tương tự như Heliza. Anh ấy đề xuất một Honeypot có tên RASSH cung cấp các cải tiến về khả năng mở rộng, bản địa hóa và khả năng học tập. Nó thực hiện điều này bằng cách sử dụng các thư viện mới hơn và một Honeypot thích ứng cao gọi là Kippo [8]. RASSH so sánh chức năng của nó với Heliza thể hiện hành vi tương tự. Cả Heliza và RASSH đều sử dụng PyBrain để triển khai RL và cả hai đều sử dụng các hành động hướng đến tương tác của con người.

Vì mỗi cuộc tấn công được coi là một giai đoạn, chính sách học tập được đánh giá vào cuối mỗi giai đoạn này. Phương pháp học theo chính sách này được triển khai trên cả hai Honeypot sử dụng SARSA (3) với các tham số sau:

- ϵ: Chính sách Greedy
- Môi trường Honeypot không được biết đối với tác tử học tập. Chúng tôi muốn nó học hỏi và cuối cùng hội tụ. Để đạt được điều này, chúng tôi đặt

trình khám phá của mình thành ϵ-greedy

- γ: Xác định phân phối xác suất chi phối chuyển đổi trạng thái
- γ được áp dụng khi một phần thưởng trong tương lai được ước tính. Đối với Honeypot của chúng tôi, không có yếu tố giảm giá nào được áp dụng, vì các đợt tấn công là các lệnh được xác định dễ dàng giữa các sự kiện mở và đóng SSH. Điều này cho phép tính toán phần thưởng hồi tố.
- $\alpha = 0.05$
- PyBrain cũng cho phép thiết lập các tham số để tạo toàn bộ không gian trạng thái/hành động. Điều này có liên quan đến việc tạo ra các Honeypot, cho các nhiệm vụ trong phần tiếp theo

CHƯƠNG 3. MÔ HÌNH HONEYPOT

Ở chương này chúng tôi sẽ trình bày mô hình Honeypot và cách triển khai.

3.1. Mục tiêu

Ở mục này, chúng tôi sẽ cải thiện chức năng của Honeypot bằng cách tăng tương tác của kẻ tấn công. Phương pháp của chúng tôi sửa đổi các triển khai trước đó với các tham số RL hiệu quả hơn đối với phần mềm độc hại tự động. Chức năng phần thưởng của chúng tôi được đơn giản hóa để tạo ra một hình thức không gian hành động trạng thái mới. Điều này được triển khai trên Honeypot thích ứng của chúng tôi và ba nhiệm vụ sau đây được thực hiện cho đề xuất này:

- So sánh trực tiếp với nghiên cứu trước đây bằng cách sử dụng tập hợp hành động rút gọn trong môi trường mô phỏng.
- Triển khai hai Honeypot trực tiếp với một bộ hành động giảm nhắm mục tiêu phần mềm độc hại tự động đã biết.
- So sánh số lần chuyển đổi lệnh trên cả 2 Honeypot với bộ hành động giảm với nghiên cứu trước đây.

3.2. Cách thức tấn công

Dựa vào kiến thức cũng như quá trình vận hành các hệ thống Honeypot trước kia, chúng tôi tổng hợp khả năng của kẻ tấn công được dựa theo ba khía cạnh:

1. **Mục đích của kẻ tấn công**: Kẻ tấn công có thể theo đuổi ba hướng (theo mô hình Tính bảo mật - Tính toàn vẹn - Tính sẵn sàng) như sau:

- Tính bảo mật (confidentiality): Kẻ tấn công muốn lấy thông tin của mô hình phát hiện mã độc.
- Tính toàn vẹn (integrity): Kẻ tấn công có thể kiến cho mô hình đưa ra các kết quả sai lệch.
- Tính sẵn sàng (availability): Kẻ tấn công có thể làm cho mô hình không hoạt động được hoặc làm gián đoạn các chức năng của mô hình.

2. **Kiến thức của kẻ tấn công:** Các phương pháp tấn công sẽ thay đổi tùy vào lượng kiến thức của kẻ tấn công đối nhiều mục đích khác nhau. Ở đây có một số phương thức tiêu biểu:

   - **Quét cổng (Port Scanning):** Tấn công viên thường quét các cổng mạng của Honeypot để tìm các dịch vụ mạng đang chạy. Bằng cách này, họ có thể tìm ra các lỗ hổng bảo mật tiềm năng và xác định các điểm yếu của hệ thống.
   - **Tấn công từ chối dịch vụ (Denial-of-Service, DoS):** Tấn công DoS có thể được thực hiện bằng cách gửi nhiều yêu cầu không hợp lệ hoặc quá tải hệ thống Honeypot, dẫn đến việc làm cho dịch vụ hoặc hệ thống trở nên không khả dụng.
   - **Tấn công từ chối dịch vụ phân tán (Distributed Denial-of-Service, DDoS):** Tấn công DDoS gửi các yêu cầu đến Honeypot từ hàng ngàn hoặc hàng triệu thiết bị khác nhau, nhằm làm quá tải hệ thống mục tiêu. Điều này gây ra sự cản trở hoặc làm cho hệ thống Honeypot không khả dụng.
   - **Tấn công thâm nhập (Intrusion):** Tấn công viên có thể cố gắng xâm nhập vào Honeypot bằng cách sử dụng các phương pháp như tấn công mật khẩu, khai thác lỗ hổng bảo mật, hoặc tìm cách đánh lừa Honeypot để có quyền truy cập vào hệ thống.
   - **Khai thác lỗ hổng (Exploitation):** Tấn công viên có thể tìm cách khai thác các lỗ hổng bảo mật trên Honeypot. Họ sẽ tìm cách tiến hành

các cuộc tấn công thành công để kiểm tra tính ổn định và an ninh của Honeypot.

- **Thu thập thông tin (Information Gathering):** Tấn công viên có thể cố gắng thu thập thông tin về Honeypot, như địa chỉ IP, hệ điều hành, cổng mạng mở, hoặc các thông tin khác để xác định mục tiêu tiếp theo hoặc lập kế hoạch tấn công.

3. **Rủi ro:** Nếu kẻ tấn vượt qua Honeypot và truy cập vào hệ thống thật, có thể có một số yếu tố và cách thức tấn công được sử dụng. Dưới đây là một số ví dụ:

