|  |  |
| --- | --- |
|  |  |
| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **---------------------------------------** | |
| **[Học viện Công nghệ Bưu chính Viễn thông](http://ptit.edu.vn/Default.)** | |
|  | |
| **PHẠM HOÀNG LONG** | |
|  | |
|  | |
| **ỨNG DỤNG HỌC CỦNG CỐ (REINFORCEMENT LEARNING)**  **TRONG PHÂN TÍCH VÀ PHÁT HIỆN MÃ ĐỘC IOT BOTNET** | |
| **LUẬN VĂN THẠC SĨ KỸ THUẬT**  **(Theo định hướng nghiên cứu)**  HÀ NỘI - 2020 | |
|  |  |
| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **---------------------------------------** | |
| **[Học viện Công nghệ Bưu chính Viễn thông](http://ptit.edu.vn/Default.)** | |
|  | |
| **PHẠM HOÀNG LONG** | |
|  | |
| **ỨNG DỤNG HỌC CỦNG CỐ (REINFORCEMENT LEARNING)**  **TRONG PHÂN TÍCH VÀ PHÁT HIỆN MÃ ĐỘC IOT BOTNET** | |
| CHUYÊN NGÀNH : **KHOA HỌC MÁY TÍNH** | |
| MÃ SỐ: 8.48.01.01    **LUẬN VĂN THẠC SĨ KỸ THUẬT**  **NGƯỜI HƯỚNG DẪN KHOA HỌC TS. NGÔ QUỐC DŨNG**  **HÀ NỘI - 2020** | |

# LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tác giả luận văn

Phạm Hoàng Long

LỜI CẢM ƠN

Để hoàn thành được luận văn này, em xin gửi lời cảm ơn sâu sắc nhất tới TS. Ngô Quốc Dũng, giảng viên trực tiếp hướng dẫn, tận tình chỉ bảo và định hướng cho em trong suốt quá trình nghiên cứu và thực hiện luận văn.

Em xin gửi lời cảm ơn chân thành tới tất cả các thầy cô giáo của Học viện Công nghệ Bưu chính Viễn thông đã giảng dạy và chỉ bảo cho em trong tất cả các môn học để em có thể tiếp thu được nhiều nhất những kiến thức bổ ích.

Em cũng xin cảm ơn anh Trung và em Hiếu đã giúp đỡ rất nhiều để em có thể có điều kiện tốt nhất để hoàn thành luận văn này.

Cuối cùng, em xin gửi lời cảm ơn tới gia đình, bạn bè và những người đã luôn ở bên cổ vũ tinh thần, tạo điều kiện thuận lợi cho em để em có thể học tập tốt và hoàn thiện luận văn.

Dù đã cố gắng hết sức nhưng trong luận văn không thể tránh khỏi những sai sót, em mong nhận được sự góp ý để hoàn thiện hơn.

Em xin chân thành cảm ơn!

# MỤC LỤC

[LỜI CAM ĐOAN i](#_Toc56025164)

[LỜI CẢM ƠN ii](#_Toc56025165)

[MỤC LỤC iii](#_Toc56025166)

[DANH SÁCH BẢNG vi](#_Toc56025167)

[DANH SÁCH HÌNH VẼ vii](#_Toc56025168)

[MỞ ĐẦU 1](#_Toc56025169)

[Chương 1. Mã độc IoT Botnet và các nghiên cứu liên quan 4](#_Toc56025170)

[1.1. Tổng quan về IoT 4](#_Toc56025171)

[1.2. Kiến trúc cơ bản của IoT 5](#_Toc56025172)

[1.2.1. Sensor 6](#_Toc56025173)

[1.2.2. Actuator 6](#_Toc56025174)

[1.2.3. Device 7](#_Toc56025175)

[1.2.4. Gateway 7](#_Toc56025176)

[1.2.5. IoT Integration Middleware 8](#_Toc56025177)

[1.2.6. Application 8](#_Toc56025178)

[1.3. Vấn đề an ninh với IoT 8](#_Toc56025179)

[1.4. Mã độc IoT Botnet 9](#_Toc56025180)

[1.4.1. Giới thiệu chung 9](#_Toc56025181)

[1.4.2. Các thành phần cơ bản của mạng IoT botnet 10](#_Toc56025182)

[1.4.3. Vòng đời của một mạng IoT Botnet cơ bản 11](#_Toc56025183)

[1.4.4. Ví dụ về mã độc IoT Botnet (Mirai Botnet) 15](#_Toc56025184)

[1.5. Các phương pháp phát hiện mã độc IoT Botnet 17](#_Toc56025185)

[1.5.1. Phương pháp phân tích động 17](#_Toc56025186)

[1.5.2. Phương pháp phân tích tĩnh 19](#_Toc56025187)

[1.6. Tổng kết chương 19](#_Toc56025188)

[Chương 2. Xây dựng thuật toán trích chọn đặc trưng bằng phương pháp học củng cố và áp dụng để phát hiện mã độc 21](#_Toc56025189)

[2.1. Giới thiệu về đồ thị PSI 21](#_Toc56025190)

[2.1.1. Xây dựng đồ thị hàm gọi 21](#_Toc56025191)

[2.1.2. Xây dựng đồ thị PSI 22](#_Toc56025192)

[2.2. Giới thiệu học củng cố 23](#_Toc56025193)

[2.2.1. Giới thiệu 23](#_Toc56025194)

[2.2.2. Một vài thuật toán học củng cố hay được sử dụng 25](#_Toc56025195)

[2.3. Giới thiệu về mạng nơ-ron hồi quy phân loại 28](#_Toc56025196)

[2.3.1. Giới thiệu 28](#_Toc56025197)

[2.3.2. Kiến trúc mạng 29](#_Toc56025198)

[2.4. Ứng dụng học củng cố trong phân tích và phát hiện mã độc 33](#_Toc56025199)

[2.4.1. Tổng quan 33](#_Toc56025200)

[2.4.2. Định nghĩa PSI-walk 35](#_Toc56025201)

[2.4.3. Học củng cố 37](#_Toc56025202)

[2.4.4. Phân loại 41](#_Toc56025203)

[2.5. Tổng kết chương 42](#_Toc56025204)

[Chương 3. KẾT QUẢ THỰC NGHIỆM 43](#_Toc56025205)

[3.1. Bộ dữ liệu PSI-graph 43](#_Toc56025206)

[3.2. Bộ dữ liệu PSI-walk 44](#_Toc56025207)

[3.3. Tiêu chí đánh giá 45](#_Toc56025208)

[3.4. Kết quả và đánh giá 46](#_Toc56025209)

[KẾT LUẬN 51](#_Toc56025210)

[DANH MỤC TÀI LIỆU THAM KHẢO 52](#_Toc56025211)

# DANH SÁCH BẢNG

[Bảng 3.1. Mô tả bộ dữ liệu PSI-graph 44](#_Toc55336929)

[Bảng 3.2. Thống kê về số lượng cạnh và đỉnh trong PSI-graph 44](#_Toc55336930)

[Bảng 3.3. Kết quả thực nghiệm 47](#_Toc55336931)

# DANH SÁCH HÌNH VẼ

[Hình 1.1. Kiến trúc IoT tham khảo 6](#_Toc56025584)

[Hình 1.2. Các thành phần cơ bản của mạng IoT botnet[13] 10](#_Toc56025585)

[Hình 1.3. Dòng sự kiện liên quan đến Mirai botnet[16] 15](#_Toc56025586)

[Hình 1.4. Cơ chế lây lan của Mirai botnet 17](#_Toc56025587)

[Hình 2.1. Một phần của biểu đồ gọi hàm của mẫu Linux.Mirai[9] 22](#_Toc56025588)

[Hình 2.2. Chu kỳ học củng cố 25](#_Toc56025589)

[Hình 2.3. Mô hình hoạt động của Q-learning và Deep Q-learning 26](#_Toc56025590)

[Hình 2.4. Mạng nơ-ron hồi quy với khuyên tại nơ-ron 29](#_Toc56025591)

[Hình 2.5. Sơ đồ mạng nơ-ron hồi quy 29](#_Toc56025592)

[Hình 2.6. Một cách phân loại các mạng nơ-ron hồi quy 30](#_Toc56025593)

[Hình 2.7. Sơ đồ RNN hai chiều 31](#_Toc56025594)

[Hình 2.8. Sơ đồ RNN sâu 2 chiều 32](#_Toc56025595)

[Hình 2.9. Một nơ-ron trong LSTM 33](#_Toc56025596)

[Hình 2.10. Tổng quan mô hình đề xuất 34](#_Toc56025597)

[Hình 2.11. Một ví dụ về PSI-walk 36](#_Toc56025598)

[Hình 2.12. Kiến trúc mạng phân loại LSTM 41](#_Toc56025599)

[Hình 3.1. Hình ảnh minh họa các PSI-walk và nhãn của chúng 45](#_Toc56025600)

[Hình 3.2. Đường ROC của model 47](#_Toc56025601)

[Hình 3.3. Confusion matrix 48](#_Toc56025602)

[Hình 3.4. Lịch sử huấn luyện 49](#_Toc56025603)

MỞ ĐẦU

Cách mạng Công nghiệp 4.0 (hay Cách mạng Công nghiệp lần thứ Tư) là sự ra đời của một loạt các công nghệ mới, kết hợp tất cả các kiến thức trong lĩnh vực vật lý, kỹ thuật số, sinh học, và ảnh hưởng đến tất cả các lĩnh vực, nền kinh tế, các ngành kinh tế và ngành công nghiệp. Đặc biệt là sự phát triển của Internet (IoT) đang diễn ra mạnh mẽ trên toàn thế giới. Theo dự báo mới nhất, quy mô thị trường của IoT được dự kiến là tổng cộng 4,3 nghìn tỷ đô la vào năm 2020, nhiều hơn 456% so với một phân khúc trong năm 2014 [12] .Và theo một báo cáo gần đây, sẽ có khoảng 8.4 tỷ kết nối trên toàn thế giới vào năm 2020. Con số này dự kiến sẽ tăng lên 20,4 tỷ vào năm 2022[4]. Việc sử dụng các ứng dụng IoT đang gia tăng ở tất cả các nơi trên thế giới. Các quốc gia chính trong đó bao gồm Tây Âu, Bắc Mỹ và Trung Quốc [4]. Số lượng kết nối máy với máy (M2M) dự kiến sẽ tăng từ 5,6 tỷ vào năm 2016 lên 27 tỷ vào năm 2024[4]. Bước nhảy vọt về số lượng này cho thấy sự mở rộng của công nghiệp IoT và có thể tạo thành nền tảng của nền kinh tế kỹ thuật số. Ngành công nghiệp IoT dự kiến sẽ tăng trưởng về doanh thu từ 892 tỷ đô la năm 2018 lên 4 nghìn tỷ đô la vào năm 2025[7].

Với sự phát triển nhanh và ở được áp dụng trong mọi lĩnh vực của đời sống thì việc bảo mật là một trong những vấn đề lớn nhất với IoT. những thách thức về bảo mật của IoT đặc biệt như các vấn đề về quyền riêng tư, xác thực, quản lý, lưu trữ thông tin, v.v. Do những lỗ hổng này, các ứng dụng IoT bị đe dọa bởi các mối nguy hiểm khác nhau . Đã có nhiều cuộc tấn công bảo mật và quyền riêng tư vào các ứng dụng đã được triển khai trên toàn thế giới. Cuộc tấn công Mirai trong quý cuối năm 2016 được ước tính đã lây nhiễm khoảng 2,5 triệu thiết bị được kết nối với Internet và khởi động cuộc tấn công từ chối dịch vụ phân tán (DDoS) [6]. Sau Mirai, Hajime và Reaper là các cuộc tấn công botnet lớn khác đã phát động chống lại một số lượng lớn các thiết bị IoT[6]. Sau những cuộc tấn công như vậy thì hậu quả để lại là không thể đo đếm được.

Do đó, việc xử lý bảo mật trong IoT ngày càng trở nên phổ biến đối với các nhà nghiên cứu, đặc biệt là các vấn đề về phần mềm độc hại IoT. Gần đây, đã có nhiều nghiên cứu với mục tiêu bảo mật cho các thiết bị IoT khỏi các mối đe dọa của các phần mềm độc hại. Những nghiên cứu này có thể được phân loại thành hai lĩnh vực chính: phân tích động và phân tích tĩnh .

Phương pháp động bao gồm giám sát các lệnh thực thi trong thời gian chạy và phát hiện các hành vi bất thường. Cách tiếp cận này được thực hiện bằng cách thu thập thông tin như các cuộc gọi hệ thống, hành vi mạng, dấu vết hướng dẫn, thay đổi sổ đăng ký và ghi bộ nhớ trong quá trình thực tế. Để thực hiện phương pháp này, phần quan trọng nhất là xây dựng một môi trường đầy đủ (ví dụ: trình giả lập ảo, thiết bị thực tế) cho phần mềm độc hại để thể hiện hành vi.Tuy nhiên, việc giám sát các quy trình thực hiện rất tốn tài nguyên và trong một số trường hợp, phần mềm độc hại có thể lây nhiễm vào môi trường thực. Ngoài ra, trong thời gian thực hiện, không thể giám sát đầy đủ các hành vi vì nhiều loại phần mềm độc hại yêu cầu điều kiện kích hoạt để thể hiện các hành vi độc hại. Ngoài các giới hạn phổ biến của phân tích động, việc thực thi các ứng dụng thực thi IoT còn đối mặt với nhiều vấn đề như kiến trúc đa dạng (ví dụ: MISP. ARM, PowerPC, Sparc) và hạn chế tài nguyên của các thiết bị IoT. Do đó, khó có thể đảm bảo môi trường nền tảng đáp ứng các yêu cầu để thực thi IoT hoạt động chính xác và đầy đủ.

Phương pháp tĩnh được thực hiện bằng cách phân tích và phát hiện các độc hại mà không thực hiện chúng. Một trong những ưu điểm chính của phân tích tĩnh là khả năng quan sát cấu trúc của phần mềm độc hại, tất cả các đường dẫn thực thi có thể có trong mẫu phần mềm độc hại mà không xem xét tính đa dạng của kiến trúc bộ xử lý, do đó làm cho phương pháp này có hiệu quả để giải quyết các vấn đề về gen dị của các thiết bị IoT. Một trong những phương pháp phân tích tĩnh đó là phân tích dựa trên đồ thị đặc trưng PSI.Đồ thị PSI là đồ thị biểu diễn hành vi của các tệp thực thi trong file ELF . Nhận thấy việc phát hiện mã độc dựa trên phân tích các đặc trưng dưới dạng đồ thị (graph-based) đem lại hiệu quả hơn so với phân tích đặc trưng dưới dạng luồng (flow-based), do dạng đồ thị có thể thể hiện được mối quan hệ chặt chẽ giữa dữ liệu và giảm thời gian so sánh chéo trong mô hình. Trong đó có thể chia các đặc trưng thành đặc trưng cấu trúc và đặc trưng phi cấu trúc.

Tuy nhiên với đồ thị PSI vẫn còn dư thừa rất nhiều đường đi không phải hành vi của mã độc ,dễ dẫn đến việc gây nhiễu cho mô hình phát hiện mã độc sau này .Trong luận văn này đề xuất sử dụng phương pháp học củng cố để học được hành vi tương ứng với đường đi của mã độc trên đồ thị PSI

Nội dung luận văn được chia làm 3 chương như sau

* **CHƯƠNG 1:** Giới thiệu về IoT Botnet , các tác hại của IoT Botnet đối với mọi mặt trong cuộc sống .Các thành phần và cách hoạt động của mã độc IoT Botnet và các phương pháp để phát hiện chúng.
* **CHƯƠNG 2:**Ứng dụng học củng cố trên đồ thị đặc trưng PSI để phân tích và phát hiện mã độc.
* **CHƯƠNG 3:** Thử nghiệm và đánh giá: Mô tả quá trình xây dựng bộ dữ liệu ,đánh giá hiệu quả của mô hình đề xuất.

