|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| LUẬN VĂN THẠC SĨ  **NGHIÊN CỨU THUẬT TOÁN DGCNN TRONG PHÂN TÍCH MÃ ĐỘC IOT BOTNET DỰA TRÊN ĐỒ THỊ CẤU TRÚC PSI-GRAPH** |
| **Chuyên ngành: An toàn thông tin**  **Mã số: 8480202** |
| Hà Nội, 2020 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| LUẬN VĂN THẠC SĨ  **NGHIÊN CỨU THUẬT TOÁN DGCNN TRONG PHÂN TÍCH MÃ ĐỘC IOT BOTNET DỰA TRÊN ĐỒ THỊ CẤU TRÚC PSI-GRAPH** |
| **Chuyên ngành: An toàn thông tin**  **Mã số: 8480202**  *Họ và tên học viên*:  **Ngô Hoàng Hạnh Nhân**  Khoá: CAO HỌC AN TOÀN 5  *Người hướng dẫn khoa học:*  **TS. Ngô Quốc Dũng**  Học viện Công nghệ Bưu chính Viễn thông |
| Hà Nội, 2020 |

**MỤC LỤC**

[DANH MỤC CÁC HÌNH VẼ i](#_Toc38714913)

[Chương 1. Tổng quan về IoT botnet và các phương pháp phát hiện IoT botnet 1](#_Toc38714914)

[1.1. Tổng quan về thiết bị IoT 1](#_Toc38714915)

[1.2. Kiến trúc cơ bản của thiết bị IoT 3](#_Toc38714916)

[1.2.2. Sensor 4](#_Toc38714917)

[1.2.3. Actuator 5](#_Toc38714918)

[1.2.4. Device 5](#_Toc38714919)

[1.2.5. Gateway 6](#_Toc38714920)

[1.2.6. IoT Integration Middleware 6](#_Toc38714921)

[1.2.7. Application 7](#_Toc38714922)

[1.3. Vấn đề an ninh với IoT 8](#_Toc38714923)

[1.4. Mã độc IoT Botnet 10](#_Toc38714924)

[1.4.1. Các thành phần cơ bản của mạng IoT botnet 11](#_Toc38714925)

[1.4.2. Mô hình hoạt động của một mạng IoT botnet cơ bản 12](#_Toc38714926)

[1.4.3. Ví dụ về mạng IoT botnet: Mirai botnet [9] 14](#_Toc38714927)

[1.5. Tổng quan về các phương pháp phát hiện IoT Botnet 15](#_Toc38714928)

[1.5.1. Phương pháp phân tích động 15](#_Toc38714929)

[1.5.2. Phương pháp phân tích tĩnh 16](#_Toc38714930)

[1.5.3. Phát hiện sử dụng phương pháp học sâu (Deep Learning) 17](#_Toc38714931)

[1.6. Tổng kết chương 18](#_Toc38714932)

[Chương 2. Ứng dụng thuật toán DGCNN và đồ thị cấu trúc PSI-graph trong phát hiện IoT botnet 19](#_Toc38714933)

[2.1. Tổng quan 19](#_Toc38714934)

[2.1.1. Mạng nơ ron tích chập - Convolutional Neural Network (CNN) [17] 19](#_Toc38714935)

[2.1.2. Đồ thị cấu trúc PSI-graph 26](#_Toc38714936)

[2.2. Phương pháp đề xuất 27](#_Toc38714937)

[2.2.2. Trích xuất đồ thị có cấu trúc PSI-graph từ các tập tin ELF đầu vào 28](#_Toc38714938)

[2.2.3. Trích xuất các đồ thị con PSI-rooted 29](#_Toc38714939)

[2.2.4. Tiền xử lý dữ liệu - Data preprocessing phase 31](#_Toc38714940)

[2.2.5. Tác vụ phân lớp 32](#_Toc38714941)

[2.3. Tổng kết chương 33](#_Toc38714942)

[Chương 3. Thực nghiệm kết quả 34](#_Toc38714943)

[3.1. Giới thiệu bộ dữ liệu IoT botnet 34](#_Toc38714944)

[3.2. Kết quả thực nghiệm, đánh giá độ chính xác và hiệu quả 34](#_Toc38714945)

[3.3. Tổng kết chương 34](#_Toc38714946)

[KẾT LUẬN 35](#_Toc38714947)

[TÀI LIỆU THAM KHẢO 36](#_Toc38714948)

# DANH MỤC CÁC HÌNH VẼ

[Hình 1.1. Ngôi nhà thông minh 2](#_Toc38655852)

[Hình 1.2. Ước tính đến năm 2023 thế giới đạt 25 tỷ thiết bị IoT 3](#_Toc38655853)