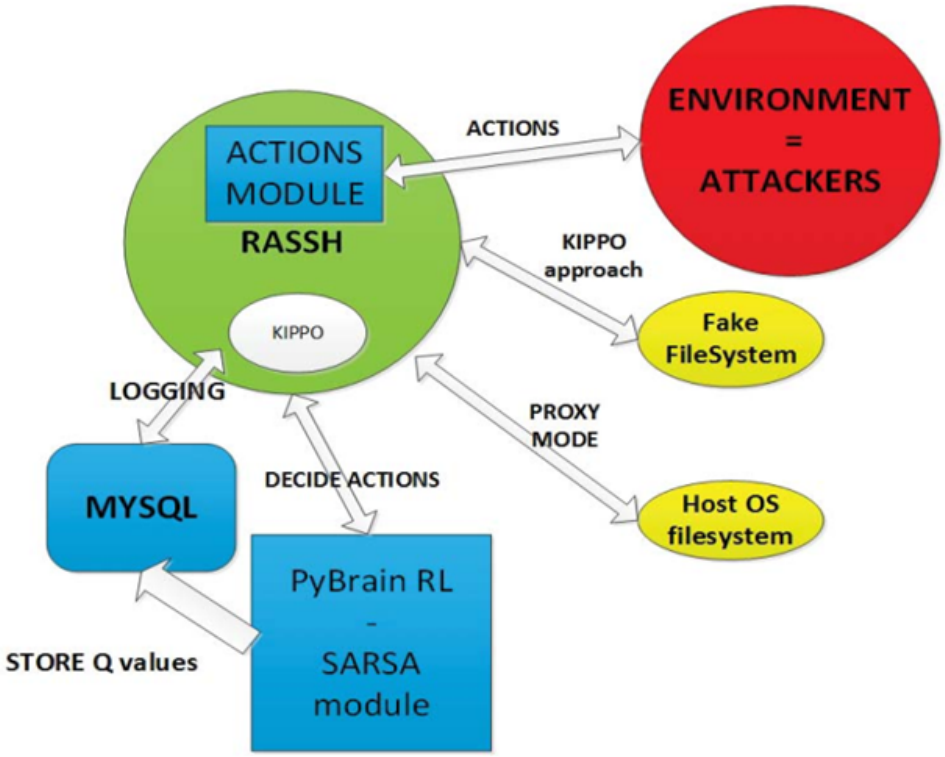
   - **Lợi dụng lỗ hổng mạng:** Tấn công viên có thể tìm thấy các lỗ hổng mạng trong hệ thống thật và sử dụng chúng để tiến hành tấn công. Điều này có thể là do các cổng mạng không an toàn, các dịch vụ chưa được bảo vệ đầy đủ hoặc quản lý sai sót cấu hình mạng.
   - **Tấn công qua các lỗ hổng ứng dụng:** Tấn công viên có thể khai thác các lỗ hổng trong các ứng dụng hoặc phần mềm mà hệ thống thật sử dụng. Điều này có thể bao gồm việc sử dụng mã độc, lỗi bảo mật hoặc các lỗ hổng khác để xâm nhập vào hệ thống.
   - **Sử dụng các phương pháp xã hội (Social Engineering):** Tấn công viên có thể sử dụng kỹ thuật xã hội để lừa đảo hay phục dựng các tin nhắn, email, trang web giả mạo hoặc tấn công quản trị viên hệ thống để lấy thông tin đăng nhập, mật khẩu hoặc các thông tin nhạy cảm khác.
   - **Sử dụng tài nguyên bên ngoài:** Tấn công viên cũng có thể sử dụng các tài nguyên bên ngoài để tấn công, ví dụ như sử dụng các máy tính từ xa (remote computer) hoặc các máy tính thuộc botnet (một mạng lưới các máy tính bị chiếm đoạt) để thực hiện các cuộc tấn công từ xa.

3.3. Mô hình giăng bẫy

Trong chương này chúng tôi sẽ trình bày tổng quan về mô hình giăng bẫy - Honeypot. Mô hình này dựa vào các đặc điểm tấn công được giải thích chi tiết ở chương Ba.

3.3.1. Tổng quan mô hình đề xuất

Như đã nêu trước đây, RASSH là một sự phát triển của Kippo đạt được bằng cách triển khai trong Python các mô-đun cần thiết thúc đẩy việc học tăng cường. Kiến trúc của hệ thống, được trình bày trong Hình 3.1, bao gồm một số mô-đun với các vai trò cụ thể: mô-đun Honeypot, mô-đun Actions, mô-đun RL.



Hình 3.1: *Kiến trúc RASSH*

Việc triển khai mô-đun này dựa trên mã hiện có của Kippo nhưng cải thiện khả năng của nó thông qua việc triển khai một số lệnh mới (useradd, userdel, v.v.). Bên cạnh các lệnh mô phỏng đã tồn tại (ifconfig, wget, ping, v.v.), chúng

tôi đã triển khai các lệnh mới sử dụng một cách tiếp cận khác mà chúng tôi gọi là chế độ proxy. Ở chế độ proxy, mô-đun Honeypot hoạt động như một proxy giữa lớp hệ điều hành Linux và lớp giả lập mà những kẻ tấn công truy cập. Bằng cách này, trước khi thực hiện lệnh ở cấp hệ điều hành, mô-đun sẽ chờ các hành động do mô-đun RL quyết định.

Chúng tôi có tham khảo các mô hình cải tiến khác như RL-Kippo (RLH-Pot[7]). Qua các nghiên cứu liên quan này, chúng tôi đã lập trình các Honeypot bằng RASSH và RL-Kippo, trong đó RASSH đã triển khai một phiên bản cải tiến của Heliza bằng Kippo và quan sát hành vi tương tự với tập dữ liệu. Kippo không còn được phát triển tích cực nữa nhưng đã được mở rộng và phát hành dưới tên Cowrie [1] nhưng chúng tôi sẽ không áp dụng trong thực nghiệm. Cả hai Honeypot trên đều được cài đặt trong môi trường phát triển Eclipse. Thuật toán RL được triển khai bằng PyBrain, tương tự như RASSH và cho phép so sánh chặt chẽ giữa hai thuật toán này. Cả hai Honeypot đều hoạt động tương tự với cùng một tập dữ liệu tấn công. Quy tắc cập nhật SARSA (3) tính toán các giá trị Q bằng cách kết hợp phần thưởng đạt được từ việc chọn hành động a, giá trị hiện tại của cặp hành động trạng thái và giá trị chiết khấu của cặp hành động trạng thái tiếp theo. Áp dụng hệ số chiết khấu theo cách này cho phép đại lý chiết khấu giá trị của phần thưởng trong tương lai, vì chúng có thể không có giá trị đó khi bạn đến đó. Lệnh tấn công hiện tại được phân tích thành trạng thái hiện tại và được chuyển đến mô-đun PyBrain SARSA. PyBrain chọn hành động sẽ trả về giá trị tối đa cho một trạng thái nhất định. Các giá trị Q được trả lại cho Honeypot để lưu trữ trong bảng tra cứu. Honeypot tiếp tục bằng cách tương tác với lệnh tấn công tùy thuộc vào hành động được chọn bởi tác tử học tập SARSA. Honeypot thích ứng có các yếu tố sau:

- Sửa đổi Honeypot. Cowrie là một bản phân phối Honeypot được sử dụng rộng rãi, ghi lại tất cả các tương tác SSH trong cơ sở dữ liệu MySQL. Điều này đã được sửa đổi để tạo ra các tham số để chuyển đến tác tử học tập. Tùy thuộc vào hành động được chọn bởi tác tử học tập, Honeypot sẽ cho

phép, chặn hoặc thay thế các lệnh tấn công

- Tác tử SARSA. Mô-đun này nhận các tham số cần thiết từ Honeypot thích ứng và tính toán $Q_(s_t, a_t)$ theo công thức (3). Nó xác định các phản hồi được chọn bởi Honeypot thích ứng và tìm hiểu theo thời gian những hành động nào mang lại phần thưởng lớn nhất.