# Mã độc IoT Botnet và các nghiên cứu liên quan

*Chương này sẽ trình bày về khái niệm Iot Botnet, các thành phần và cách thức hoạt động của mã độc. Giới thiệu các phương pháp phân tích động và tĩnh để phát hiện mã độc cùng với những ưu nhược điểm của các phương pháp từ đó đưa ra hướng tiếp cận mới.*

## Tổng quan về IoT

Internet of Things, hay IoT, internet vạn vật là đề cập đến hàng tỷ thiết bị (vật lý hoặc ảo hóa ) trên khắp thế giới hiện được kết nối với internet, thu thập và chia sẻ dữ liệu. Nhờ bộ xử lý giá rẻ và mạng không dây, có thể biến mọi thứ kết nối được với nhau. Điều này bổ sung sự “thông minh kỹ thuật số” cho các thiết bị, cho phép chúng giao tiếp mà không cần có con người tham gia và hợp nhất thế giới kỹ thuật số và vật lý.

Ý tưởng về việc thêm cảm biến và trí thông minh vào các đối tượng cơ bản đã được thảo luận trong suốt những năm 1980 và 1990, nhưng ngoài một số dự án ban đầu – bao gồm cả máy bán hàng tự động kết nối internet – nguyên nhân chỉ đơn giản là vì công nghệ chưa sẵn sàng.

Vì vậy khi phát minh ra việc sử dụng thẻ RFID – chip năng lượng thấp có thể giao tiếp không dây – đã giải quyết được một số vấn đề này, cùng với sự sẵn có ngày càng cao của internet băng thông rộng và mạng di động và mạng không dây. Việc áp dụng IPv6 sẽ cung cấp đủ địa chỉ IP cho mọi thiết bị trên thế giới cũng là một bước cần thiết để IoT mở rộng quy mô. Kevin Ashton đã đặt ra cụm từ ‘Internet of Things’ vào năm 1999, mặc dù phải mất ít nhất một thập kỷ nữa để công nghệ bắt kịp tầm nhìn.

Gắn thêm thẻ RFID vào các thiết bị đắt tiền để giúp theo dõi vị trí của chúng là một trong những ứng dụng IoT đầu tiên. Nhưng kể từ đó, chi phí cho việc thêm cảm biến và kết nối internet vào các đối tượng đã tiếp tục giảm và các chuyên gia dự đoán rằng chức năng cơ bản này một ngày có thể chỉ tốn 10 xu, giúp kết nối gần như mọi thứ với internet.

IoT ban đầu thú vị nhất đối với kinh doanh và sản xuất, trong đó ứng dụng của nó đôi khi được gọi là machine-to-machine (M2M), nhưng giờ đây người ta nhấn mạnh vào việc lấp đầy nhà cửa và văn phòng của chúng ta bằng các thiết bị thông minh, biến nó thành thứ gì đó phù hợp với hầu hết tất cả mọi người.

Ngày nay các thiết bị IoT đang ảnh hưởng đến nhiều khía cạnh khác nhau của cuộc sống từ y tế, nông nghiệp, quản lý năng lượng cho tới lĩnh vực quân sự… và dần trở thành xu thế.

Theo dự báo mới nhất, quy mô thị trường của IoT được dự kiến là tổng cộng 4,3 nghìn tỷ đô la vào năm 2020, nhiều hơn 456% so với một phân khúc trong năm 2014 [12] .Và theo một báo cáo gần đây, sẽ có khoảng 8.4 tỷ kết nối trên toàn thế giới vào năm 2020. Con số này dự kiến sẽ tăng lên 20,4 tỷ vào năm 2022[4]. Việc sử dụng các ứng dụng IoT đang gia tăng ở tất cả các nơi trên thế giới. Các quốc gia chính trong đó bao gồm Tây Âu, Bắc Mỹ và Trung Quốc [4]. Số lượng kết nối máy với máy (M2M) dự kiến sẽ tăng từ 5,6 tỷ vào năm 2016 lên 27 tỷ vào năm 2024[4]. Bước nhảy vọt về số lượng này cho thấy sự mở rộng của công nghiệp IoT và có thể tạo thành nền tảng của nền kinh tế kỹ thuật số. Ngành công nghiệp IoT dự kiến sẽ tăng trưởng về doanh thu từ 892 tỷ đô la năm 2018 lên 4 nghìn tỷ đô la vào năm 2025[7].

## Kiến trúc cơ bản của IoT

IoT được ứng dụng trong nhiều lĩnh vực, đi cùng với đó là nhiều nền tảng, tiêu chuẩn công nghệ khác ra đời. Tuy nhiên xuất hiện vấn đề không đồng nhất của các nền tảng IoT. Vấn đề chính là tìm một nền tảng IoT phù hợp cho một lĩnh vực ứng dụng nhất định. Mặc dù các nền tảng IoT cung cấp chức năng tương tự hoặc thậm chí như nhau, nhưng việc triển khai và các công nghệ cơ bản của chúng khác nhau. Điều này dẫn đến các khái niệm và kiến trúc đa dạng, làm phức tạp việc so sánh nhiều nền tảng.

Vì không có kiến trúc chung được áp dụng, người dùng phải đi sâu vào mô tả các nền tảng và phải hiểu từng kiến trúc và các thành phần của chúng từ đầu. Thủ tục này tốn thời gian và cần phải biết trước. Từ đó việc có một kiến trúc tham chiếu trừu tượng là cần thiết.

Một kiến trúc IoT dựa đề xuất trên tham chiếu một số nền tảng IoT hiện đại. Hơn nữa, kiến ​​trúc tham chiếu được so sánh với ba nền tảng IoT nguồn mở OpenMTC, FIWARE, Site- Where và Amazon Web Services IoT.

A screenshot of a cell phone

Description automatically generated

Hình 1.1. Kiến trúc IoT tham khảo

### Sensor

Sensor là một thành phần phần cứng, được sử dụng để đo các tham số của môi trường vật lý của nó và để chuyển chúng thành tín hiệu điện, ví dụ, bằng cách đo nhiệt độ hoặc độ ẩm của phòng. Nếu được yêu cầu, Sensor có thể được cấu hình bằng phần mềm, nhưng không thể tự chạy phần mềm. Thông thường, Sensor được kết nối hoặc được tích hợp vào Device gửi dữ liệu được thu thập. Các ví dụ nổi bật cho các Device là RaspberryPis, BananaPis, Arduino hoặc BeagleBones. Kết nối có thể được thiết lập bằng dây hoặc không dây, ví dụ, qua radio.

### Actuator

Actuator là một thành phần phần cứng, có thể tác động, kiểm soát hoặc điều khiển môi trường vật lý, ví dụ, bằng cách đưa ra tín hiệu quang hoặc âm thanh. Actuator nhận lệnh từ Device mà nó được kết nối. Họ chuyển các tín hiệu điện thành hành động vật lý tương ứng. Cũng giống như Sensor, Actuator thường được kết nối hoặc thậm chí được tích hợp vào Actuator, nhờ đó kết nối có thể được thiết lập bằng dây hoặc không dây. Nếu được yêu cầu, Actuator truyền động có thể được cấu hình bằng phần mềm nhưng không thể tự chạy phần mềm (nghĩa là phải nhận được tín hiệu mới chạy).

### Device

Device là một thành phần phần cứng, được kết nối với Sensor và/hoặc Actuators thông qua dây hoặc không dây hoặc thậm chí tích hợp các thành phần này. Để xử lý dữ liệu từ Sensor và để điều khiển Actuators, thông thường phải có phần mềm ở dạng Drivers. Driver trong kiến trúc của cho phép các phần mềm khác trên Device truy cập Sensors và Actuators. Nó đại diện cho khả năng đầu tiên sử dụng phần mềm để xử lý dữ liệu do Sensors tạo ra và để kiểm soát các Actuators ảnh hưởng đến môi trường vật lý. Do đó, Device là entry point của môi trường vật lý vào thế giới kỹ thuật số.

Các thiết bị hoặc là (i) khép kín (self-contained) hoặc (ii) được kết nối với một hệ thống khác, ví dụ: với IoT Integration Middleware.

### Gateway

Device thường được kết nối với Gateway trong trường hợp Device không có khả năng kết nối trực tiếp với các hệ thống khác, ví dụ: do không thể giao tiếp qua một giao thức cụ thể hoặc do các hạn chế kỹ thuật khác. Để giải quyết những vấn đề này, Gateway được sử dụng để bù đắp những hạn chế đó bằng cách cung cấp các công nghệ và chức năng cần thiết để translate giữa các giao thức khác nhau và bằng cách chuyển tiếp giao tiếp giữa Device và các hệ thống khác. Do đó, Gateway chịu trách nhiệm hỗ trợ các công nghệ và giao thức truyền thông cần thiết và translate dữ liệu nếu cần thiết.

Ví dụ, Device giao tiếp với Gateway thông qua giao thức IoT, như giao thức ZigBee (ZigBee is an IEEE standard 802.15.4 (a technical standard that defines operation of low-rate WPANs) hoặc MQTT (Message Queuing Telemetry Transport - open standard protocols that work for small devices). Khi Gateway được thông báo ở định dạng nhị phân độc quyền từ Device, Gateway sẽ dịch thông tin thành JSON hoặc XML và chuyển tiếp dữ liệu tới một hệ thống web. Tương tự, Gateway có thể dịch các lệnh thành các công nghệ truyền thông, giao thức và định dạng được hỗ trợ bởi Device tương ứng. Gateway cũng có thể thực thi một số chức năng xử lý dữ liệu, như tổng hợp dữ liệu, tùy thuộc vào khả năng xử lý của nó.

### IoT Integration Middleware

IoT Integration Middleware chịu trách nhiệm nhận dữ liệu từ các Device được kết nối để xử lý dữ liệu nhận được. Device có thể giao tiếp trực tiếp với IoT Integration Middleware nếu thiết bị hỗ trợ công nghệ giao tiếp thích hợp, chẳng hạn như WiFi, giao thức truyền tải tương ứng, như HTTP hoặc MQTT và định dạng tải trọng tương thích, như JSON hoặc XML. Mặt khác, Device giao tiếp qua Gateway với IoT Integration Middleware. Do đó, từ quan điểm chức năng, nó đóng vai trò là lớp trung gian tích hợp cho các loại Sensors, Actuators, Devices và Applications khác nhau. Ngoài ra nó có thể có nhiều chức năng khác như dashboard, rule engine, quản lý device, user… có thể được truy cập bằng API như API REST/HTTP.

### Application

Application đại diện cho các phần mềm sử dụng IoT Integration Middleware để yêu cầu dữ liệu Sensor hoặc kiểm soát các hành động vật lý từ Actuators. Ví dụ: một hệ thống phần mềm kiểm soát nhiệt độ của tòa nhà. Application trong kiến trúc mà đưa ra cũng có thể là một IoT Integration Middleware khác, tuỳ vào mục đích sử dụng.

## Vấn đề an ninh với IoT

Bảo mật là một trong những vấn đề lớn nhất với IoT. Trong nhiều trường hợp những cảm biến sẽ thu nhận nhiều vấn đề nhạy cảm và quan trọng – ví dụ như những gì bạn đang nói và làm trong nhà của mình. Giữ sự bảo mật là điều tối quan trọng đối với niềm tin của người tiêu dùng, nhưng cho đến nay vấn đề bảo mật của IoT vẫn cực kỳ kém. Quá nhiều thiết bị IoT thiếu những điều cơ bản về bảo mật như mã hóa dữ liệu trong quá trình sử dụng.

Các lỗ hổng trong phần mềm là một vấn đề quan trọng, nhiều thiết bị IoT không được cập nhật bảo trì thường xuyên vì vậy vấn đề sẽ khó để được phát hiện và giải quyết kịp thời. Tin tặc hiện đang tích cực nhằm mục tiêu vào các thiết bị IoT như bộ định tuyến và webcam vì sự thiếu bảo mật vốn có của chúng khiến chúng dễ dàng bị xâm nhập và biến thành các botnet khổng lồ.

Các nhà nghiên cứu đã tìm thấy 100.000 webcam có thể bị hack một cách dễ dàng, trong khi một số smartwatch kết nối internet dành cho trẻ em đã được tìm thấy có chứa các lỗ hổng bảo mật cho phép tin tặc theo dõi vị trí của người dùng, nghe lén các cuộc hội thoại hoặc thậm chí giao tiếp với người dùng.Khi chi phí cho một thiết bị thông minh trở nên không đáng kể, những vấn đề này sẽ chỉ trở nên phổ biến và khó chữa hơn.

IoT đã thu hẹp khoảng cách giữa thế giới kỹ thuật số và thế giới vật lý, điều đó có nghĩa là việc hack vào các thiết bị có thể gây ra hậu quả nguy hiểm trong thế giới thực. Việc đột nhập vào các cảm biến kiểm soát nhiệt độ trong nhà máy điện có thể làm cho các nhà khai thác đưa ra quyết định sai lầm và để lại những hậu quả vô cùng thảm khốc; hoặc ví dụ như kiểm soát một chiếc xe không người lái có thể gây nguy hiểm đến tính mạng con người và các phương tiện giao thông

Đã có nhiều cuộc tấn công bảo mật và quyền riêng tư vào các ứng dụng đã được triển khai trên toàn thế giới. Cuộc tấn công Mirai trong quý cuối năm 2016 được ước tính đã lây nhiễm khoảng 2,5 triệu thiết bị được kết nối với Internet và khởi động cuộc tấn công từ chối dịch vụ phân tán (DDoS) [6] . Sau Mirai, Hajime và Reaper là các cuộc tấn công botnet lớn khác đã phát động chống lại một số lượng lớn các thiết bị IoT[6]. Sau những cuộc tấn công như vậy thì hậu quả để lại là không thể đo đếm được.

## Mã độc IoT Botnet

### Giới thiệu chung

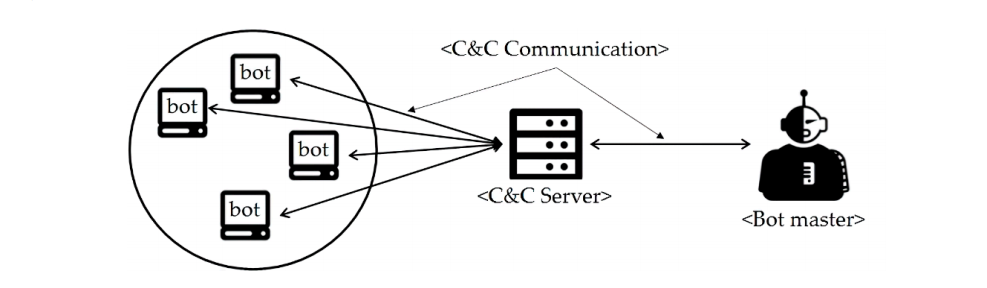
Botnet được gọi đầy đủ là “Bots network”, dùng để chỉ một mạng lưới các máy tính bị chi phối bởi đối tượng nào đó và bị điều khiển bởi một máy tính khác từ xa. Botnet là một phần mềm độc hại. Rất nhiều thiết bị hiện nay đang bị nhiễm bởi một Botnet nào đó mà chúng ta không phát hiện ra.

Botnet là một mạng nguy hiểm nó hoạt động dựa trên sự điều khiển bởi một mạng khác từ xa, chiếm và điều hành hoạt động của vật chủ đang bị nhiễm. Botnet được cài trên nhiều thiết bị, và nó nằm sẵn trong các thiết bị đó chỉ đợi lệnh từ Bot Herder (chương trình chỉ huy Botnet) để tấn công đến mục tiêu được chỉ định. Một hệ thống Botnet có thể từ 1.000 đến vài chục nghìn thiết bị đang bị nhiễm Bot.