[Hình 1.3. Kiến trúc IoT tham khảo [2] 4](#_Toc38655854)

[Hình 1.4. RaspberryPis 4](#_Toc38655855)

[Hình 1.5. Luồng tín hiệu từ Sensor đến Actuator 5](#_Toc38655856)

[Hình 1.6. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [3] 7](#_Toc38655857)

[Hình 1.7. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [4] 7](#_Toc38655858)

[Hình 1.8. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [5] 8](#_Toc38655859)

[Hình 1.9. Các thành phần cơ bản của mạng IoT botnet 11](#_Toc38655860)

[Hình 1.10. Vòng đời của một IoT botnet 12](#_Toc38655861)

[Hình 1.11. Các sự kiện chính của Mirai botnet 14](#_Toc38655862)

[Hình 1.12. Mô hình hoạt động của Mirai botnet 14](#_Toc38655863)

[Hình 2.1. Ví dụ phép tích chập 21](#_Toc38655864)

[Hình 2.2. Hàm Sigmoid Activation Function 22](#_Toc38655865)

[Hình 2.3. Hàm Tanh Activation Function 23](#_Toc38655866)

[Hình 2.4. Hàm Softmax Activation Function 23](#_Toc38655867)

[Hình 2.5. Hàm ReLU Activation Function 24](#_Toc38655868)

[Hình 2.6. Max-pooling 25](#_Toc38655869)

[Hình 2.7. Mạng nơ-ron đơn giản 27](#_Toc38655870)

[Hình 2.8. Mạng nơ-ron sau khi dropout 27](#_Toc38655871)

[Hình 2.9. Mô hình đề xuất 29](#_Toc38655872)

[Hình 2.10. Duyệt cây đồ thị BFS 31](#_Toc38655873)

[Hình 2.11. Duyệt cây đồ thị DFS 31](#_Toc38655874)

# Tổng quan về IoT botnet và các phương pháp phát hiện IoT botnet

Chương này gồm 2 phần chính gồm: Phần 1 tập trung giới thiệu tổng quan các kiến thức về các thiết bị IoT phổ thông và IoT Botnet, Phần 2 giới thiệu các nghiên cứu liên quan gồm phân tích động và phân tích tĩnh, đưa ra ưu và hạn chế của mỗi loại phương pháp. Bên cạnh đó đi vào phân tích một số phương pháp tiếp cận phân tích IoT botnet đã biết, từ đó phân tích ưu điểm và hạn chế của mỗi phương pháp tiếp cận.

## Tổng quan về thiết bị IoT

IoT là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết (identifiable) cũng như chỉ sự tồn tại của chúng trong một kiến trúc mang tính kết nối. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT, nơi thiết lập các quy chuẩn toàn cầu cho RFID (một phương thức giao tiếp không dây dùng sóng radio) cũng như một số loại cảm biến khác. IoT sau đó cũng được dùng nhiều trong các ấn phẩm đến từ các hãng công nghệ lớn và các nhà nghiên cứu.

Vào tháng 6 năm 2009, Ashton từng cho rằng hiện nay máy tính gần như phụ thuộc hoàn toàn vào con người để chuyển tải dữ liệu. Gần như tất cả trong số 50 petabyte dữ liệu đang có trên Internet đều được ghi lại hoặc tạo ra bởi con người chúng ta thông qua các các thức như gõ chữ, nhấn nút, chụp ảnh, quét mã vách.... Con người chính là nhân tố quyết định trong thế giới Internet hiện nay. Thế nhưng con người lại có nhiều nhược điểm: chúng ta chỉ có thời gian hạn chế, khả năng tập trung và độ chính xác cũng ở mức thấp so với máy móc. Điều đó có nghĩa là chúng ta không giỏi trong việc thu thập thông tin về thế giới xung quanh và đây là một vấn đề lớn.

Nếu như máy tính có khả năng giúp con người thu thập tất cả những dữ liệu về mọi thứ xung quanh thì chúng ta có thể theo dõi mọi thứ. Chúng ta sẽ biết chính xác khi nào các vật dụng cần phải sửa chữa, thay thế khi nào chúng còn mới và khi nào thì chúng hết hạn sử dụng. Chưa kể đến việc chúng ta có thể kiểm soát chúng mọi lúc mọi nơi. IoT có tiềm năng thay đổi thế giới giống như cách mà Internet đã thay đổi cuộc sống của chúng ta. Ngôi nhà thông minh với các bóng đèn thông minh, máy giặt thông minh, tủ lạnh thông minh,.. có thể xem là bước đầu của IoT bởi chúng đều được liên kết với nhau hoặc liên kết với Internet.