3.3.2. Action Model

Mô-đun này đóng vai trò trung gian giữa mô-đun Honeypot và mô-đun RL. Mục đích của nó là thực hiện các hành động trong trình bao mà mô-đun Honeypot cung cấp cho những kẻ tấn công. Quyết định hành động nào sẽ được kích hoạt và khi nào, xuất phát từ mô-đun RL. Mô-đun này độc lập với phương pháp thực hiện các lệnh (mô phỏng hoặc proxy). Cách tiếp cận sau đây đã được sử dụng cho từng hành động:

- **Allow:** Là mô-đun cho phép thực hiện lệnh.
- **BLock:** Nếu mô-đun RL quyết định kích hoạt hành động này, đối với mỗi lệnh sẽ có một thông báo lỗi được lưu trong cơ sở dữ liệu được in trên trình bao và lệnh không được thực thi.
- **Fake output/Substitute:** nếu mô-đun RL quyết định kích hoạt hành động này, đối với mỗi lệnh sẽ có một thông báo đầu ra giả được lưu trữ trong cơ sở dữ liệu sẽ được in trên trình bao và lệnh không được thực thi. Đầu ra được lưu trữ là một bản sao đã sửa đổi của một bản bình thường và đối với các loại lệnh khác nhau, nó được liệt kê theo từng dòng (ví dụ: đầu ra lệnh Ping).
- **Delay:** Đây cũng là một cách triển khai đơn giản, nghĩa là nếu hành động này được kích hoạt, mã dòng ngủ với một khoảng thời gian cụ thể sẽ nằm trong vòng lặp và sau khi hết khoảng thời gian đó, lệnh được thực hiện.

- **Insult:** Nếu mô-đun RL quyết định kích hoạt hành động này, mô-đun sẽ định vị địa lý địa chỉ IP và thông báo xúc phạm được lưu trữ trong cơ sở dữ liệu bằng ngôn ngữ bản địa sẽ được in trên trình bao. Lệnh sẽ không được thực thi.

3.3.3. Hàm phần thưởng

Như đã nêu, việc kết hợp RL vào một Honeypot để cho phép nó thích ứng với các phương pháp tấn công trong thời gian thực. Những đóng góp trước đây liên quan đến khái niệm này đã sử dụng hành động đặt trong mô hình RL cung cấp tương tác của con người. Các hành động như xúc phạm kẻ tấn công và trì hoãn phản hồi đã được đề xuất và triển khai thành các Honeypot nguyên mẫu. Hai chức năng phần thưởng tương ứng với hai mục tiêu của các Honeypot đã được triển khai:

- Thu thập các lệnh liên quan đến kẻ tấn công và thông tin chuyển tiếp (r_t)
- Kéo dài thời gian tương tác của kẻ tấn công (r_d)

Chúng tôi chỉ ra rằng lưu lượng tấn công là tự động. Việc sử dụng các hành động như insult và delay một cách giả tạo góp phần kéo dài thời gian của một đợt tấn công và không cung cấp thêm thông tin chi tiết về âm mưu của phần mềm độc hại tự động. Do đó, không yêu cầu thực hiện khen thưởng lần thứ hai (r_d). Việc triển khai của chúng tôi theo đuổi mục tiêu tăng số lượng lệnh của kẻ tấn công được nhập vào Honeypot (r_t) của chúng tôi. Chúng tôi đề xuất rằng các Honeypot thích ứng được triển khai trên vectơ tấn công SSH nên triển khai RL với một bộ hành động giảm nhằm vào phần mềm độc hại tự động. Do chúng tôi đã loại bỏ insult như một phản ứng thực tế đối với các cuộc tấn công phần mềm độc hại tự động, nên chúng tôi cần sửa đổi chức năng phần thưởng cho việc triển khai RL trên Honeypot. RASSH tự so sánh mình với Heliza bằng cách lập biểu đồ quá trình học tập để thu thập thông tin của kẻ tấn công (các

chuyển đổi), trên trạng thái wget. Công thức được trình bày như sau:

$$r_t(s_i, a_j) = \begin{cases} 1 & \text{nếu } i \in C \\ \min\limits_{x \in Y}(I_d(i, x)) & \text{nếu } i \in I \\ 0 & \text{cho trường hợp khác} \end{cases}$$

Trong đó Y ∈ C ∪ L

Công thức cho phần thưởng chuyển tiếp r_t, dựa trên state/action (s_i, a_j) như sau:

- Nếu chuỗi đầu vào i là lệnh C của kẻ tấn công tùy chỉnh, thì phần thưởng $r_t(s_i, a_j) = 1$
- Nếu chuỗi đầu vào i là lệnh bash Linux L, thì thưởng $r_t(s_i, a_j) = 0$
- Nếu chuỗi đầu vào i được xác định là một chuỗi xúc phạm I, thì giá trị chuẩn hóa tối thiểu Levenstein [4] khoảng cách l_d so với Y hoặc ENTER được tính toán.

Tập hợp I được coi là điểm khác ngoài Y và ENTER. Phần thưởng cho điều này có được bằng cách tính toán khoảng cách Levenstein giữa lệnh tấn công và Y, ENTER. Như đã nêu, chúng tôi đề xuất rằng insult không phải là một mục hành động có liên quan cho các cuộc tấn công tự động. Chúng tôi đề xuất giảm bộ hành động thành Allow, Block, Substitute. Allow và Block là những phản hồi thực tế đối với hành vi của phần mềm độc hại. Botnet có thể sử dụng các cấu trúc if-else phức tạp để xác định các bước tiếp theo trong quá trình thỏa hiệp. Sử dụng Substitute để trả lại phản hồi thay thế cho lệnh tấn công có khả năng làm tăng số lần chuyển đổi tấn công sang các lệnh mới hơn. Tập hợp hành động giảm kết hợp với tập hợp trạng thái Y, tạo ra một không gian trạng thái/hành động rời rạc. Chức năng phần thưởng được đơn giản hóa sẽ giúp Honeypot tìm

hiểu và hội tụ nhanh hơn. Công thức sửa đổi cho phần thưởng như sau:

$$r_t(s_i, a_j) = \begin{cases} 1 & \text{nếu } i \in C \\ 0 & \text{cho trường hợp khác} \end{cases}$$

Trong đó Y $\in$ C $\cup$ L

CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ

Ở chương này chúng tôi tiến tạo môi trường, cài đặt và đưa ra các tiêu chí đánh giá về mức độ hiệu quả của mô hình.

4.1. Thiết lập thí nghiệm

Hệ thống giăng bẫy của chúng tôi được thực hiện trên môi trường hệ điều hành Ubuntu 18.04 với cấu hình RAM là 8GB và dung lượng 20GB. Ngoài ra, ngôn ngữ chính được sử dụng để xây dựng hệ thống là Python 2.7.

4.1.1. Tập dữ liệu

4.1.1.1. Lệnh Linux

Lệnh Linux là các câu lệnh được gõ trong cửa sổ terminal hoặc command line để thực hiện các tác vụ trong hệ điều hành Linux. Các lệnh Linux có thể thực hiện các chức năng như di chuyển, sao chép, xóa tập tin và thư mục, thao tác với quyền truy cập và phân quyền, cài đặt phần mềm, quản lý dịch vụ, kết nối và kiểm tra mạng, và nhiều tác vụ khác. Cấu trúc của một lệnh Linux bao gồm tên lệnh và các tham số hoặc tùy chọn liên quan đến tác vụ cần thực hiện.

Tập lệnh ở bảng 4.1 là trích xuất 1 phần các lệnh sẽ được thực thi trên các Honeypot. Kippo lưu trữ tất cả các tương tác trong tệp nhật ký theo tiêu chuẩn. Nó cũng lưu trữ tất cả các tệp đã tải xuống trong một thư mục tải xuống. Điều này giúp có thể tích lũy số lượng tất cả các lệnh đã thử trên Honeypot. Tất cả các lệnh này đại diện cho các tương tác hậu thỏa hiệp. Do đó, các sự kiện như lần thử không thành công, tấn công từ điển và bruteforce đều bị loại trừ vì chúng thể hiện các tương tác trước khi thỏa hiệp.