Thiết bị bị nhiễm Botnet thường được gọi là các “Zombie” (xác sống) và sẽ bị chi phối và điều khiển bởi một Botmaster. Botmaster có thể cản trở hoạt động, gây gián đoạn mất nhiều thời gian, giảm năng suất hoạt động của thiết bị , hoặc chúng chiếm và sử dụng dữ liệu đánh cắp với mục đích riêng của hacker đứng đằng sau các mạng Botnet.

IoT Botnet là việc các Botnet xâm nhập và bị botmaster chiếm quyền điều khiển trên các thiết bị IoT, ví dụ như xe tự vận hành, các webcam hoặc cũng có thể là tivi, tủ lạnh, …., các thiết bị này sẽ bị botmaster sử dụng vào các tình huống cụ thể như đánh cắp dữ liệu hoặc sử dụng các thiết bị này làm công cụ để tấn công mạng.

### Các thành phần cơ bản của mạng IoT botnet

******

Hình 1.2. Các thành phần cơ bản của mạng IoT botnet[13]

Một mạng botnet điển hình bao gồm các bot, máy chủ điều khiển C&C (Command and Control), và botmaster. Trong đó:

* Bot là một thiết bị IoT đã bị lây nhiễm chương trình độc hại mà botmaster phát tán, cho phép botmaster ra lệnh và kiểm soát thiết bị nạn nhân. Các bot cũng thường được gọi là các zombie hoặc các drone. Botnet là một mạng lưới các bot.
* Botmaster là kẻ tấn công tinh vi được biết đến là đã thiết kế và điều khiển mạng botnet.
* Thông qua máy chủ C&C, botmaster có thể gửi các lệnh và nhận các phản hồi từ bot.

### Vòng đời của một mạng IoT Botnet cơ bản

Vòng đời của một mạng IoT Botnet là một chuỗi các giai đoạn kế tiếp nhau. Do đó, sự kết thúc của vòng đời, tức là thành công của cuộc tấn công, chỉ đạt được sau khi tất cả các giai đoạn trước đó đã được thực hiện thành công. Cụ thể, vòng đời này bao gồm năm giai đoạn: hình thành, tuyển dụng, tương tác, thực hiện tấn công và ẩn mạng[19].

#### Giai đoạn hình thành

Giai đoạn đầu tiên của vòng đời được là quá trình hình thành botnet. Điều quan trọng là phải hiểu lý do đằng sau việc tạo ra botnet, cũng như kiến trúc và thiết kế thông thường của chúng.

Giai đoạn hình thành botnet có thể được chia thành ba giai đoạn hoặc quy trình: động lực, thiết kế và thực hiện. Đầu tiên, một botmaster tiềm năng (người quản lý các bot) cần có lý do chính đáng để tạo botnet. Sau đó là quyết định về thiết kế và kiến trúc của mạng botnet. Kiến trúc này có thể là: tập trung, các bot được quản lý thông qua một máy chủ lệnh và điều khiển (C&C) duy nhất và được phân phối tới tất cả các bot của mạng botnet hoạt động đồng thời như máy chủ và máy khách, hoặc kết hợp, một hoặc nhiều mạng phân tán, mỗi mạng có một hoặc nhiều máy chủ C&C hơn.

#### Giai đoạn tuyển dụng

Các botmaster sẽ rò quét để tìm kiếm các lỗ hổng từ các thiết bị để biến chúng thành bot của mình . Sau khi các bot bị chiếm quyền điều khiển sẽ kích hoạt cơ chế tự lây lan , tự động rò quét ngẫu nhiên các địa chỉ IP trong mạng và xác định các thiết bị IoT có khả năng khai thác như sử dụng các cổng dịch vụ Telnet, SSH, FTP…

#### Giai đoạn tương tác

Giai đoạn này đề cập đến tất cả các tương tác được thực hiện trong quá trình hoạt động của mạng botnet. Một trong những điểm khác biệt chính giữa mạng botnet và các loại phần mềm độc hại khác là sự tồn tại của liên lạc bằng cách sử dụng máy chủ C&C. Các quá trình liên quan đến giai đoạn thứ ba này có thể được phân loại là nội bộ và bên ngoài.

* Tương tác nội bộ là những tương tác được thực hiện giữa các thành viên của mạng botnet, tức là từ quản trị viên bot tới các bot hoặc ngược lại, hoặc chỉ giữa các bot. Ở đây, được chia thành hai quy trình: đăng ký và giao tiếp C&C.
  + Quy trình đăng ký. Đăng ký là quá trình thông qua đó một máy chủ bị xâm nhập trở thành một phần của mạng botnet. Trong số tất cả các kỹ thuật được quan sát trong các mạng botnet được nghiên cứu khác nhau, tồn tại hai loại đăng ký: tĩnh và động. Trong đăng ký tĩnh, tất cả thông tin cần thiết để trở thành một phần của mạng botnet đều được mã hóa cứng. Thông thường, địa chỉ IP của máy chủ C&C được cung cấp trong mã của các bot. GT-Bot, Agobot và SDbot là một số ví dụ về các thức này.Trong đăng ký động, các bot phải yêu cầu rõ ràng thông tin cần thiết để trở thành một phần của botnet cho một bên thứ ba hoặc mạng trung lập. Một ví dụ về loại đăng ký này được sử dụng trong Phatbot, sử dụng các máy chủ bộ nhớ cache của Gnutella để tải xuống danh sách các bot trong cùng mạng
  + C&C Communications. Phần lớn các tương tác trong mạng botnet xảy ra sau khi quá trình đăng ký hoàn tất. Những tương tác này là giao tiếp C&C.Các bot sẽ định kỳ yêu cầu thông tin từ máy chủ C&C hoặc thụ động nhận các yêu cầu từ C&C đẩy về , các bot thực hiện phân tích kiến trúc thiết bị IoT này tải về các tập tin nhị phân mã độc phù hợp (MIPS, ARM, PowerPC,…).Các giao thức thường được sử dụng là IRC, HTTP và P2P
* Tương tác bên ngoài là những tương tác liên quan đến giao tiếp được thực hiện giữa một thành viên của mạng botnet và một máy chủ không bị xâm phạm. Chúng thường tương ứng với quyền truy cập vào các dịch vụ phổ biến được cung cấp trên Internet. Dịch vụ bên ngoài chính được botnet sử dụng là DNS. Trong mạng botnet tập trung, bot thường khởi chạy các truy vấn DNS để giải quyết địa chỉ IP của máy chủ C&C. Một tương tác bên ngoài phổ biến khác là việc sử dụng mạng P2P như một lớp trung gian để ẩn các giao tiếp C&C

#### Giai đoạn tấn công

Mục tiêu cuối cùng của botnet là thực hiện một cuộc tấn công. Tính năng chính của các cuộc tấn công botnet là số lượng lớn những kẻ tấn công tham gia vào quá trình này. Do đó, các cuộc tấn công chính do botnet phát động là:

* Từ chối dịch vụ phân tán (DDoS). Tấn công từ chối dịch vụ (DoS) là những nỗ lực nhằm ngăn chặn việc sử dụng hợp pháp một dịch vụ hoặc đơn giản là làm giảm tính khả dụng của dịch vụ đó
* Gửi thư rác. Email spam chứa một số thông tin nhất định được tạo ra để gửi đến một số lượng lớn người nhận
* Lừa đảo. Đây là một hoạt động gian lận được định nghĩa là hành vi lừa một người tiết lộ thông tin nhạy cảm . Điều này có thể được thực hiện bằng một bản sao của một trang web hiện có.
* Click gian lận. Nó bao gồm việc nhấp chuột vào quảng cáo trực tuyến hoặc truy cập vào các trang web nhất định với mục đích tăng doanh thu từ trang web của bên thứ ba hoặc làm cạn kiệt ngân sách của nhà quảng cáo

#### Giai đoạn ẩn mình

Cơ chế ẩn mình cũng được coi là một phần của vòng đời botnet . Đây là những cơ chế được thiết kế để ẩn botnet và khiến việc phát hiện ra các thành phần của nó (bot, botmaster, kênh C&C) rất khó khăn.

Có nhiều kỹ thuật ẩn có thể được phát hiện. Sau đây những phương pháp được sử dụng rộng rãi trong mạng botnet

* Mật mã. Về giai đoạn tương tác, giao tiếp C&C thường được mã hóa trong các mạng botnet hiện đại để ngăn chúng bị phân tích. Việc phát triển các kỹ thuật có trách nhiệm pháp lý để phát hiện các thông tin liên lạc C&C ngày càng trở nên khó khăn khi sử dụng mật mã
* Tính đa hình. Nó bao gồm việc tạo ra các phiên bản khác nhau của mã nguồn của một chương trình để thay đổi trong khi chức năng của nó vẫn còn. Kỹ thuật này làm cho quá trình phát hiện dựa trên chữ ký được sử dụng bởi hầu hết các công cụ chống vi-rút hiện tại trở nên khó khăn hơn
* Giả mạo IP. Nó bao gồm việc gửi các gói IP với một địa chỉ nguồn giả. Nó được sử dụng rộng rãi trong các cuộc tấn công DoS với mục đích tránh các bộ lọc IP.
* Giả mạo e-mail. Tương tự như giả mạo IP, giả mạo e-mail bao gồm việc gửi e-mail có địa chỉ người gửi giả
* Mạng thông lượng nhanh. Kỹ thuật này cho phép ẩn bất kỳ máy chủ cuối cùng nào trong mạng bằng một số lượng lớn proxy chuyển hướng các yêu cầu đến từ người dùng. Các proxy này thay đổi rất thường xuyên bằng cách sử dụng các mục nhập DNS có TTL thấp, do đó, việc theo dõi các liên lạc trở nên khó khăn. Kỹ thuật này có thể được sử dụng để ẩn một máy chủ C&C trung tâm trong mạng botnet.
* Bot cố gắng tìm kiếm các loại mã độc khác trên thiết bị để hủy hoặc xóa chúng ngay khi lây nhiễm thành công để đảm bảo tài nguyên bởi các thiết bị IoT là những thiết bị có tài nguyên hạn hẹp (ví dụ Mirai tìm và hủy các tiến trình của mã độc .anime và Qbot).

Hầu hết các IoT botnet đều thực hiện theo 1 quy trình tương tự, và đều yêu cầu các thông tin quan trọng như IP, URL, tên miền,… được gọi là PSI.

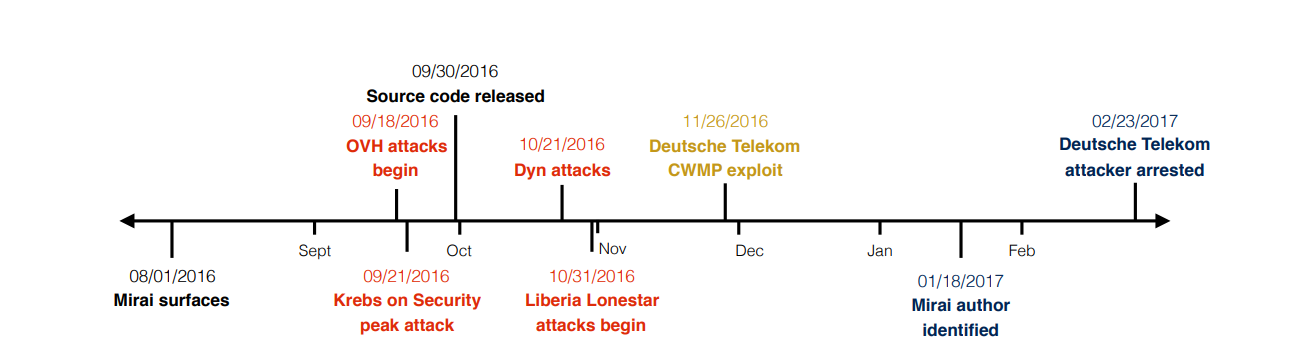
Kích thước của một mạng IoT botnet không giống nhau, từ một mạng lớn với khoảng hàng trăm nghìn bot cho đến mạng nhỏ chỉ tầm vài trăm thiết bị. Bất kể kích thước của chúng, các mạng IoT botnet này đều có chung mục đích, nó không chỉ gây mất an toàn thông tin cho hạ tầng Internet mà còn tham gia vào các loại mối đe dọa và tấn công khác phụ thuộc vào mục đích của hacker và mục tiêu tấn công.

### Ví dụ về mã độc IoT Botnet (Mirai Botnet)

#### Giới thiệu

Mirai là phần mềm độc hại lây nhiễm các thiết bị thông minh chạy trên bộ xử lý ARC, biến chúng thành một mạng lưới các bot được điều khiển từ xa hoặc “zombie” , thường được sử dụng để khởi chạy các cuộc tấn công DDoS .

#### Những sự kiện liên quan đến Mirai botnet



Hình 1.3. Dòng sự kiện liên quan đến Mirai botnet[16]

Các báo cáo sớm nhất về Mirai xuất hiện vào ngày 31 tháng 8 năm 2016 , mặc dù phải đến giữa tháng 9 năm 2016 Mirai mới thu hút sự quan tâm bằng các cuộc tấn công DDoS lớn nhắm vào Krebs on Security và OVH. Vào đầu năm 2017, các tác giả viết ra mã nguồn Mirai bị phát hiện .Sau đó mã nguồn của Mirai được phát hiện trên trnag web hackforums.net, điều này đã làm cho các biến thể của Mirai gia tăng một cách nhanh chóng. Một sự kiện đáng chú ý về biến thể của Mirai được bổ sung hỗ trợ cho việc khai thác bộ định tuyến thông qua CPE , giao thức quản lý WAN (CWMP), dựa trên HTTP giao thức cho phép tự động cấu hình và quản lý từ xa bộ định tuyến, modem và các thiết bị agent khác (CPE). Việc khai thác này đã dẫn đến sự cố ngừng hoạt động tại Deutsche Telekom vào cuối tháng 11 năm 2016 và đến tháng 2 năm 2017 kẻ tình nghi đã bị bắt[16].

#### Cơ chế lây lan của mã độc Mirai Botnet

Bước đầu tiên, mã độc sẽ dò quét cổng SSH22/TELNET23 hoặc 2323 của những địa chỉ IPv4 ngẫu nhiên nhưng nằm trong dải IP được xác định trước. Mirai dò quét nhanh bằng cách gửi các gói tin TCP SYN thăm dò tới cổng TELNET. Nếu phát hiện ra cổng TELNET của thiết bị mở, Mirai sẽ cố gắng khởi tạo phiên TELNET bằng cách sử dụng ngẫu nhiên 10/62 tài khoản mặc định trên các dòng thiết bị IoT mà nó biết.

Tiếp theo, nếu khởi tạo phiên thành công, Mirai gửi thông tin (IP, tài khoản) về cho máy chủ báo cáo của kẻ tấn công. Trong các chu trình dò quét lần sau, những IP đã tồn tại trên máy chủ báo cáo sẽ không được quét nữa.

Sau khi nhận được thông tin, máy chủ báo cáo gửi một chương trình nạp (loader) tới thiết bị IoT. Chương trình nạp sẽ xác định chủng loại, firmware, OS…của thiết bị. Từ những thông tin lấy được, chương trình nạp sẽ tải về loại mã độc tương ứng để có thể chạy trên thiết bị này. Mirai mất khoảng 98 giây để thực hiện hành động lây nhiễm cho 1 thiết bị IoT.