Bảng 4.1: *Tập lệnh thực nghiệm*

STT	Lệnh thực thi
1	/gweerwe323f
2	sudo /bin/sh
3	/bin/busybox cp
4	mount ;/gweerwe323f
5	echo -e '47726f70/' > //.nippon
...	...
65	chmod 777 usb$_b$us
66	./usb$_b$us
67	/gweerwe323f

4.1.2. Trình giăng bẫy

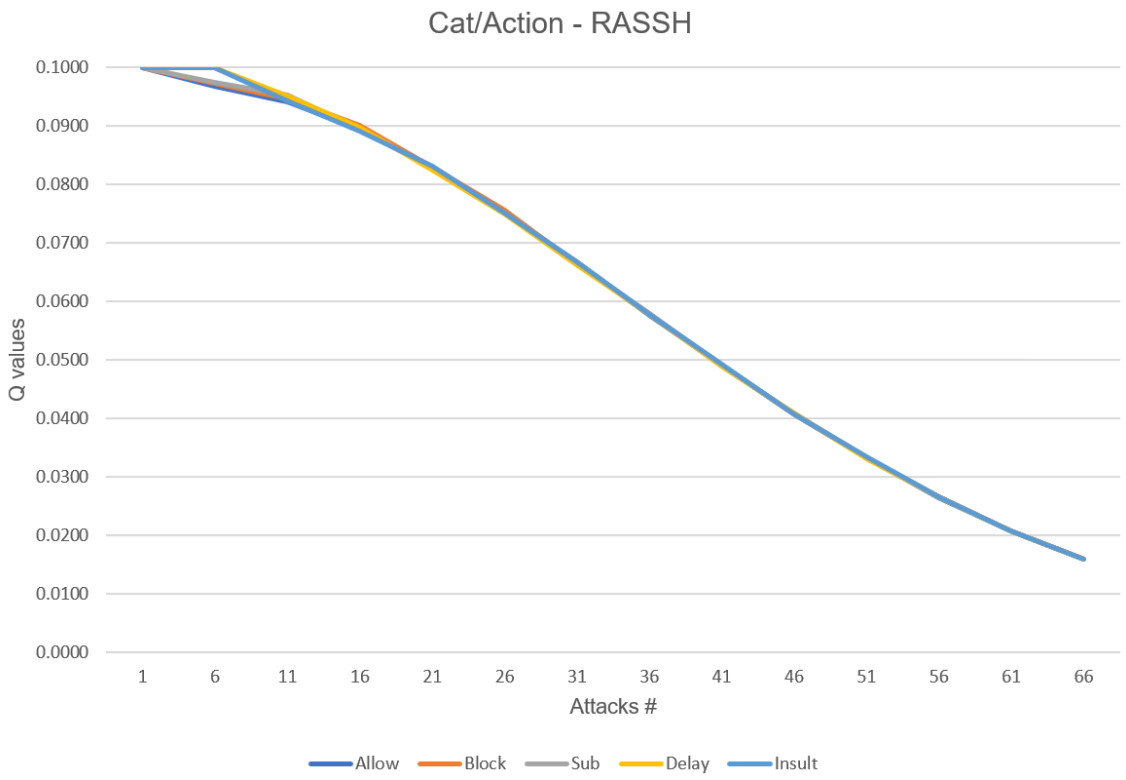
Như đã trình bày trong Chương 3, chúng tôi sử dụng hai Honeypot kết hợp học tăng cường là RASSH, RL-Kippo (RLHPot) cùng một lúc. Hai hệ thống này đều được thiết lập với các mẫu có sẵn và cho ra kết quả khả quan.

Honeypot RL-Kippo đã sử dụng PyBrain, MySQL và mô-đun trạng thái/hành động tương tự như RASSH (được sửa đổi thành công thức (5)). Khung phần mềm RASSH đã thể hiện hành vi học tập tương tự như Heliza. Cả hai đều trình bày một biểu đồ giá trị hành động cho trạng thái Wget, cho phần thưởng với công thức phần thưởng đã sửa đổi (5). Trong thử nghiệm có kiểm soát của chúng tôi, Honeypot đã bị tấn công nhiều lần. Mỗi lệnh tấn công đều điền vào bảng giá trị Q như được mô tả trong Phần 3

4.2. Kết quả thí nghiệm

Ở mục này, nhóm sẽ trình bày các kết quả thực nghiệm và đưa ra đánh giá.

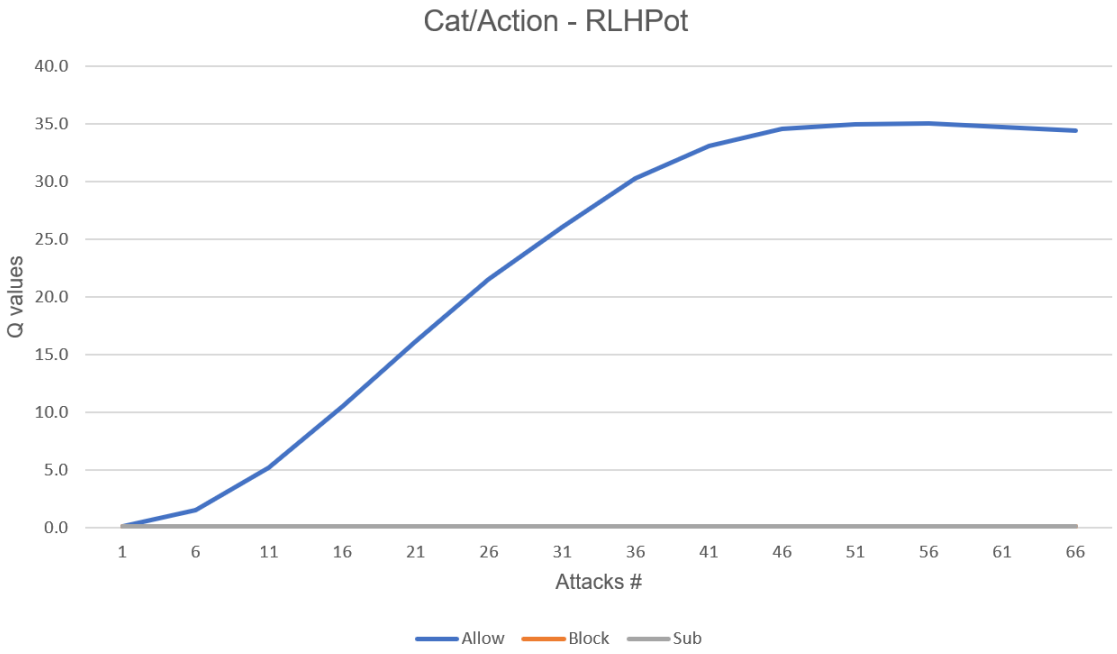
4.2.1. Triển khai và so sánh các Honeypot trực tiếp với một bộ hành động rút gọn nhắm mục tiêu vào phần mềm độc hại tự động đã biết



Hình 4.1: *Các giá trị Cat/action cho phần thưởng (5) khi triển khai RASSH.*

So sánh quá trình học của cả 2 loại Honeypot dựa vào hình 4.1 và hình 4.2 đã thực nghiệm với tập lệnh Cat lấy từ bảng 4.1 và được thực thi liên tục tổng cộng 70 lần. Với RASSH, giá trị Q của cả 3 trạng thái có xu hướng giảm xuống mặc dù ban đầu giá trị là 0.1. Trái ngược với Honeypot trên, RLHPot có xu hướng tăng điểm lên và khi tới mốc tấn công thứ 46, hàm phần thưởng (5) có xu hướng chững lại và ổn định dần và duy trì đến lần tấn công cuối cùng. Đó là trạng thái Allow còn 2 trạng thái còn lại thì không có sự thay đổi gì, vẫn chỉ là 0.1 là giá trị khởi tạo mặc định.

Đặc biệt, với RASSH chúng tôi thấy cả 3 đường trạng thái dường như bị dính chùm vào nhau, dẫn đến sẽ khó thấy rõ được, tuy vậy ở cột mốc lần tấn công



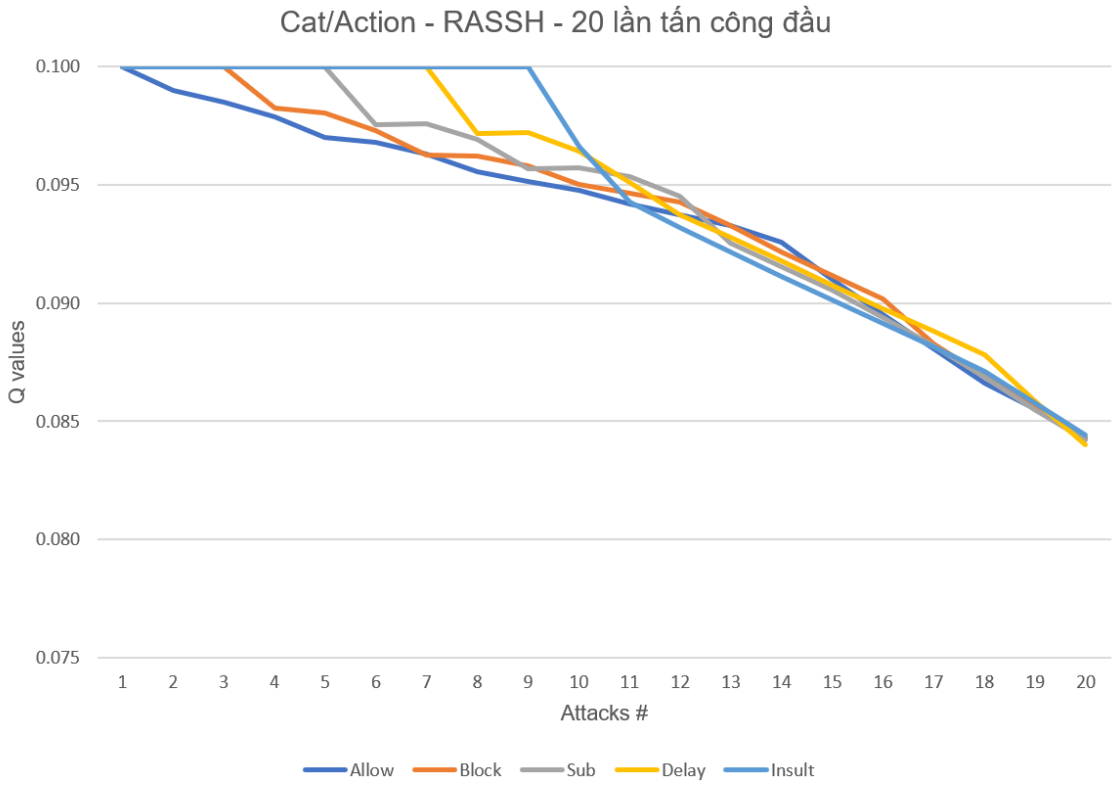
Hình 4.2: *Các giá trị Cat/action cho phần thưởng (5) khi triển khai RLHPot.*

thứ 21 về trước vẫn còn 1 đoạn mà cả 3 đường tách rời. Vì vậy chúng tôi quyết định tạo thêm biểu đồ hàm phần thưởng (5) với 20 lần tấn công đầu cho cả 2 Honeypot để tiện so sánh. Ở mục này, nhóm sẽ trình bày các kết quả thực nghiệm và đưa ra đánh giá.

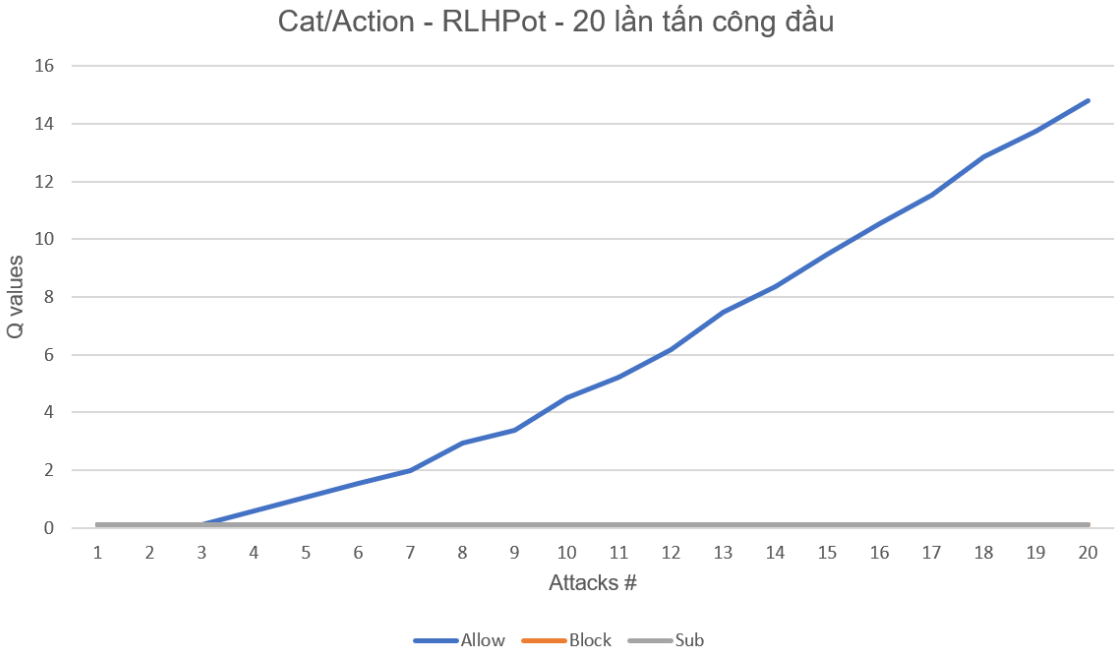
Tương tự như tập lệnh Cat,các lệnh Wget cũng được xem xét và thử nghiệm trên cả 2 Honeypot và cho ra kết quả tương đối khả quan, và chúng được trình bày qua hình 4.5 và hình 4.6

Tiếp tục so sánh quá trình học của cả 2 loại Honeypot dựa vào hình 4.5 và hình 4.6 đã thực nghiệm với tập lệnh Wget lấy từ 1 phần bảng 4.1 và được thực thi liên tục tổng cộng 106 lần. Ngay lần đầu tiên, có thể thấy giá trị Q của RASSH tăng đều lên trong khi đó RL-Kippo vẫn chỉ là 0.1. Chỉ khi tới lần tấn công thứ 46 đã có sự thay đổi rõ rệt khi mà Honeypot thứ hai cũng bắt đầu tăng vọt và vượt qua điểm phần thưởng của Honey đầu. Và từ lần đó cả 2 Honeypot đều giữ vững số điểm cho đến lần tấn công cuối cùng. Đó là trạng thái Allow, còn 2 trạng thái còn lại trong thực nghiệm không có sự biến động.