Mã độc sau khi chạy trên thiết bị thì sẽ xóa dấu vết bằng cách xóa file tải về và “thay tên đổi họ” tiến trình chạy trong bộ nhớ chính của thiết bị. Vì đã xóa file tải về nên khi thiết bị reboot, các thông tin trong bộ nhớ chính bị xóa, mã độc sẽ không hoạt động trên thiết bị nữa. Để “củng cố địa vị”, Mirai kill các tiến trình sử dụng cổng TCP22/23 và cũng kill các tiến trình của các mã độc khác như .anime hay .Qbot, mặc dù đây cũng là các biến thể của Mirai.

Sau khi mã độc chạy, thiết bị IoT giờ đã trở thành một bot hay Zoombie trong mạng botnet Mirai, và có nhiệm vụ nhận lệnh từ máy chủ C2 (Command & Control) để tấn công DDoS. Vì là chu trình lây lan nên từ thiết bị này, mã độc tiếp tục dò quét cổng SSH22/TELNET23/2323 của các thiết bị IoT khác.



Hình 1.4. Cơ chế lây lan của Mirai botnet

## Các phương pháp phát hiện mã độc IoT Botnet

Phân tích mã độc là một bước quan trọng trong để có thể ngăn chặn và tiêu diệt hoàn toàn mã độc ra khỏi máy tính và hệ thống mạng; khôi phục lại hiện trạng của mạng như ban đầu; truy tìm nguồn gốc tấn công.

Hầu hết mã độc ở dạng các chương trình, dịch vụ (dạng binary) không thể đọc thông thường. Cách phân tích mã độc chính là: sử dụng các công cụ và các kĩ thuật phân tích. Có hai kĩ thuật phân tích chính:

* Phân tích tĩnh (Phân tích mà không cần mã độc chạy trong hệ thống)
* Phân tích động (Phân tích mã độc chạy thật trong hệ thống).

### Phương pháp phân tích động

Phân tích động là phương pháp xây dựng môi trường phân tích cách ly với hệ thống thật. Môi trường đó được xây dựng các hệ thống để mô phỏng các dịch vụ Internet và xây dựng các môi trường phù hợp để mã độc có thể chạy được. Cho phép người phân tích quan sát xem một tệp mẫu nghi ngờ khi thực thi sẽ làm những gì ?, chạy như thế nào ?, thực hiện những hành động gì trên máy tính, hoặc tương tác với các dịch vụ mạng thông qua các công cụ theo dõi.

Một số công trình nghiên cứu phát hiện mã độc IoT botnet dựa trên phương pháp phân tích tĩnh được [17] kể đến như:

Celeda và cộng sự [18] đã đề xuất phương pháp hiện mã độc Chuck Norris Botnet phát hiện mã độc lây lan thông qua giao thức telnet do các thiết bị sử dụng mật khẩu yếu hoặc mặc định của nhà sản xuất trên các thiết bị mô-đem bị lây nhiễm. Tuy nhiên nghiên cứu chỉ áp dụng được trên kiến trúc MIPS. Trong khi IoT đa kiến trúc vi xử lý khác như ARM, PowerPC.

Jonas và cộng sự [15] đề xuất phương pháp phát hiện mã độc sử dụng framework Avatar, trong đó thực hiện phân tích firmware các thiết bị nhúng bằng cách tiêm vào một phần mềm trung gian đặc biệt để mô phỏng mã độc trong quá trình thực thi. Tuy nhiên quá trình mô phỏng chậm hơn nhiều so với quá trình thực thi trên thiết bị thực do việc đồng bộ không đảm bảo tốc độ truyền tin.

Cùng hướng tiếp cận đó, Yin Minn Pa Pa và cộng sự [24] đã đề xuất phương pháp với IoT honeypot để chặn bắt mã độc IoT dựa trên giao thức telnet và IoTBOX để phân tích mã độc IoT đa kiến trúc vi xử lý. Nhưng phương pháp này chỉ tập trung vào phân tích các hành vi mạng.

Ahmad Darki và cộng sự [2] đã đề xuất phương pháp RARE sử dụng phân tích tĩnh để cung cấp các thông tin cho quá trình phân tích động, từ đó tùy chỉnh môi trường mô phỏng giúp mã độc có thể bộc lộ hết tất cả các hành vi độc hại, kết quả đạt 94% các mẫu mã độc có thể kích hoạt thành công. Tuy nhiên, các đặc trưng thu thập qua phân tích tĩnh còn đơn giản, chỉ thu thập được địa chỉ IP và tên miền, và quá trình tương tác giữa bot và máy chủ C&C chưa đầy đủ khi chưa thể tùy chỉnh được máy chủ C&C.

Từ một số nghiên cứu được đề cập cho thấy nhược điểm của phân tích động là chỉ cho phép phân tích đơn luồng và không thể quan sát tất cả các khả năng thực thi của mã độc. Đồng thời kiến trúc vi xử lý của các thiết bị IoT rất đa dạng (MIPS, ARM, PowerPC…) nên đòi hỏi việc xây dựng môi trường thực thi đảm bảo cho các thiết bị IoT hoạt động để thu thập dữ liệu làm đầu vào cho quá trình phân tích là rất phức tạp [17].

### Phương pháp phân tích tĩnh

Phân tích tĩnh là phương pháp phân tích bằng cách đọc mã thực thi của mã độc sử dụng một bộ phân tích (Disassembler) để dịch ngược các đoạn mã bên trong một chương trình mã độc thành dạng hợp ngữ, từ đó tìm hiểu các chỉ lệnh nhằm biết chính xác chương trình mã độc có thể làm những việc gì. Những chỉ lệnh được thực thi bởi CPU, do vậy những chỉ lệnh này sẽ cho người phân tích biết chính xác những gì chương trình mã độc thực hiện. Tuy nhiên để có thể thực hiện được việc phân tích tĩnh, đòi hỏi người phân tích phải am hiểu sâu về hợp ngữ, về các mã chỉ dẫn lệnh và các khái niệm và các hàm API trong hệ điều hành.

Hướng tiếp cận này sử dụng những kỹ thuật như đồ thị luồng điều khiển (CFG), đồ thị luồng dữ liệu (DFG), thực thi tượng trưng (SE) để phân tích các đặc trưng phát hiện mã độc như API, Opcode, CFG, DFG, PSI, FLF.

Công trình nghiên cứu phát hiện mã độc IoT botnet dựa trên phương pháp phân tích tĩnh được [17] kể đến là của Costin và cộng sự [1], họ đã đề xuất một framework để thu thập, lọc, bóc tách và phân tích tĩnh firmware quy mô rộng, từ đó phát hiện lỗ hổng bảo mật, mã độc. Tuy nhiên nghiên cứu này chỉ dựa trên các đặc trưng rời rạc mà không đi vào khai thác sự tương tác, liên quan giữa các đặc trưng… Trong khi đó, mã độc IoT Botnet luôn có quy trình hoạt động khá tương đồng nhau và có sự tương tác với nhau.

Có thể thấy ưu điểm chính của phương pháp phân tích tĩnh là có khả năng khai thác được tất cả các đường dẫn thực thi có thể có trong các mẫu phần mềm độc hại, giúp có một cái nhìn tổng quan các khả năng có thể xảy ra trong tập tin thực thi. Tuy nhiên nhược điểm của phương pháp này không phát hiện được các mẫu mã độc phức tạp (obfuscate) và đa hình [11].

## Tổng kết chương

Chương này đã giới thiệu về mã độc cũng như là những nguy hiểm mà nó mang lại. Nhiều công trình nghiên trong phân tích phát hiện mã độc IoT botnet bao gồm phân tích tĩnh, phân tích động ra đời, tuy nhiên các cách tiếp cận hiện nay vẫn còn nhiều hạn chế . Việc phát hiện mã độc dựa trên phân tích các đặc trưng dưới dạng đồ thị (graph-based) đem lại hiệu quả hơn so với phân tích đặc trưng dưới dạng luồng (flow-based), do dạng đồ thị có thể thể hiện được mối quan hệ chặt chẽ giữa dữ liệu và giảm thời gian so sánh chéo trong mô hình.

Trong khi đó với những thành quả vượt bậc trong sử dụng machine learning trong những năm gần đây.Việc kết hợp phân tích tĩnh sử dụng đồ thị đặc trưng kết hợp với học củng cố để phân tích đường đi là hành vi của mã độc là một cách tiếp cận đầy hứa hẹn.

# Xây dựng thuật toán trích chọn đặc trưng bằng phương pháp học củng cố và áp dụng để phát hiện mã độc

*Từ những hạn chế từ những phương pháp phân tích, phát hiện mã độc IoT Botnet, Chương 2 tập trung giới thiệu phương pháp đề xuất kết hợp giữa đồ thị đặc trưng có cấu trúc PSI và phương pháp học củng cố để phân tích trích chọn đặc trưng và phát hiện mã đôc.*

## Giới thiệu về đồ thị PSI

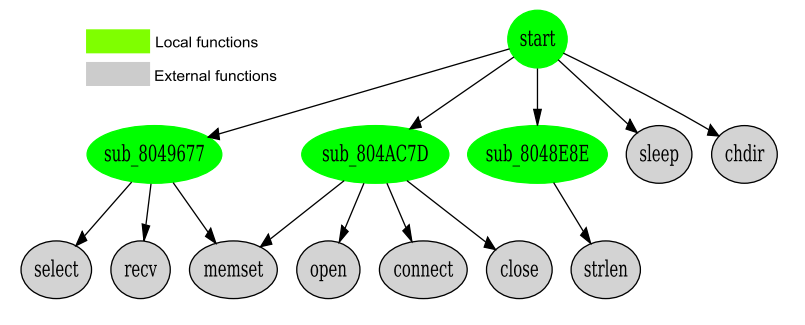
Đồ thị PSI là đồ thị có cấu trúc thể hiện luồng hoạt động của các hàm trong file ELF , các hàm này là các hàm có chứa thông tin có thể liên quan đến các hành vi của mã độc .Để xây dựng đồ thị PSI từ các tập tin ELF ta cần thực hiện 2 bước là xây dựng đồ thị hàm gọi và xây dựng đồ thị PSI [9]

### Xây dựng đồ thị hàm gọi

Đồ thị hàm gọi (FCG) là một đồ thị luồng điều khiển, biểu thị mối quan hệ giữa các chương trình con trong một chương trình. Đồ thị gọi hàm được định nghĩa là một đồ thị có hướng G = (V, E), bao gồm tập đỉnh V và tập cạnh E, trong đó đỉnh trong đồ thị tương ứng với hàm có trong chương trình, và edge có nghĩa là mối quan hệ người gọi-người gọi giữa hai hàm [23]. Có hai loại hàm được sử dụng trong một chương trình, được đặt tên là hàm cục bộ và hàm bên ngoài. Chức năng cục bộ được thực thi và được viết bởi người viết để thực hiện một nhiệm vụ cụ thể; trong khi các hàm bên ngoài là các hàm hệ thống hoặc thư viện do tác giả viết, các hàm cục bộ có nhiều tên khác nhau trong các chương trình khác nhau mặc dù chúng có chức năng tương tự; đồng thời, tên của các chức năng bên ngoài nhất quán trong tất cả các chương trình. Hơn nữa, trái với các hàm cục bộ, các hàm bên ngoài không bao giờ gọi bất kỳ hàm cục bộ nào bởi vì chúng đều là từ hệ thống hoặc thư viện.

Với một tệp thực thi, trước tiên, ta sử dụng mã nhận dạng trình đóng gói có tên DiE — Detect It Easy [3] —để kiểm tra xem tệp có được đóng gói hay không và loại công cụ đóng gói nào được sử dụng. Sau đó, nếu điều này được xác định, một số trình giải nén được sử dụng để giải nén tệp tương ứng, chẳng hạn như UPX [21]. Các tệp thực thi không thể được giải nén bằng các công cụ này không được xem xét trong cách tiếp này. Sau bước giải nén, ta sử dụng IDA Pro —một trình gỡ và gỡ lỗi nổi tiếng dành cho các tệp nhị phân của Linux, MacOS và Windows để tháo rời mẫu. Ở bước này, ta sử dụng IDAPython để tạo một trình cắm thêm trích xuất mã lắp ráp của nhị phân và một tập hợp các chức năng đã xác định. Khi thông tin liên quan này đã được trích xuất, đồ thị hàm gọi sẽ được xây dựng.

Như trong Shang et al. [20], ta sử dụng tìm kiếm theo chiều rộng đầu tiên để xây dựng đồ thị hàm gọi. Từ các hàm điểm đầu vào, các mối quan hệ người gọi-bộ nhớ giữa hai hàm được xác định và thêm vào biểu đồ. Một đồ thị bao gồm các hàm địa phương và các hàm bên ngoài được thể hiện



Hình 2.1. Một phần của biểu đồ gọi hàm của mẫu Linux.Mirai[9]

### Xây dựng đồ thị PSI

Đồ thị hàm gọi là một biểu diễn dữ liệu cho luồng điều khiển của một chương trình. Loại biểu đồ này có thể hiện các luồng cho mọi lần chạy chương trình có thể xảy ra. Đồ thị cuộc gọi thường phức tạp với số lượng lớn các nút và cạnh, do đó, mất nhiều thời gian hơn để tính toán và nhiều bộ nhớ hơn để lưu trữ. Mặc dù mọi mối quan hệ cuộc gọi của một chương trình được biểu diễn trong biểu đồ, có thể một số mối quan hệ cuộc gọi sẽ không bao giờ xảy ra trong các lần chạy thực tế của chương trình. Để giảm thiểu sự phức tạp của tính toán, ta sẽ xây dựng PSI-Graph bằng cách chọn các chức năng và mối quan hệ gần với hoạt động của các bước của IoT botnet.

Trước khi xây dựng PSI-Graph, trước tiên, phương pháp sẽ trích xuất tất cả các PSI hiện có bên trong tệp nhị phân bằng một trình cắm thêm của IDA Pro. Các PSI này có thể được mã hóa và thường chứa thông tin ngữ nghĩa quan trọng có thể tiết lộ ý định của kẻ tấn công. Ví dụ, chuỗi “/ dev / watchdog; / dev / misc / watchdog ”luôn tồn tại trong Linux.Mirai, thể hiện rằng botnet cố gắng ngăn quá trình khởi động lại thiết bị. Trong một số trường hợp, các chuỗi đã trích xuất được làm xáo trộn hoặc mã hóa bằng cách sử dụng các cơ chế làm xáo trộn giống nhau. Cuối cùng, ta sử dụng một thuật toán đơn giản để tạo PSI-Graph cho mỗi mẫu thực thi.

Sau khi có được PSI của các mẫu thực thi, ta xây dựng đồ thị PSI đại diện cho từng mẫu thực thi. Xem xét sự cân bằng giữa độ chính xác của phân loại và độ phức tạp tính toán, ta chọn các hàm chứa PSI có độ dài ít nhất ba ký tự. Rõ ràng là đồ thị PSI đơn giản hơn nhiều so với đồ thị hàm gọi. Trung bình, một PSI-Graph chỉ chứa khoảng 16 đỉnh và 60 cạnh, so với 156 đỉnh và 360 cạnh của đồ thị hàm gọi.

Tuy nhiên đồ thị PSI vẫn còn quá nhiều đỉnh và cạnh có thể gây ra nhầm lẫn trong quá trình training . Vì vậy việc phân tích tìm ra đường đi thực sự là hành vi của mã độc là điều rất cần thiết . Hiện nay việc sử dụng học củng cố để phân tích đường đi vô cùng phổ biến và đạt được nhiều thành công rực rỡ , vì vậy sử dụng học củng cố để phân tích đường đi là luồng mà mã độc sử dụng để tấn công các thiết bị IoT là một giải cách tiếp cận mới đầy hứa hẹn.