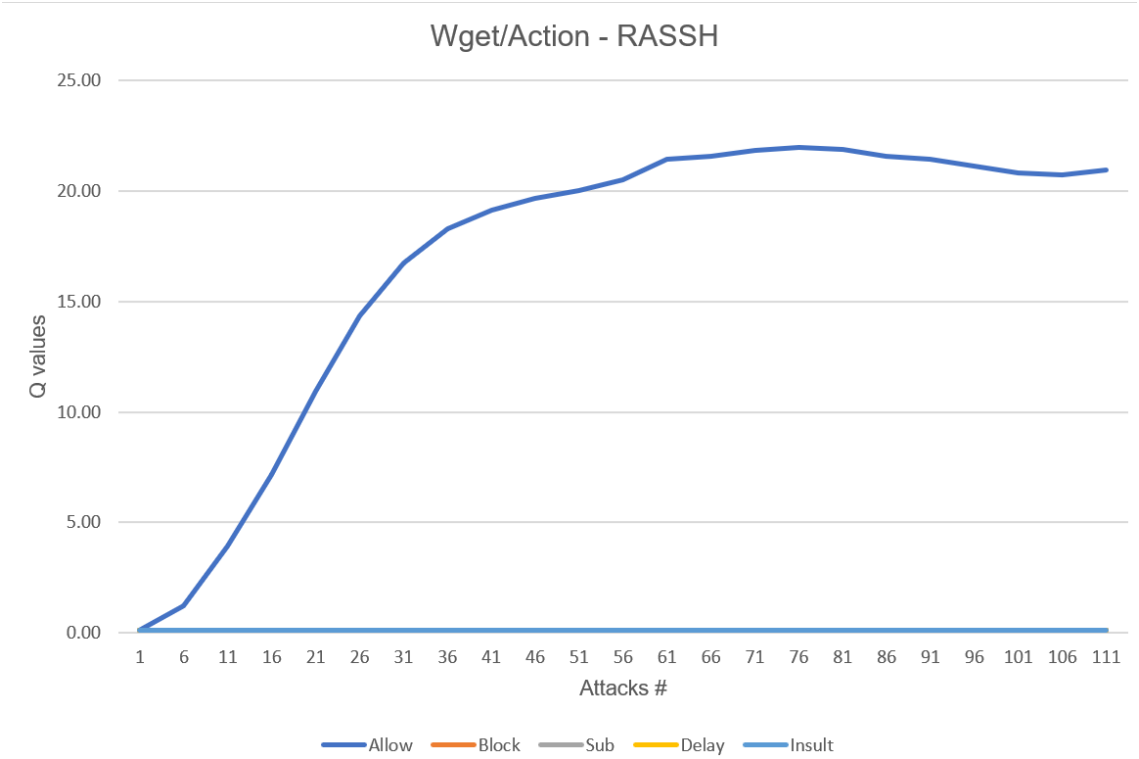
Mẫu bot ssh giống Mirai trong Bảng 4.1 cho thấy nó có sử dụng Cat và Wget



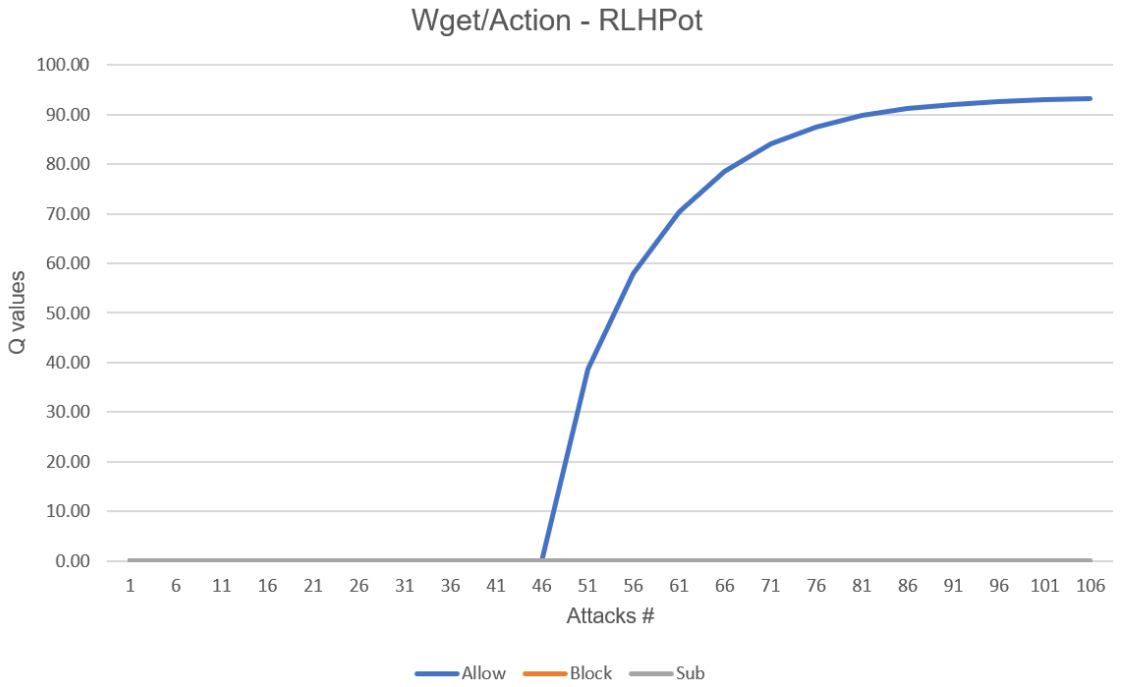
Hình 4.3: *Các giá trị Cat/action cho phần thưởng (5) khi triển khai RASSH trong 20 lần đầu.*



Hình 4.4: *Các giá trị Cat/action cho phần thưởng (5) khi triển khai RLHPot trong 20 lần đầu.*