## Giới thiệu học củng cố

### Giới thiệu

Học củng cố (reinforcement learning) là một lĩnh vực con của học máy (machine learning) bên cạnh học giám sát và học không giám sát, nghiên cứu cách thức một tác nhân (agent) trong một môi trường (environment) nên chọn thực hiện các hành động (action) nào để cực đại hóa một khoản thưởng (reward) nào đó về lâu dài. Các thuật toán học củng cố cố gắng tìm một chính sách (policy) ánh xạ các trạng thái (state) của thế giới tới các hành động mà tác nhân nên chọn trong các trạng thái đó.

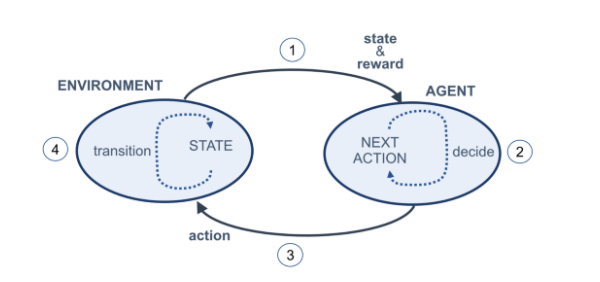
Học củng cố đã đạt nhiều thành tựu trong quãng đời đang phát triển của nó. Nổi bật nhất chính là sự kiện AlphaGo của DeepMind đánh bại Lee Sedol – một trong những kỳ thủ mạnh nhất trong lịch sử cờ vây.

Cách tiếp cận của học củng cố tập trung nhiều hơn vào việc học từ sự tương tác. Thực thể học không được cho biết những hành động cần thực hiện, mà thay vào đó phải tự khám phá xem hành động nào tạo ra phần thưởng lớn nhất - mục tiêu của nó, bằng cách tự kiểm tra chúng bằng cách “thử và sai”. Hơn nữa, những hành động này có thể ảnh hưởng không chỉ đến phần thưởng trước mắt mà còn ảnh hưởng đến phần thưởng trong tương lai (phần thưởng bị trì hoãn) vì các hành động hiện tại sẽ quyết định các tình huống trong tương lai. Hai đặc điểm này, “tìm kiếm thử và sai” và “phần thưởng bị trì hoãn”, là hai đặc điểm quan trọng của học củng cố.

Trong học củng cố, có 2 thành phần quan trọng:

* Tác nhân (agent) đại diện cho “giải pháp”, là một chương trình máy tính với một vai trò duy nhất là đưa ra quyết định (hành động) để giải quyết các vấn đề.
* Môi trường (environment) đó là đại diện của “vấn đề”, là mọi thứ xảy ra sau quyết định của tác nhân. Môi trường phản ứng với những hệ quả của hành động, đó là trạng thái và phần thưởng.

Một chu kỳ học củng cố bắt đầu với việc Tác nhân quan sát môi trường (bước 1) và nhận về trạng thái và phần thưởng. Tác nhân sử dụng trạng thái và phần thưởng này để quyết định hành động tiếp theo cần thực hiện (bước 2). Sau đó, Tác nhân sẽ gửi hành động đó tới môi trường nhằm cố gắng kiểm soát nó theo cách có lợi (bước 3). Cuối cùng, môi trường chuyển đổi và trạng thái bên trong của nó thay đổi do hệ quả của trạng thái trước đó và hành động của Tác nhân (bước 4). Sau đó, chu kỳ lặp lại. Mỗi chu kỳ này liên tục được lặp lại và Tác nhân ngày càng học được những chiến lược phù hợp. Khi thuật toán hội tụ, Tác nhân xây dựng xong chính sách tối ưu và có khả năng giải quyết vấn đề cần thực hiện.



Hình 2.2. Chu kỳ học củng cố

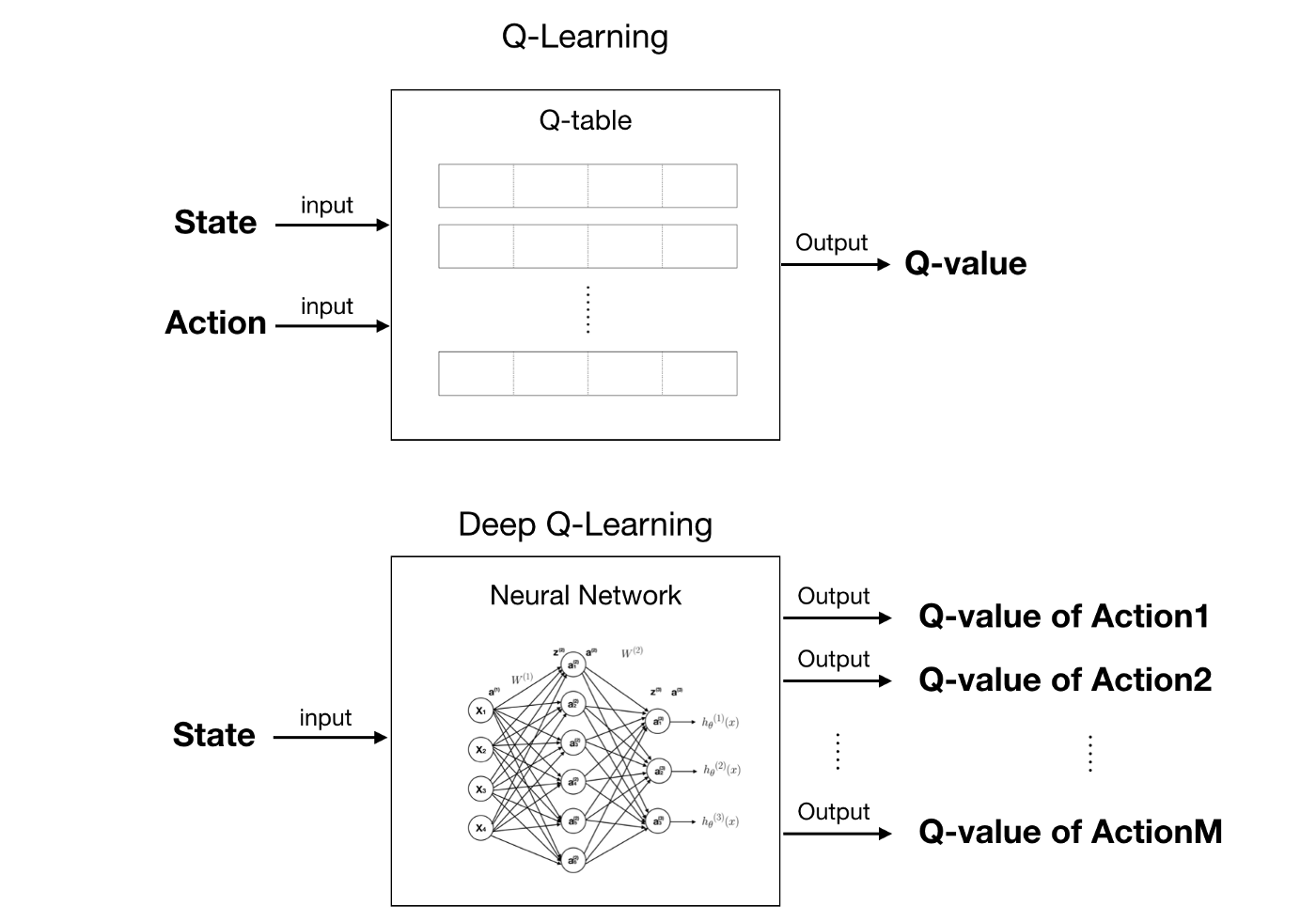
### Một vài thuật toán học củng cố hay được sử dụng

#### Q-learning

Q-learning là một thuật toán học củng cố không mô hình. Mục tiêu của Q-learning là học một chính sách, chính sách cho biết máy sẽ thực hiện hành động nào trong hoàn cảnh nào. Q-learning tìm một chính sách tối ưu theo nghĩa là nó tối đa hóa giá trị mong đợi của tổng số phần thưởng trên bất kỳ và tất cả các bước tiếp theo, bắt đầu từ trạng thái hiện tại. Q-learning có thể xác định một chính sách lựa chọn hành động tối ưu cho bất kỳ trường hợp cụ thể nào, với thời gian thăm dò vô hạn và chính sách một phần ngẫu nhiên. "Q" đặt tên theo tên hàm phần thưởng trả về, được sử dụng để cải thiện hoạt động và có thể nói là đại diện cho "chất lượng" của một hành động được thực hiện trong một trạng thái nhất định.

#### Deep Q-learning

Với phương thức ma trận, không gian chúng ta cần là n trạng thái với m hành động. Khi những con số này lớn như 1000 trạng thái và 1000 hành động trên mỗi trạng thái, chúng ta sẽ cần một ma trận gồm 1 triệu phần tử. Do đó, phương pháp ma trận rất khó sử dụng trong hầu hết các vấn đề thú vị như cờ vua, cần 10²⁰ không gian trạng thái. Thay vì ghi chính xác giá trị Q bằng ma trận, chúng ta sẽ sử dụng Mạng nơ-ron (NN) để tính toán xấp xỉ.



Hình 2.3. Mô hình hoạt động của Q-learning và Deep Q-learning

#### Double Q-learning

Double Q-Learning ( H. van Hasselt 2010) đã được đề xuất để giải quyết vấn đề đánh giá quá cao giá trị hành động (Q-value) trong Q-Learning cơ bản.Vấn đề đánh giá quá cao là tác nhân làm cho agent luôn chọn hành động không tối ưu trong bất kỳ trạng thái nhất định nào chỉ vì nó có giá trị Q tối đa. Trong Q-learning cơ bản, chính sách tối ưu của agent là luôn chọn hành động tốt nhất trong bất kỳ trạng thái nào. Giả định đằng sau ý tưởng là hành động tốt nhất có giá trị Q dự kiến tối đa. Tuy nhiên, Tác nhân không biết gì về môi trường lúc đầu, nó cần ước tính Q(s, a) lúc đầu và cập nhật chúng ở mỗi lần lặp. Các giá trị Q như vậy có rất nhiều nhiễu và không bao giờ chắc chắn liệu hành động có giá trị Q dự kiến tối đa có thực sự là tốt nhất hay không.

Thật không may, hành động tốt nhất thường có các giá trị Q nhỏ hơn so với một hành động không tối ưu trong hầu hết các trường hợp. Theo chính sách tối ưu trong Q-Learning cơ bản, Tác nhân có xu hướng thực hiện hành động không tối ưu ở bất kỳ trạng thái nào chỉ vì nó có giá trị Q tối đa. Vấn đề như vậy được gọi là sự đánh giá quá cao của giá trị hành động (Q-value).

Khi sự cố như vậy xảy ra, nhiễu giá trị Q ước tính sẽ gây ra sai lệch lớn trong quy trình cập nhật. Hậu quả là quá trình học tập sẽ rất phức tạp và lộn xộn.

Vì Q (s, a) tốt nhất được ước tính bằng cách sử dụng Q (s, a) hiện tại, rất nhiễu. Sự khác biệt giữa Q (s, a) tốt nhất và hiện tại trong hàm mất mát cũng rất lộn xộn và chứa những positive bias gây ra bởi những nhiễu khác nhau. Những positive bias tác động rất lớn đến thủ tục cập nhật.

Hơn nữa, nếu nhiễu của tất cả các giá trị Q có phân phối đồng đều, cụ thể hơn, các giá trị Q được đánh giá quá cao, thì đánh giá quá cao không phải là vấn đề vì các nhiễu này không ảnh hưởng đến sự khác biệt giữa Q (s ', a ) và Q (s, a).

Double Q-Learning sử dụng hai hàm giá trị hành động khác nhau, Q và Q’, làm công cụ ước tính. Ngay cả khi Q và Q’ bị nhiễu, những nhiễu này có thể được xem là phân phối đồng đều. Do đó, thuật toán này giải quyết vấn đề đánh giá quá cao giá trị hành động.

#### Double Deep Q Network

Triển khai Double Q-Learning với Mạng nơ-ron được gọi là Mạng lưới Deep Deep Q (Double DQN). Double DQN được đề xuất trong H. van Hasselt, 2016. Lấy cảm hứng từ Double Q-Learning, Double DQN sử dụng hai Mạng nơ-ron khác nhau, Mạng lưới Deep Q (DQN) và Mạng đích.

## Giới thiệu về mạng nơ-ron hồi quy phân loại

### Giới thiệu

Mạng nơ-ron hồi quy (Recurrent Neural Network - RNNs) là một loại mạng nơ-ron mạnh mẽ và thuộc về một trong những thuật toán đầy hứa hẹn nhất hiện tại, bởi vì chúng là những kiến trúc duy nhất có bộ nhớ bên trong.

RNNs đã có tương đối lâu, giống như nhiều thuật toán deep learning khác. Chúng được tạo ra ban đầu vào những năm 1980, nhưng chỉ cho thấy tiềm năng thực tiễn vào kể từ một vài năm gần đây, nhờ vào sự gia tăng sức mạnh tính toán, khối lượng dữ liệu khổng lồ mà chúng ta có ngày nay và sự phát minh của LSTM vào những năm 1990.

Với bộ nhớ bên trong chúng, RNNs có khả năng nhớ những sự kiện quan trọng về đầu vào chúng vừa nhận, qua đó cho phép chúng trở nên chính xác trong dự đoán sự kiện tiếp xảy ra tiếp theo.

Đây là lý do tại sao RNNs là thuật toán thích hợp cho các chuỗi dữ liệu như thời gian, giọng nói, văn bản, dữ liệu tài chính, âm thanh, video, thời tiết, và nhiều loại dữ liệu hơn nữa bởi vì chúng có thể thực hiện hiểu sâu những chuỗi này cùng với bối cảnh của nó hơn nhiều thuật toán khác.

RNNs tạo ra kết quả dự đoán trong chuỗi dữ liệu mà nhiều thuật toán không thể xử lý.

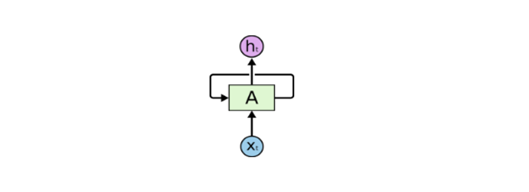
Ngày nay, RNNs được sử dụng trong nhiều phần mềm, chẳng hạn như Siri của Apple hay Google Dịch. RNNs có thể được ứng dụng trong:

* Mô hình hóa ngôn ngữ và tạo văn bản
* Dịch máy
* Nhận dạng giọng nói
* Tạo mô tả hình ảnh

### Kiến trúc mạng

#### Sơ đồ mạng

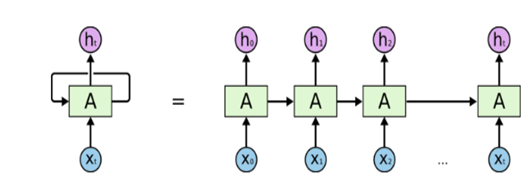
Mạng nơ-ron hồi quy là mạng với những vòng lặp (hồi quy) ở trong nó, cho phép thông tin có thể tiếp tục tồn tại.



Hình 2.4. Mạng nơ-ron hồi quy với khuyên tại nơ-ron

Trong sơ đồ ở trên, một phần mạng nơ-ron A nhận dữ liệu từ input xt và đưa ra giá trị ht. Một vòng lặp cho phép thông tin có thể được đưa qua nhiều bước xử lý của mạng. Một mạng nơ-ron quy hồi có thể được xem như nhiều bản mạng giống nhau, bản này truyền dữ liệu đến bản kế tiếp.

Xem xét khi chúng ta mở vòng lặp ra:

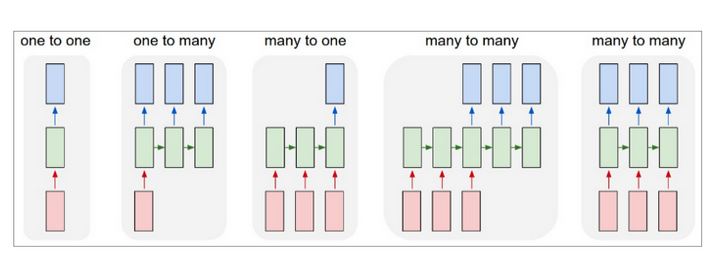


Hình 2.5. Sơ đồ mạng nơ-ron hồi quy

Bản chất chuỗi của kiến trúc này tương tự như các dãy hay danh sách. Kết quả của mỗi A sẽ được tái sử dụng trong bước tiếp theo cùng với input của bước đó để tạo ra kết quả cuối cùng. RNNs kết hợp vectơ input với vectơ trạng thái bằng một hàm cố định (nhưng có thể huấn luyện) để tạo ra vectơ trạng thái mới. Chúng chính là kiến trúc tự nhiên để xử lý những dữ liệu như thế.

Sự thuân tiện của RNNs trong việc xử lý chuỗi tự nhiên là chúng cho phép chúng ta xử lý chuỗi các vectơ: Chuỗi ở input, output hay ở cả hai.

Cụ thể:



Hình 2.6. Một cách phân loại các mạng nơ-ron hồi quy

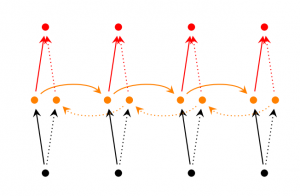
Mỗi hình chữ nhật là một vectơ và mũi tên thể hiện hàm. Input màu đỏ, output màu xanh dương và màu xanh lá thể hiện trạng thái mạng. Từ trái qua phải:

* Chế độ xử lý Vanilla: input cố định và output cố định (VD: phân loại ảnh)
* Output là chuỗi vectơ (VD: đặt tiêu đề ảnh lấy một bức ảnh và đưa ra một chuỗi từ)
* Input là chuỗi vectơ (VD: phân tích ngữ nghĩa để phân loại một câu cho trước là khẳng định hay phủ định)
* Input và output đều là chuỗi vectơ (VD: Dịch Máy: một RNN đọc một câu bằng tiếng Anh và chuyển sang tiếng Việt)
* Input và output đều là chuỗi vectơ nhưng được đồng bộ (VD: phân loại video và chúng ta muốn gắn nhãn mỗi khung hình của video)

Trong mọi trường hợp, đều không có hạn chế đối với độ dài chuỗi bởi vì hàm chuyển hồi quy (màu xanh lá) là cố định và có thể được áp dụng nhiều lần tùy ý.

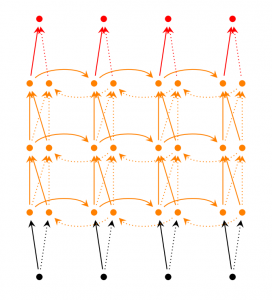
#### Một số mạng nơ-ron thường gặp

* RNNs hai chiều (bidirectional RNNs): dựa trên ý tưởng rằng đầu ra tại thời điểm t có thể không chỉ phụ thuộc vào các phần tử trước đó trong chuỗi, mà còn phụ thuộc vào các phần tử nằm sau. Ví dụ: để dự đoán từ bị thiếu trong chuỗi, chúng ta cần xem xét cả ngữ cảnh bên trái và bên phải. RNNs hai chiều khá đơn giản. Chúng chỉ là hai RNNs xếp chồng lên nhau. Đầu ra sau đó được tính toán dựa trên trạng thái ẩn của cả hai RNNs.



Hình 2.7. Sơ đồ RNN hai chiều

* RNNs sâu (hai chiều) (Deep (bidirectional) RNNs) tương tự như RNNs hai chiều, nhưng có nhiều lớp cho mỗi bước thời gian. Trong thực tế, điều này cho chúng ta khả năng huấn luyện cao hơn (nhưng cũng cần rất nhiều dữ liệu đào tạo).



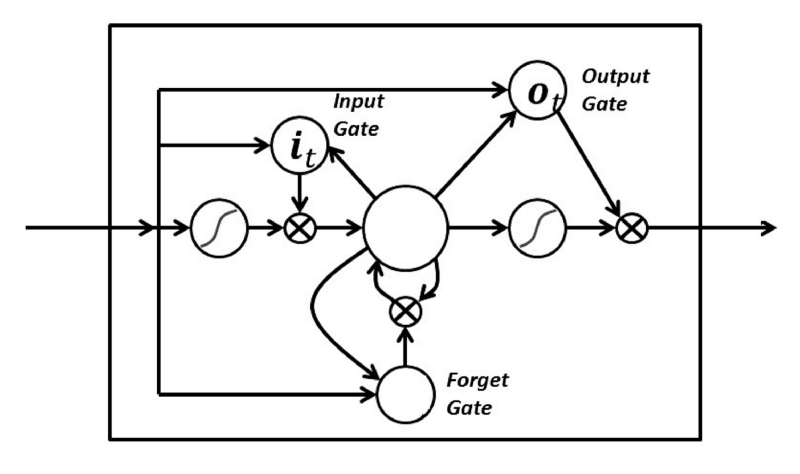
Hình 2.8. Sơ đồ RNN sâu 2 chiều

* Mạng LSTM (Long Short-Term Memory):

LSTM cho phép RNNs nhớ những yếu tố đầu vào của nó trong một thời gian dài. Điều này là do LSTM chứa thông tin của nó trong một bộ nhớ, giống như bộ nhớ của máy tính và LSTM có thể đọc, viết và xóa thông tin khỏi bộ nhớ của nó.

Bộ nhớ này có thể được xem như là một tế bào có cổng, trong đó cổng có nghĩa là tế bào quyết định có lưu trữ hay xóa thông tin (nếu nó mở cổng hay không) dựa trên tầm quan trọng mà nó gán cho thông tin. Việc gán tầm quan trọng diễn ra thông qua trọng số, cũng được học bằng thuật toán.

Trong một LSTM chúng ta có ba cổng: đầu vào, cổng “quên” (forget) và cổng ra. Các cổng này xác định có hay không để đầu vào mới vào (cổng vào), xóa thông tin vì nó không quan trọng (cổng “quên”) hoặc để nó tác động đến đầu ra ở bước thời gian hiện tại (cổng đầu ra). Bạn có thể thấy một minh họa của một RNN với ba cổng bên dưới:



Hình 2.9. Một nơ-ron trong LSTM

Các cổng trong LSTM tương tự nhau, sử dụng hàm sigmod, nghĩa là nó có giá trị từ 0 đến 1. Sự tương tự này cho phép thực hiện lan truyền ngược.

Vấn đề toán học của việc đạo hàm tiêu biến được giải quyết thông qua LTSM bởi vì nó xử lý đủ tốt đạo hàm, do đó quá trình huấn luyện tương đối ngắn và chính xác cao.

## Ứng dụng học củng cố trong phân tích và phát hiện mã độc

Luận văn đề xuất một mô hình kết hợp Học củng cố nhằm sinh các PSI-walk từ đồ thị PSI-graph và sử dụng bộ dữ liệu PSI-walk thu được và Học sâu nhằm giải quyết bài toán phát hiện mã độc botnet trên các thiết bị IoT. Trong phần này, luận văn sẽ trình bày chi tiết kiến trúc mô hình cũng như luồng xử lý mà trong đó, PSI-walk là đặc trưng trọng tâm.

### Tổng quan

Trong phần này, luận văn trình bày tổng quan kiến trúc của mô hình bao gồm các giai đoạn xử lý chính cũng như các giai đoạn tiền xử lý dữ liệu. Đầu tiên,ta thu thập bộ dữ liệu PSI-graph đã được công bố bởi Huy Trung và các cộng sự [9]. Bộ dữ liệu PSI-graph được học thông qua một mô hình Học củng cố nhằm sinh ra bộ dữ liệu PSI-walk. Cuối cùng, các PSI-walk được coi như các đặc trưng đại diện cho tập tin ELF tương ứng và được huấn luyện trong một mô hình mạng LSTM đơn giản cho mục đích phát hiện mã độc. Luồng xử lý dữ liệu được minh họa thông qua hình 2.10 dưới đây.

Diagram

Description automatically generated

Hình 2.10. Tổng quan mô hình đề xuất

Chú ý, luận văn không sử dụng toàn bộ bộ dữ liệu PSI-graph trong luận văn này. Nhận xét rằng mô hình mạng LSTM dựa trên xử lý ngôn ngữ tự nhiên chỉ hoạt động tốt trên các dữ liệu tường minh, nên ta lược bỏ các đồ thị có toàn bộ các đỉnh bị mã hóa hoặc làm rối. Điều này là dễ hiểu bởi các đỉnh bị mã hóa hoặc làm rối thường chỉ xuất hiện trong đồ thị tương ứng đó, do đó, chúng ảnh hưởng đến phân phối của dữ liệu, dễ gây ra các vấn đề về data mismatch (dữ liệu giữa tập huấn luyện và tập kiểm thử có sự khác biệt lớn). Vì vậy, chỉ các đồ thị chứa ít nhất một đỉnh tường minh, hay không bị mã hóa thì mới được thực nghiệm với phương pháp đề xuất trong luận văn này.

### Định nghĩa PSI-walk

Như đã nói ở trên, luận văn thừa kế phương pháp đại diện tập tin ELF thông qua đồ thị PSI-graph [9]. Theo đó,

Định nghĩa 1. Đồ thị PSI-graph là đồ thị có hướng được định nghĩa là , trong đó:

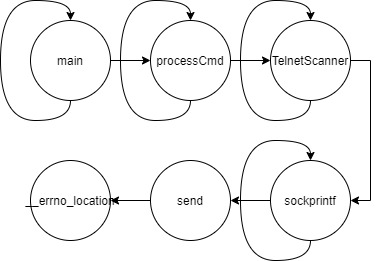
* V là tập các đỉnh PSI, thông thường là tên các hàm chức năng có trong mã nguồn của tập tin ELF.
* E là tập các cạnh có hướng, thể hiện mối liên kết giữa các đỉnh đồ thị.

Ngoài ra, đồ thị PSI là còn là đa đồ thị có khuyên, theo đó, giữa các đỉnh có thể có nhiều liên kết và một đỉnh cũng có thể có một hoặc nhiều khuyên.

Nghiên cứu bởi Huy Trung và các cộng sự [9] xem toàn bộ PSI-graph làm đặc trưng cho bài toán phân loại, và đạt được kết quả phân loại tốt. Tuy nhiên, nhận thấy rằng toàn bộ đồ thị có thể chứa nhiều thông tin không phải đặc trưng, hoặc một số thành phần trong đồ thị của mã độc là bình thường và tương đồng với đồ thị của tập tin lành tính; Huy Trung và các cộng sự đã phát triển PSI-rooted subgraph [10]. Trong đó, tác giả xem xét các đồ thị con trong toàn bộ đồ thị PSI-graph, nhấn mạnh những đồ thị con có cấu trúc đặc trưng cho hành vi mã độc, chẳng hạn như các chu trình. PSI-rooted subgraph cho kết quả phân loại tốt tương đương, nhưng có phương pháp xử lý tiết kiệm tài nguyên và thời gian hơn so với PSI-graph.

Trong luận văn này sẽ tập trung vào một cách xử lý khác đối với đồ thị PSI-graph. Phương pháp này chỉ trích xuất duy nhất một đường đi mang nhiều thông tin nhất trong đồ thị PSI-graph. Một đồ thị PSI-graph sẽ có nhiều đường đi khác nhau, mỗi đường đi sẽ thể hiện một cách thức mà ELF được vận hành. Đối với IoT botnet, một số hoặc nhiều các đường đi trong đó là hành vi bình thường của một chương trình phần mềm, do đó không thể làm đặc trưng cho hành vi mã độc của tập tin ELF đó. Chẳng hạn như quá trình cố gắng kết nối đến CnC server, v.v. Ví dụ minh họa dưới đây sẽ cho chúng ta cách nhìn khái quát về PSI-walk, đường đi tối ưu trong đồ thị PSI-graph.

Định nghĩa 2. Đường đi PSI-walk là một đường đi có hướng với các đỉnh và cạnh không nhất thiết phân biệt trong đồ thị PSI graph được định nghĩa là một tập hợp hữu hạn các cạnh liên kết một tập hợp hữu hạn các đỉnh . PSI-walk mà định nghĩa có thể chứa khuyên như là một phần phụ trợ bổ sung thêm thông tin cho đường đi chính. Một đường đi PSI-walk có hình ảnh trực quan được mô tả tại Hình 2.11 dưới đây.



Hình 2.11. Một ví dụ về PSI-walk

Hình 2.11 minh họa một PSI-walk của botnet BASHLITE. Nhìn vào PSI-walk, chúng ta có thể thấy đặc trưng hành vi rõ nét của IoT botnet: xử lý lệnh được truyền về từ CnC server thông qua hàm processCmd, rà quét lây nhiễm thông qua TelnetScanner và thực hiện tấn công DOS thông qua hàm send. Những hàm mang nhiều thông tin này sẽ là đặc trưng rất quan trọng giúp quá trình phân loại ở các giai đoạn sau diễn ra hiệu quả. PSI-walk cũng giữ lại các khuyên có sẵn của một đỉnh trong đồ thị PSI-graph nhằm tăng tính đa dạng cho bộ dữ liệu PSI-walk. Bên cạnh đó, chúng ta còn thấy tên các đỉnh còn mang nhiều thông tin về chức năng của hàm đó. Điều này được khẳng định một lần nữa thông qua việc nghiên cứu mã nguồn của một số botnet điển hình như BASHLITE và Mirai. Đó cũng chính là lý do các PSI-graph chỉ chứa các đỉnh bị mã hóa sẽ cho PSI-walk không có khả năng phân loại, bởi vì tên các hàm này không có mối tương quan với các PSI-walk khác trong bộ dữ liệu.

Hơn nữa, rõ ràng một đồ thị PSI-graph có thể có nhiều đường đi tối ưu, mang nhiều thông tin đặc trưng. Tuy nhiên, phương pháp sinh PSI-walk không nhất thiết phải chỉ ra đường đi tốt nhất. Một PSI-walk tối ưu hoặc gần tối ưu là đủ để các bộ phân loại hoạt động chính xác. Mặc dù mô hình sinh PSI-walk là mô hình Học củng cố đơn giản. Với tính ngẫu nhiên trong chiến lược chọn hành động hay là một quan hệ đánh đổi exploration-exploitation có thể khiến cho PSI-walk tìm được chưa phải là tối ưu nhất. Nhưng một PSI-walk gần tối ưu mang một số đặc trưng mã độc vẫn đóng góp tốt cho nhiệm vụ phân loại. Điều này đã được khẳng định thông qua thực nghiệm.

### Học củng cố

Luận văn sử dụng thuật toán học củng cố để sinh PSI-walk từ bộ dữ liệu PSI-graph. Bởi vì bản chất của bài toán sinh PSI-walk là tìm đường đi tối ưu trong đồ thị, nên việc áp dụng Học củng cố là hoàn toàn phù hợp. Với đặc tính thử-sai, Học củng cố dựa vào việc ngẫu nhiên di chuyển trong đồ thị PSI-graph và các chính sách tối ưu đã biết để tối ưu hóa đường đi tìm được. Bên cạnh đó, quá trình triển khai và điều chỉnh thuật toán Học củng cố cũng dễ dàng nhờ vào các tham số và cách định nghĩa phần thưởng.

Thực nghiệm cho thấy một thuật toán cơ bản như Q-learning cho kết quả tốt. Q-learning là một thuật toán Học củng cố off-policy, tìm hành động tốt nhất để thực hiện tại một trạng thái xác định nhờ vào giá trị Q-value. Trong quá trình học, Q-learning mục tiêu tối đa hóa phần thưởng nhận được trong khi thực hiện hành động tương tác với môi trường. Các thành phần của Q-learning nhằm sinh PSI-walk từ đồ thị PSI-graph sẽ được mô tả chi tiết sau đây.