Hình 4.5: *Các giá trị Wget/action cho phần thưởng (5) khi triển khai RASSH.*



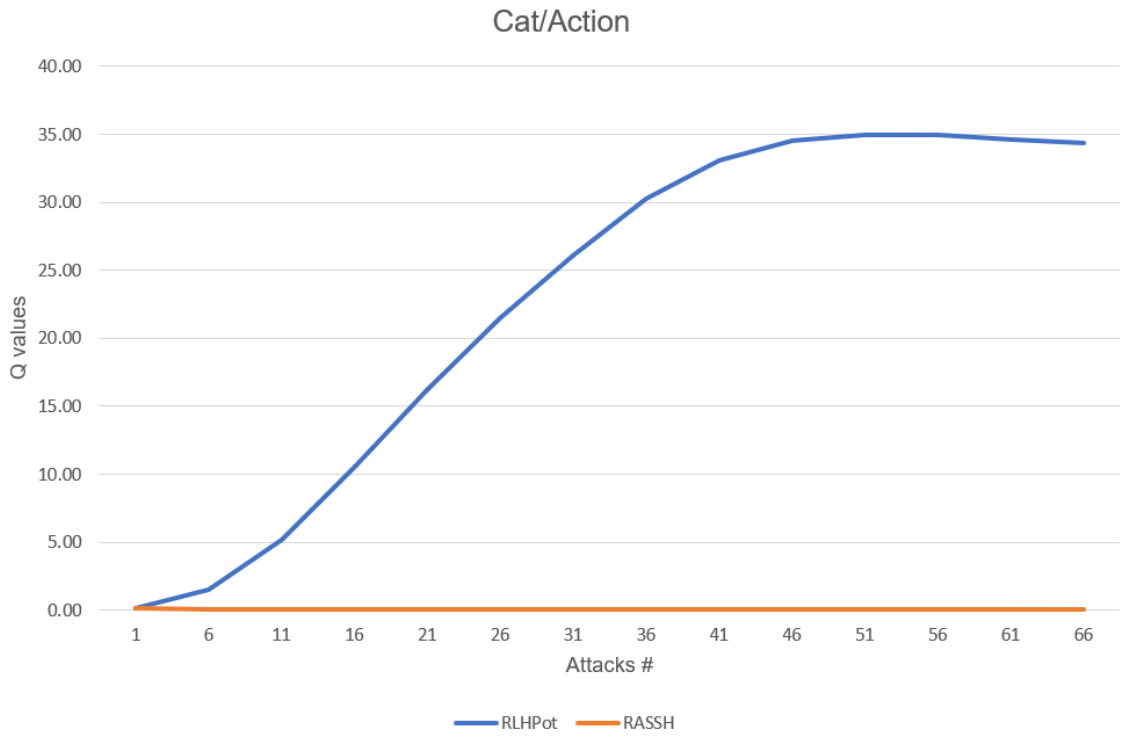
Hình 4.6: *Các giá trị Wget/action cho phần thưởng (5) khi triển khai RLHPot.*

trong cấu hình của nó. Điều này cho phép chúng tôi trích xuất từ tập dữ liệu phần thưởng được trao cho Wget/action và Cat/action. Chúng tôi tập trung vào việc thu thập từ 70 đến 100 cuộc tấn công từ bot này. Phần mềm độc hại SSH khác đã tương tác với Honeypot. Nhưng những lệnh này thường sử dụng các lệnh khác nhau (chẳng hạn như cuộn tròn) hoặc quá hiếm khi sửa đổi quá mức phần thưởng. Các biểu đồ được tạo ra từ các trích xuất này được trình bày trong Hình 5 và 6.

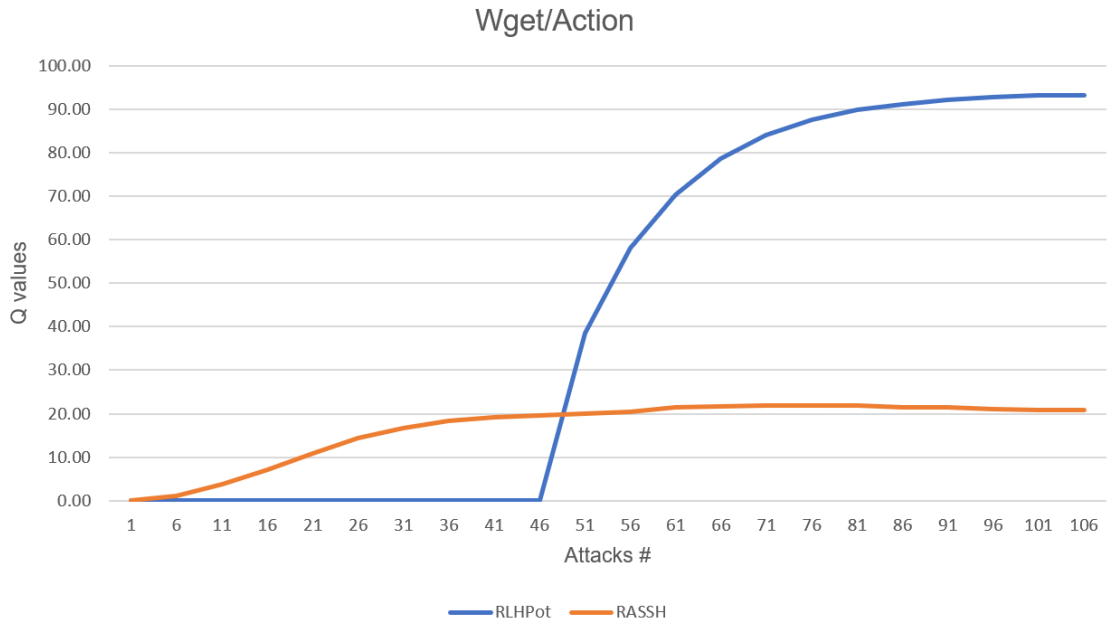
Hình 4.2 và 4.6 phản ánh rõ nhất về sự cải tiến. Chúng tôi có thể thấy rõ rằng Honeypot thích ứng xác định cho phép, là hành động tốt nhất để thực hiện khi tấn công lần thứ 46 (Wget) và lần đầu (Cat). Mặc dù Honeypot đã bắt được 100 cuộc tấn công, nhưng các biểu đồ thể hiện một động thái hướng tới hội tụ về chính sách học tập tối ưu. Vì mỗi cuộc tấn công có cả lệnh Cat và lệnh Wget trong chuỗi tấn công, chúng tôi có thể tạo ra các biểu đồ chi tiết hơn khi so sánh với các biểu đồ từ RASSH. Trong cả hai biểu đồ, cho phép là những hành động tạo ra phần thưởng tốt nhất. Khi kiểm tra nhật ký từ bộ dữ liệu Honeypot, chúng tôi nhận thấy việc chặn các lệnh đôi khi dẫn đến việc lệnh được phát hành lại. Điều này ngay lập tức đóng góp vào số lần chuyển đổi nhưng thú vị hơn là nó cung cấp thông tin chi tiết về hoạt động của bot mà không cần dùng đến hộp cát để kiểm tra mã.

4.2.2. Triển khai và so sánh RASSH và RLHPot

Chúng tôi đã triển khai hai Honeypot cùng một lúc. Để so sánh hiệu suất, chúng tôi trích xuất tất cả các lệnh được thực thi trên cả 2 Honeypot. Kippo lưu trữ tất cả các tương tác trong tệp nhật ký theo tiêu chuẩn. Nó cũng lưu trữ tất cả các tệp đã tải xuống trong một thư mục tải xuống. Điều này giúp có thể tích lũy số lượng tất cả các lệnh đã thử trên Honeypot. Tất cả các lệnh này đại diện cho các tương tác hậu thỏa hiệp. Do đó, các sự kiện như lần thử không thành công, tấn công từ điển và bruteforce đều bị loại trừ vì chúng thể hiện các tương tác trước khi thỏa hiệp. Tương tác chiếm ưu thế đến từ bot SSH giống



Hình 4.7: *Các giá trị Wget/action cho phần thưởng (5) khi triển khai cả 2 Honeypot.*



Hình 4.8: *Các giá trị Wget/action cho phần thưởng (5) khi triển khai cả 2 Honeypot.*

Mirai như đã thấy trong Bảng 4.1.