Luận văn coi bài toán tìm đường đi tối ưu trong đồ thị PSI-graph như là vấn đề kinh điển trong Học củng cố: trò chơi Robot tìm đường đi tối ưu trong bản đồ/mê cung. Theo đó:

Agent (Tác tử) là robot đứng tại một đỉnh của đồ thị PSI-graph. Tác tử tương tác với môi trường bằng cách di chuyển trong đồ thị PSI-graph. Tác tử chỉ có thể di chuyển đến các đỉnh kề với đỉnh nó đang đứng. Quá trình di chuyển của tác tử sẽ được ghi nhận như là một PSI-walk. Sau khi quá trình học kết thúc, tác tử thực hiện di chuyển theo chính sách tối ưu đã được học để sinh ra PSI-walk cuối cùng.

Action (Hành động) là tập hợp các hành động mà tác tử có thể thực hiện. Do đó, không gian hành động sẽ bao gồm toàn bộ các đỉnh của PSI-graph. Như đã nói ở trên, tác tử chỉ có thể tới các đỉnh kề với đỉnh nó đang đứng. Hơn nữa, để tránh tác tử bị kẹt lại trong khuyên của một đỉnh, thuật toán cũng không cho phép tác tử đi vào khuyên của đỉnh nó đang đứng.

Environment (Môi trường) là không gian mà tác tử tác động, thay đổi trạng thái và nhận về phần thưởng. Tác tử học thông qua quá trình tương tác với môi trường. Ở đây, môi trường bao gồm đồ thị PSI-graph và cơ chế phản hồi với tác tử.

State (Trạng thái) là tình huống hiện tại của mô hình, bao gồm đỉnh hiện tại tác tử đang đứng, danh sách các cạnh tác tử đã đi qua. Khi môi trường trả về trạng thái tiếp theo của tác tử, biến trạng thái sẽ được cập nhật thông qua việc thay đổi đỉnh mà tác tử đang đứng và chèn thêm cạnh vừa đi vào danh sách cạnh nói trên.

Reward (Phần thưởng) là cách thức mà chúng ta đánh giá độ tốt của một hành động đối với một trạng thái cụ thể trên cơ sở mục tiêu bài toán. Độ tốt của một hành động sẽ được đánh giá thông qua lượng thông tin mà hành động tác tử thực hiện mang lại. Thuật toán định nghĩa một số quy định về phần thưởng dưới đây.

Ở một mức độ nào đó, các hàm mang nhiều thông tin là hàm có nhiều liên kết, đặc biệt là gọi nhiều các hàm khác. Chẳng hạn, lúc nghiên cứu mã nguồn của botnet họ BASHLITE, hàm processCmd chịu trách nhiệm nhận lệnh điều khiển từ CnC server thông qua tham số truyền vào từ trong hàm main, xử lý, dựa vào lệnh đó để kích hoạt cuộc tấn công tương ứng bằng cách gọi đến các hàm tấn công tương ứng, ví dụ như hàm sendTCP cho TCP flooding, hàm sendUDP cho UDP flooding, … Như vậy processCmd sẽ có nhiều cạnh ra nối với các hàm tấn công khác. PSI-walk cần bao gồm những đỉnh như thế này, từ đó mở rộng ra các đỉnh kề với nó. Do đó, độ quan trọng của một hàm sẽ phụ thuộc vào bậc ra (hay bậc ngoài) của đỉnh chứa hàm đó trong đồ thị PSI-graph.

Định nghĩa 3. Bậc ra của một đỉnh trong đồ thị được định nghĩa là số cạnh kề xuất phát từ đỉnh đó và được ký hiệu là

* Một số đỉnh trong đồ thị PSI-graph có thể có . Trường hợp này có thể kể đến một hàm trong thư viện chuẩn C như printf.
* Tổng các bậc ra trong một đồ thị bằng số cạnh trong đồ thị đó:

Bên cạnh đó, đồ thị PSI-graph là đa đồ thị có hướng có khuyên. Khuyên của một đỉnh sẽ không nhiều ý nghĩa bằng một liên kết gọi hàm từ đỉnh đó sang một đỉnh khác. Do đó, công thức phần thưởng phải thỏa mãn yêu cầu: đi tới một đỉnh có bậc ra càng lớn thì càng nhận được phần thưởng lớn, và đi tới một đỉnh có cùng bậc nhưng nhiều khuyên hơn thì sẽ nhận được ít phần thưởng hơn. Trong trường hợp, tác tử đi tới một đỉnh chỉ có khuyên thì sẽ nhận được một phần thưởng rất nhỏ.

Nói tóm lại, tác tử sau khi thực hiện hành động đi đến một đỉnh kề có bậc ra và số khuyên sẽ nhận được một phần thưởng được xác định bởi công thức sang đây:

Bên cạnh đó, dựa theo một số quy ước đặt tên hàm trong một số ngôn ngữ lập trình, đặc biệt là ngôn ngữ C thường được sử dụng để lập trình trên các thiết bị nhúng, các thiết bị IoT, một hàm bắt đầu bởi “\_” hoặc “\_\_” thường không có nhiều ý nghĩa, do đó đi đến các hàm này cũng chỉ nhận được phần thưởng bằng 1 như trường hợp đỉnh chỉ có khuyên ở trên. Lý do cho phần thưởng bằng 1 chính là giúp cho thuật toán vẫn đi vào những đỉnh này trong trường hợp không còn đỉnh tối ưu hơn nào khác để có thể đi tới.

Bên cạnh đó, thuật toán ưu tiên đi đến những đỉnh có khuyên hơn là những đỉnh không khuyên. Do đó, phần thưởng nhận được khi đi vào những đỉnh có khuyên sẽ được nhân đôi, cụ thể .

Hơn nữa, đồ thị PSI-graph có thể chứa chu trình. Do đó, để tác tử không bị mắc kẹt trong một chu trình, thuật toán định nghĩa một cơ chế giảm phần thưởng nhận được tại đỉnh đó nếu tác tử đã đi qua nó. Cơ chế này khiến cho việc ở trong chu trình càng lâu thì phần thưởng nhận được càng ít đi và giúp cho tác tử sau một hoặc một số lần lặp có thể phá vỡ chu trình và tìm một con đường tối ưu khác kế tiếp. Theo đó, mỗi lần sử dụng một cạnh để thăm một đỉnh, phần thưởng của việc đi đến đỉnh đó qua cạnh đó trong các lần sau sẽ chỉ còn một nửa. Cụ thể, di chuyển đến một đỉnh đã được thăm từ đỉnh hiện tại sẽ nhận được một phần thưởng bằng:

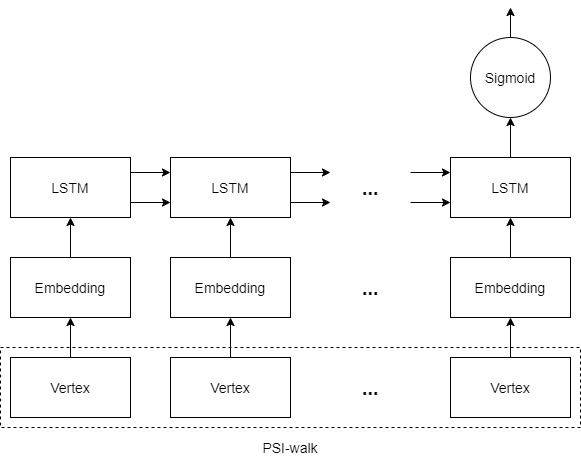
Trong đó T là số lần mà tác tử đã thăm đỉnh cần đến từ đỉnh tác tử đứng hiện tại, hay là số lần tác tử đi qua cạnh nối chúng.

Thuật toán Q-learning sử dụng chiến lược tham lam epsilon. Quá trình học với mỗi PSI-graph được thực hiện thông qua 1000 lần học (episode). Khi thuật toán dừng học, bằng Q-table đã được xây dựng xong, dựa vào các giá trị Q-value, thuật toán thực hiện việc di chuyển tác tử một lần nữa theo chính sách tối ưu. Các đỉnh mà tác tử đi qua được đưa vào PSI-walk cuối cùng. Như đã đề cập ở trên, các khuyên không được thăm trong quá trình duyệt qua các đỉnh của đồ thị PSI-graph, do đó, để thể hiện những khuyên này, thuật toán sẽ lặp lại những đỉnh có khuyên trong kết quả cuối cùng. Một PSI-walk như Hình 2.11 sẽ được lưu lại dưới dạng chuỗi như sau: “*main main processCmd processCmd TelnetScanner TelnetScanner sockprintf sockprintf send \_\_errno\_location”.*

Như vậy mô hình Học củng cố lần lượt học qua từng đồ thị PSI-graph và sinh đồ thị PSI-walk tương ứng. Kết quả thu được chính là tập dữ liệu PSI-walk.

### Phân loại

Như đã đề cập ở trước, PSI-walk là một chuỗi các đỉnh PSI. Do đó luận văn tiếp cận cách sử dụng các mô hình học sâu liên quan đến xử lý chuỗi. Rõ ràng có thể coi mỗi PSI-walk là một câu với mỗi đỉnh có trong PSI-walk là các từ. Bài toán phân loại PSI-walk sẽ được đưa về bài toán phân tích tình cảm (sentiment analysis) trong xử lý ngôn ngữ tự nhiên. Luận văn sử dụng một mạng LSTM nông, một chiều để phân tích PSI-walk dựa vào đặc tính chuỗi và tính thứ tự của các đỉnh trong PSI-walk. Kiến trúc mạng được mô tả ở hình dưới.



Hình 2.12. Kiến trúc mạng phân loại LSTM

Như có thể thấy ở trên, đây là một mạng nơ-ron hồi quy nông dạng many-to-one. Đầu tiên, các đỉnh trong PSI-walk được xử lý để đưa về dạng vector thông qua lớp Embedding. Lớp này là lớp có thể huấn luyện được. Tiếp đến mỗi một embedding được truyền vào một khối LSTM và huấn luyện như một bài toán phân loại nhị phân. Nơ-ron ở lớp cuối cùng sử dụng hàm kích hoạt sigmoid tính xác suất để PSI-walk đầu vào thuộc về lớp mã độc. Bên cạnh đó, luận văn áp dụng kỹ thuật dropout để phòng chống hiện tượng overfitting.

Các thông số cụ thể của mạng LSTM bao gồm:

Bảng 2.1. Tham số mạng LSTM

|  |  |
| --- | --- |
| Mô tả tham số | Giá trị |
| Số chiều embedding từ | 16 |
| Spatial dropout sau lớp embedding | p=0.4 |
| Số chiều của nơ-ron trong LSTM unit | 16 |
| Dropout, recurrent dropout trong LSTM | p=0.2 |
| Thuật toán tối ưu | ADAM |
| Hàm mất mát | Cross-entropy |
| Số epoch | 100 |
| Batch size | 512 |

## Tổng kết chương

Chương này đã trình bày về đồ thị đặc trưng có cấu trúc PSI , và đề xuất kết hợp giữa đồ thị PSI và phương pháp học củng cố để trích xuất đặc trưng và mô hình phân loại mã độc.Việc kết hợp này đã thu lại được những kết quả tích cực.Chương 3 sẽ trình bày về bộ dữ liệu được sử dụng và các kết quả thu được.

# KẾT QUẢ THỰC NGHIỆM

*Trong chương này, luận văn đánh giá độ hiệu quả-chính xác của mô hình thông qua thực nghiệm. Bộ dữ liệu đa kiến trúc của PSI-graph được sử dụng lại từ nghiên cứu [10]. Bộ dữ liệu này chứa 10010 mẫu PSI-graph bao gồm 6165 mẫu IoT botnet và 3845 mẫu lành tính. Thực nghiệm được thực hiện trên PC với cấu hình như sau: Windows 10 Pro 64-bit, Intel Core i5, 8Gb RAM. Có thể thấy việc huấn luyện các mô hình trên không yêu cầu một máy tính mạnh có nguồn tài nguyên lớn. Mã nguồn thực nghiệm được viết bằng ngôn ngữ Python.*

## Bộ dữ liệu PSI-graph

Tác giả bài báo PSI-graph [9] thu thập bộ dữ liệu ELF thông qua 3 nguồn chính:

* Bộ dữ liệu IoTPOT [24] chứa 4000 mẫu IoT botnet.
* Bộ dữ liệu VirusShare [22] chứa 3779 mẫu IoT botnet.
* 4000 mẫu lành tính được thu thập thông qua các kho dữ liệu trực tuyến và IoT SOHO và các tập tin này được kiểm tra lại bằng các engine của hệ thống VirusTotal.

Sau đó, các tập tin ELF được dịch ngược bằng phần mềm IDA Pro [5] để sinh đồ thị luồng điều khiển (CFG). Các đồ thị này được xử lý, chọn lọc để sinh ra đồ thị PSI cuối cùng. Các tập tin ELF bị hỏng, gây rối và không thể được dịch ngược sẽ bị loại bỏ khỏi tập dữ liệu. Tập dữ liệu PSI-graph cuối cùng bao gồm 6165 IoT botnet. Các botnet này thuộc nhiều loại mã độc botnet khác nhau, nhưng chủ yếu thuộc 2 họ điển hình: BASHLITE và Mirai. Chúng cũng được thu thập từ nhiều thiết bị có kiến trúc tập lệnh khác nhau bao gồm: ARM, MIPS, Intel 80386, x86-64, PowerPC, Motorola, Spark và SuperH. Các bảng dưới đây mô tả số lượng IoT botnet có trong bộ dữ liệu PSI-graph tương ứng với các lớp và 2 kiến trúc CPU phổ biến.

Bảng 3.1. Mô tả bộ dữ liệu PSI-graph

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lớp/ | BASHLITE | Mirai | Botnet khác | Lành tính |
| ARM | 762 | 331 | 152 | 561 |
| MIPS | 646 | 301 | 103 | 533 |
| Các kiến trúc khác | 2312 | 1133 | 425 | 2751 |
| Tổng | 3720 | 1765 | 680 | 3845 |

Bảng 3.2 thể hiện các thống kê về số lượng đỉnh và cạnh trong bộ dữ liệu PSI-graph. Các thống kê này chính là cơ sở cho việc chọn số lần học trong mô hình Học củng cố đã trình bày ở trên.