Với kết quả cuối cùng, có thể thấy cả 2 Honeypot ban đầu chỉ trải qua vài lệnh đầu tiên nhưng sau đó số lượng chuỗi bắt đầu tăng lên khi Honeypot học được từ các hành động và phần thưởng trạng thái của nó. Phần thưởng của chúng tôi (5) có mục tiêu tối đa hóa quá trình chuyển đổi. Hình trình bày sự so sánh các chuyển đổi tích lũy cho cả hai Honeypot. Điều này bao gồm tất cả các chuyển đổi lệnh cho các lệnh công cụ trạng thái và kẻ tấn công. Càng về sau, điểm số lại có sự chênh lệch lớn. Nhật ký cho thấy rằng lần đầu tiên Honeypot thích ứng cho phép và tiếp tục như vậy cho đến khi kết thúc tất cả tấn công. Nhìn chung, có thể thấy rằng RL-Kippo dù điểm phần thưởng tăng trễ hơn nhưng về sau có điểm cao hơn và bắt được nhiều lệnh tấn công hơn so với RASSH. Điều tra các mưu đồ của bot, thông qua hộp cát và khám phá mã, có thể giải thích sự phát triển học tập này. Tuy nhiên, điều này nằm ngoài phạm vi của bài viết này. Nhiệm vụ này chứng minh rằng các hũ mật thích ứng làm tăng đáng kể kích thước của bộ dữ liệu có thể được thu thập trên Honeypot.

Thật khó để so sánh sự tương đồng với nghiên cứu trước đây khi triển khai các Honeypot trong các thí nghiệm trong thế giới thực. Một biến ảnh hưởng đến kết quả là loại lưu lượng gặp phải trên Honeypot. Phần mềm độc hại phát triển rất nhanh và biến đổi thành các phiên bản khác nhau với các phương pháp tấn công thay thế. Cấu trúc của bot gặp phải trên Honeypot của chúng tôi có một phương pháp cụ thể và tác động đáng kể đến kết quả tổng thể trong Hình . Chúng tôi xem các hình từ RLHPot là sự phản ánh trung thực hơn về hiệu quả của một Honeypot thích ứng. Nó cũng phản ánh phần thưởng của chúng tôi, xem Phương trình (5), phần thưởng cho người học của chúng tôi khi lệnh tấn công là lệnh hệ thống hoặc lệnh của kẻ tấn công.

CHƯƠNG 5. KẾT LUẬN

Ở chương này, chúng tôi đưa ra những kết luận về nghiên cứu, những hạn chế, và đồng thời đưa ra hướng cải thiện và phát triển.

5.1. Kết luận

Nghiên cứu được trình bày ở đây sử dụng một hình thức không gian hành động trạng thái mới cho phần mềm độc hại tự động. Nó được xây dựng khi hoàn thành công việc trong lĩnh vực này và cải thiện quy trình RL được triển khai trên một Honeypot thích ứng. Chúng tôi đã trình bày một sự cải thiện về sự hội tụ đối với chính sách học tập tối ưu và một dấu hiệu rõ ràng về một hành động ưu tiên ở giai đoạn đầu. Chính sách này góp phần làm tăng đáng kể số lần chuyển đổi lệnh và khối lượng tập dữ liệu tiếp theo. Mục đích chính của Honeypot là thu thập các bộ dữ liệu có thể được sử dụng để phân tích các phương pháp của các nhà phát triển phần mềm độc hại. Các phương thức giao tiếp và thiết bị đầu cuối mới cho các công nghệ hiện có như IoT, IoT công nghiệp, trò chơi, v.v., làm tăng khả năng tấn công.

Nghiên cứu này nhấn mạnh vai trò chủ đạo của lưu lượng truy cập tự động đối với việc lan truyền phần mềm độc hại. Vai trò này thể hiện rõ ràng trên các bộ dữ liệu Honeypot đã chụp. Điều rất quan trọng cần lưu ý là tất cả lưu lượng truy cập gặp phải trên các Honeypot tương tác cao tiêu chuẩn và thích ứng của chúng tôi, được triển khai trên Internet, hoàn toàn tự động. Không có sự tương tác của con người trong các tệp nhật ký, được xác minh bằng dấu thời gian của các lệnh. Phương pháp trong bài viết này đã chứng minh khả năng học tập được cải thiện và thu thập dữ liệu tổng thể cho các Honeypot thích ứng sử dụng RL. Các nhà phát triển phần mềm độc hại liên tục tìm cách xâm phạm công nghệ

mới vì những lý do độc hại hoặc vì lợi ích tài chính. Honeypot cung cấp cho các nhóm bảo mật thông tin chi tiết có giá trị về hành vi của kẻ tấn công. Nó chỉ là một công cụ được sử dụng trong cuộc chiến chống lại sự lan truyền phần mềm độc hại. Bằng cách tăng số lượng tương tác tấn công, chúng tôi tin rằng phương pháp này có thể góp phần vào cuộc chiến.

5.2. Hướng phát triển

AWS giúp dễ dàng xác định vị trí một Honeypot đối mặt với Internet tại một vị trí địa lý cụ thể. Chúng tôi sẽ triển khai Honeypot độc lập của mình trên dịch vụ AWS EC2. Honeypot này kết hợp PyBrain và dữ liệu từ quá trình học được lưu trữ trong cơ sở dữ liệu MySQL phụ trợ. AWS cũng cung cấp các dịch vụ khác như Virtual Private Cloud (VPC) và Relational Database Services (RDS). Khi sử dụng các dịch vụ này, chúng tôi có thể kết nối hàng loạt Honeypot (Honeynet) thông qua VPC và yêu cầu chúng cập nhật cơ sở dữ liệu trung tâm nằm trên một phiên bản EC2 khu vực. Khi mỗi Honeypot trong sê-ri trải qua một đợt tấn công, nó sẽ cập nhật cơ sở dữ liệu trung tâm. Các cuộc tấn công tiếp theo vào tất cả các Honeypot khác tiếp tục quá trình cập nhật các giá trị phần thưởng trung tâm này. Công việc của chúng tôi hiện tập trung vào việc cải thiện thời gian cần thiết để một Honeynet hội tụ theo chính sách tối ưu, đồng thời thực nghiệm trên nhiều môi trường và tình cảnh khác nhau, cho ra kết quả khả khả quan hơn.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] (2017), "Cowrie Honeypot", *Available from: http://www.micheloosterhof.com/cowrie.*

[2] Abbes T. Bouhoula A Ghourabi A. (2014), "Characterization of attacks collected from the deployment of Web service honeypot", *Security and Communication Networks*, 7 (2), pp. 338–351.

[3] Youssef A Hayatle O Otrok H (2013), "A Markov decision process model for high interaction honeypots", *Information Security Journal: A Global Perspective*, 22 (4), 159–170.

[4] G Navarro (2001), "A guided tour to approximate string matching", *Journal of artificial intelligence research*, 33 (1), pp. 31–88.

[5] Owezarski P (2014), "Unsupervised classification and characterization of honeypot attacks", *10th International Conference on Network and Service Management (CNSM) and Workshop.*

[6] Bica I Pauna A (2014), "RASSH-Reinforced adaptive SSH honeypot", *2014 10th International Conference on Communications (COMM).*

[7] Sosdow (2017), "An adaptive honeypot using reinforcement learning implementation", *Available from: https://github.com/sosdow/RLHPot.*

[8] Woodward A Valli C Rabadia P (2013), "Patterns and patter - an investigation into ssh activity using Kippo honeypots", *Australian Digital Forensics Conference.*

[9] State R. Dulaunoy A. Engel T Wagener G. (2011), "Heliza: talking dirty to the attackers", *Journal in computer virology*, 7 (3), 221–232.

[10] Engel T Wagener G Radu State DA, “Self-adaptive high interaction honeypots driven by game theory”, in: SSS, 2009, 741–755.