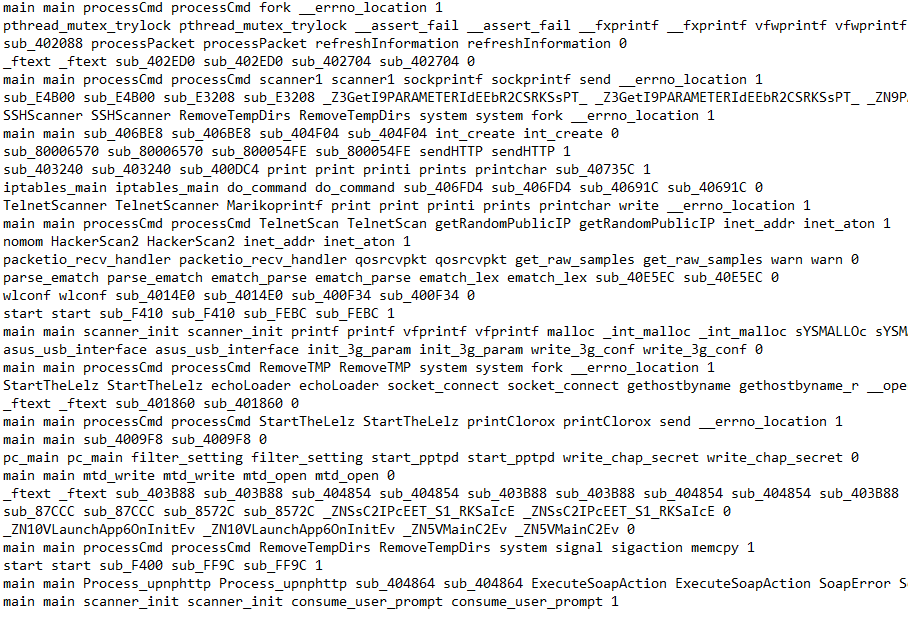
Bảng 3.2. Thống kê về số lượng cạnh và đỉnh trong PSI-graph

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Số lượng đỉnh | | Số lượng cạnh | |
| Tối đa | Trung bình | Tối đa | Trung bình |
| BASHLITE | 571 | 168 | 2816 | 624 |
| Mirai | 1813 | 38 | 18093 | 233 |
| Botnet khác | 2097 | 280 | 9882 | 1111 |
| Lành tính | 11874 | 167 | 65817 | 1694 |

## Bộ dữ liệu PSI-walk

Bộ dữ liệu PSI-graph được mô tả ở trên sẽ được đưa vào học thông qua mô hình Học củng cố và sinh được bộ dữ liệu PSI-walk. Rõ ràng là các botnet thuộc cùng một họ botnet có PSI-walk gần giống nhau hoặc giống nhau toàn bộ. Các tập tin ELF là phiên bản cross-compiled của cùng một mã nguồn botnet trên các kiến trúc tập lệnh khác nhau cũng sinh cùng một PSI-walk. Bên cạnh đó, các PSI-walk chỉ chứa các đỉnh đã bị mã hóa cũng bị loại bỏ khỏi bộ dữ liệu. Do đó bộ dữ liệu PSI-walk sẽ có ít mẫu hơn PSI-graph. Cụ thể, bộ dữ liệu PSI-walk bao gồm 499 mẫu IoT botnet và 1163 mẫu mã độc. Do bài toán thực hiện là bài toán phát hiện mã độc nên bộ dữ liệu PSI-walk chỉ được đánh nhãn mã độc – lành tính chứ không cụ thể các lớp BASHLITE và mirai, và mô hình dự đoán cũng trở thành mô hình phân loại nhị phân.

Kết quả phân tích cho thấy bộ dữ liệu PSI-walk có tập từ điển bao gồm 2665 tên đỉnh khác nhau.



Hình 3.1. Hình ảnh minh họa các PSI-walk và nhãn của chúng

## Tiêu chí đánh giá

Các tiêu chí sau được sử dụng cho việc đánh giá độ hiệu quả-chính xác của phương pháp đề xuất:

* Condition positive (P): số mẫu mã độc trong bộ dữ liệu.
* Condition negative (N): số mẫu lành tính trong bộ dữ liệu.
* True positive (TP): số mẫu mã độc được phân loại đúng là mã độc.
* True negative (TN): số mẫu lành tính được phân loại đúng là lành tính.
* False positive (FP): số mẫu lành tính bị gắn nhầm nhãn thành mã độc.
* False negative (FN): số mẫu mã độc bị gắn nhầm nhãn thành lành tính.

Các tiêu chí được sử dụng để đánh giá độ chính xác-hiệu quả của mô hình được xây dựng như sau:

* True positive rate (TPR) hay Sensitivity, Recall, Hit rate: Tỷ lệ số mẫu mã độc được dự đoán đúng trên tổng số các mẫu thực sự là mã độc. Tiêu chí cho thấy xác suất phát hiện mã độc của mô hình. Một mô hình có TPR cao đồng nghĩa với việc mô hình bỏ sót ít các mẫu thực sự là mã độc.
* False positive rate (FPR) hay Fall-out: Tỷ lệ số mẫu lành tính được dự đoán nhầm thành mã độc trên tổng số các mẫu lành tính. Tiêu chí cho thấy xác suất báo động sai của mô hình. Một mô hình có FPR thấp đồng nghĩa với việc mô hình ít khi báo động nhầm mã độc.
* Accuracy (ACC): Tỷ lệ các mẫu dự đoán đúng trên tổng các mẫu được dự đoán. ACC thể hiện độ hiệu quả của mô hình nói chung, tuy nhiên không đáng tin cậy trong các bộ dữ liệu không cân bằng.
* Precision hay positive predictive rate (PPV): Tỷ lệ số mẫu mã độc được dự đoán đúng trên tổng số các điểm được dự đoán là mã độc. PPV thể hiện độ chính xác của mô hình.
* F1-score: là trung bình cộng điều hòa (harmonic mean) của Precision và Recall. F1-score càng cao thể hiện bộ phân lớp càng tốt.
* Area Under the Curve (AUC): Tiêu chí dựa trên đường ROC để đánh giá độ hiệu quả của mô hình. Đặc biệt AUC thường được dùng trong các bài toán phân lớp nhị phân với bộ dữ liệu không cân bằng.

## Kết quả và đánh giá

Quá trình thực nghiệm với mạng LSTM được chia thành 2 pha: Pha huấn luyện và pha kiểm thử. Do bộ dữ liệu PSI-walk khá nhỏ nên tỷ lệ train:test được lấy là tỷ lệ 7:3. Trong quá trình huấn luyện, một phần nhỏ của tập huấn luyện được sử dụng như là tập xác thực (validation set) với tỷ lệ 1/5. Tập xác thực giúp đánh giá quá trình huấn luyện qua các epoch, lưu lại model có độ chính xác tốt nhất và phòng tránh overfitting trên tập kiểm thử. Sau pha huấn luyện, mô hình sẽ được kiểm thử thông qua tập dữ liệu kiểm thử. Kết quả phân loại chi tiết được trình bày dưới đây.

Bảng 3.3. Kết quả thực nghiệm

|  |  |
| --- | --- |
| Tiêu chí (%) | LSTM |
| Accuracy | 95.39 |
| Precision | 97.71 |
| Recall | 90.00 |
| F1-score | 93.70 |
| ROC AUC | 94.35 |

Bộ phân loại bằng LSTM cho kết quả tốt với 95.39% accuracy, 93.70% F1-score. Như vậy, LSTM có khả năng phát hiện hiệu quả và chính xác các PSI-walk của mã độc IoT botnet.

Chart, line chart

Description automatically generated

Hình 3.2. Đường ROC của model

Hình 3.2 mô tả đường ROC thu được. Chúng ta có thể thấy đường ROC cho phần diện tích AUC khá tốt. Điều đó có nghĩa rằng mô hình hoạt động tốt trên cả 2 lớp. Vấn đề bộ dữ liệu mất cân bằng không ảnh hưởng lớn đến khả năng dự đoán của mô hình hay mô hình không bị quá thiên lệch về lớp đa số. Hơn nữa, đường ROC còn cho thấy mô hình có tỷ lệ báo động giả thấp (FPR=1.29%), do đó tăng độ tin cậy khi có mẫu ELF được mô hình phát hiện là mã độc.

Chart, treemap chart

Description automatically generated

Hình 3.3. Confusion matrix

Hơn nữa, mặc dù LSTM là một mô hình Học sâu thường yêu cầu nhiều tài nguyên tính toán và thời gian huấn luyện lâu, mô hình nông được sử dụng lại hội tụ khá nhanh. Mô hình đạt độ chính xác trên tập xác thực lên đến 97% chỉ sau 35 epoch, mỗi epoch train chưa đến 1 giây. Quá trình kiểm thử hoàn tất chỉ sau 0.991 giây và tất cả đều thực hiện trên CPU. Như vậy, mô hình LSTM đề xuất không yêu cầu tài nguyên phần cứng mạnh và không tiêu tốn thời gian.

Chart, line chart, histogram

Description automatically generated

Hình 3.4. Lịch sử huấn luyện

So sánh với các nghiên cứu gần đây trong vấn đề phát hiện mã độc botnet trên các thiết bị IoT, rõ ràng phương pháp trên thể hiện độ hiệu quả tốt hơn. Nghiên cứu của HaddadPajouh và các cộng sự [8] sử dụng đặc trưng Opcode và mạng nơ-ron hồi quy nhằm phát hiện mã độc IoT. Tuy nhiên phương pháp này chỉ được áp dụng trên các tập tin thuộc kiến trúc ARM. Su và các cộng sự [14] sử dụng đặc trưng ảnh xám và mạng nơ-ron tích chập nhằm phát hiện mã độc IoT, tuy nhiên chỉ đạt được 94% accuracy trên bộ dữ liệu không lớn của tác giả.

So sánh với các nghiên cứu của Huy Trung và các cộng sự [9], [10], mặc dù mô hình không cho tỷ lệ phát hiện cao hơn nhưng mô hình trên vẫn có độ chính xác cao, với tỷ lệ báo động giả thấp hơn. Hơn nữa, phương pháp PSI-graph cần có phần cứng mạnh để chạy 2 mô hình graph2vec và mạng nơ-ron tích chập, phương pháp PSI-rooted subgraph cần không gian bộ nhớ lớn nhằm lưu trữ các subgraph thu được. Trong khi đó, phương pháp PSI-walk được đề xuất thì không cần yêu cầu nguồn tài nguyên mạnh, cũng như khả năng huấn luyện trong thời gian ngắn. Bên cạnh đó, phương pháp PSI-walk còn cho người nghiên cứu cách nhìn trực quan về cách botnet hoạt động, là sự tóm tắt của đồ thị PSI-graph.

Như vậy, phương pháp được đề xuất dựa trên đặc trưng PSI-walk là một cách tiếp cận đáng hứa hẹn, có thể tiếp tục được mở rộng, phát triển trên các bài toán phân loại, các loại mã độc khác trên IoT.

KẾT LUẬN

Luận văn tập trung nghiên cứu phương pháp phân tích luồng hoạt động có thể là của mã độc trên đồ thị PSI từ đó phát hiện mã độc botnet trên các thiết bị IoT thông dụng. Luận văn đã đạt được một số kết quả sau:

* Xây dựng bộ dữ liệu là tập các đồ thị PSI của các mã độc nổi tiếng.
* Đề xuất phương pháp sử dụng học củng cố trên đồ thị PSI để trích xuất các đặc trưng đại diện cho tập tin thực thi.
* Sử dụng phương pháp LSTM để phát hiện mã độc dựa trên đặc trưng đã được trích xuất.

Trong tương lai, luận văn có thể tiếp tục được nghiên cứu để tăng cường độ chính xác và hiệu quả cao hơn nữa.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti(2014), “A large-scale analysis of the security of embedded firmwares,” *Proc. 23rd USENIX Secur. Symp.*, pp. 95–110.

[2] A. Darki, C.-Y. Chuang, M. Faloutsos, Z. Qian, and H. Yin(2018), “RARE: A Systematic Augmented Router Emulation for Malware Analysis,” Pam, vol. 2 , no. March, pp. 100–113, doi: 10.1007/978-3-319-76481-8.

[3] Detect-It-Easy,https://github.com/horsicq/Detect-It-Easy , truy cập ngày 12/02/2019

[4] D. F. Rajesh Kandaswamy, “ Blockchain-based transformation,”https://www.gartner.com/en/doc/3869696-blockchain-based-transformation-a-gartner-trend-insight-report/, truy cập ngày 05/06/2018.

[5] Eagle, C(2011),”The IDA Pro Book”,The Unofficial Guide to the World’s Most Popular Disassembler. William Pollock, Clifton

[6] Flashpoint, “Mirai Botnet Linked to Dyn DNS DDoS Attacks,”https://www.ﬂashpoint-intel.com/blog/cybercrime/mirai-botnet-linked-dyn-dns-ddos-attacks/, truy cập ngày 18/12/2018

[7] Gsma, “Safety, privacy and security”, <https://www.gsma.com/publicpolicy/resources/safetyprivacy-security-across-mobile-ecosystem/> , truy cập ngày 29/02/2019

[8] H. HaddadPajouh, A. Dehghantanha, R. Khayami, và K.-K. R. Choo(2018), “A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting”, *Future Generation Computer Systems*, vol 85, tr 88–96.

[9] Huy-Trung Nguyen, Quoc-Dung Ngo,Van-Hoang Le(2019),” A novel graph-based approach for IoT botnet detection”, Springer-Verlag GmbH Germany, part of Springer Nature

[10] H.-T. Nguyen, Q.-D. Ngo, D.-H. Nguyen, và V.-H. Le(2020, January), “PSI-rooted subgraph: A novel feature for IoT botnet detection using classifier algorithms”, *ICT Express*, doi: 10.1016/j.icte.2019.12.001.

[11] H. T. Nguyen, Q. D. Ngo, and V. H. Le(*2018*), “IoT Botnet Detection Approach Based on PSI graph and DGCNN classifier,” *2018 IEEE Int. Conf. Inf. Commun. Signal Process. ICICSP*, no. Icsp, pp. 118–122, 2018, doi: 10.1109/ICICSP.2018.8549713.

[12] Ivan Valeryevich Evdokimov, Ameen Raad Jihad Alalwan, Roman Yurievich Tsarev\*, Tatiana Nikolaevna Yamskikh, Oksana Alekseevna Tsareva and Aleksander Nikolaevich Pupkov(2019, March),” A cost estimation approach for IoT projects”

[13] Jaewoo Jeon ,Youngho Cho(2019, August), “Construction and Performance Analysis of Image Steganography-based Botnet in KakaoTalk Openchat”.

[14] J. Su, D. V. Vargas, S. Prasad, D. Sgandurra, Y. Feng, và K. Sakurai, “Lightweight Classification of IoT Malware based on Image Recognition”, *arXiv:1802.03714 [cs]*,Available at: <http://arxiv.org/abs/1802.03714>. Truy cập: ngày 27/04/2020.

[15] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti(2014, February), “Avatar: A Framework to Support Dynamic Security Analysis of Embedded Systems’ Firmwares,” , pp. 23–26, doi: 10.14722/ndss.2014.23229.

[16] Manos Antonakakis, Tim April‡ Michael Bailey, Matthew Bernhard,” Understanding the Mirai Botnet” , the 26th USENIX Security Symposium.

[17] N. Q. Dũng, L. V. Hoàng, and N. H. Trung(2018), “Phát hiện mã độc IoT botnet dựa trên đồ thị PSI với mô hình Skip -gram,” no. 07, pp. 29–36.

[18] P. Čeleda, R. Krejčí, J. Vykopal, and M. Drašar(2010), “Embedded malware - An analysis of the Chuck Norris botnet,” Proc. - Eur. Conf. Comput. Netw. Defense, EC2ND, pp. 3–10, doi: 10.1109/EC2ND.2010.15.

[19] R. A. Rodr´ıguez-G´omez, G. Maci´a-Fern´andez and P. Garc´ıa-Teodoro , “Analysis of botnets through life-cycle “ , CITIC-Department of Signal Theory, Telematics and Communications, University of Granada C/Periodista Daniel Saucedo Aranda s/n E-18071, Granada, Spain

[20] Shang, S., Zheng, N., Xu, J., Xu, M., Zhang, H(2010),”Detecting malware variants via function-call graph similarity”,International Conference on Malicious and Unwanted Software. IEEE, pp. 113–120

[21] The Ultimate Packer for eXecutables,[truy cập ngày 12/02/2019](https://github.com/upx.%20Accessed%2012%20Feb%202019)

[22] VirusShare.com, https://virusshare.com/ ,truy cập ngày 06/08/2020.

[23] Xu, M., et al.:(2013) “A similarity metric method of obfuscated malware using function-call graph. J. Comput. Virol. Hacking Tech”. 9(1), 35–47.

[24] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow(2016), “IoTPOT: A novel honeypot for revealing current IoT threats,” J. Inf. Process., vol. 24, no. 3, pp. 522–533, doi: 10.2197/ipsjjip.24.522.