A close up of a sign

Description automatically generated

* + - * 1. Ngôi nhà thông minh

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 một dự báo của Gartner, đến năm 2023 số lượng thiết bị IoT được kết nối dự kiến sẽ đạt 25 tỷ thiết bị. Điều này đem lại cơ hội kinh doanh lớn, nhưng song song với đó, nó cũng tiềm tàng nhiều mối đe doạ tấn công bảo mật tinh vi hơn [1].

A screenshot of a cell phone

Description automatically generated

* + - * 1. Ước tính đến năm 2023 thế giới đạt 25 tỷ thiết bị IoT

## Kiến trúc cơ bản của thiết bị 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, nhưg 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.

[2] đã đề xuất một kiến trúc IoT dựa 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

* + - * 1. Kiến trúc IoT tham khảo [2]

### Sensor

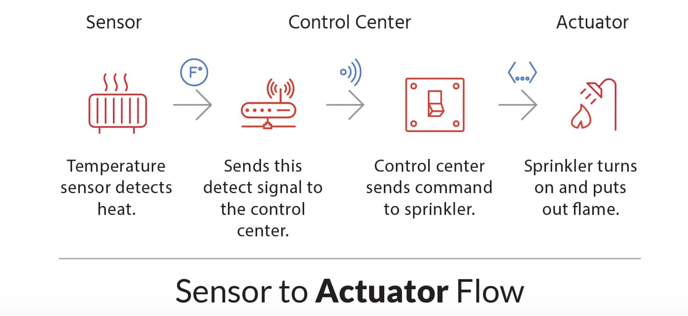
Sensor là một thành phần phần cứng, được sử dụng để đo các tham số của môi trường vật lý của nó và để chuyển chúng thành tín hiệu điện, ví dụ, bằng cách đo nhiệt độ hoặc độ ẩm của phòng. Nếu được yêu cầu, Sensor có thể được cấu hình bằng phần mềm, nhưng không thể tự chạy phần mềm. Thông thường, Sensor 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.



* + - * 1. RaspberryPis

### 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).



* + - * 1. Luồng tín hiệu từ Sensor đến Actuator

### 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 [2] 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 [2] đưa ra cũng có thể là một IoT Integration Middleware khác, tuỳ vào mục đích sử dụng, ví dụ, để tích hợp nhiều hệ thống.

Bên cạnh kiến trúc do [2] đề xuất thì có một số đề xuất khác về kiến trúc của IoT sytem nhưng đều có thể tham chiếu đến kiến trúc do [2] đề xuất như:

A screenshot of a cell phone

Description automatically generated

* + - * 1. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [3]

A screenshot of a cell phone

Description automatically generated

* + - * 1. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [4]

A screenshot of a cell phone

Description automatically generated

* + - * 1. Tương quan kiến trúc IoT do [2] đề xuất và kiến trúc của [5]

Các nền tảng IoT đang thu hút sự chú ý ngày càng tăng. Tuy nhiên, do thiếu một định nghĩa rõ ràng về các thành phần trong kiến trúc IoT, [2] đã giới thiệu một kiến trúc tham chiếu IoT trừu tượng hơn để cho phép một thuật ngữ thống nhất và để dễ dàng so sánh các nền tảng.

Tuy nhiên các thiết bị IoT đều gặp vấn đề về an ninh.

## Vấn đề an ninh với IoT

Với tính phổ biến, tính hữu dụng và tầm quan trọng của IoT đã đem lại cơ hội kinh doanh lớn, tuy nhiên cũng chính vì thế đã khiến IoT trở thành mục tiêu hấp dẫn của tội phạm mạng mạng. Trong kỷ nguyên Internet kết nối vạn vật (IoT), nhiều doanh nghiệp, tổ chức mới chỉ nghĩ nhiều đến mặt tích cực, các cơ hội và doanh thu khổng lồ; nhưng chưa thực sự sẵn sàng đương đầu hay chuẩn bị tối ưu cho sự tấn công ồ ạt trên diện rộng của hacker nhắm vào tất cả các lĩnh vực như hàng không, ngân hàng… Bản thân những hệ thống Internet of Things cũng có nhiều giải pháp bảo mật khác nhau nhưng nhìn chung đều khá thô sơ.

Các vấn đề bảo mật như quyền riêng tư, kiểm soát truy cập, liên lạc an toàn và lưu trữ dữ liệu an toàn đang trở thành những thách thức lớn đối với hệ sinh thái IoT. Sự phát triển nhanh chóng của các thiết bị và dịch vụ IoT đã dẫn đến việc nhiều thiết bị được triển khai không an toàn và dễ bị khai thác bởi hacker.

Một số thách thức chính đối với thiết bị IoT được đề cập đến trong nghiên cứu của [6] là:

* Xác thực: Việc xác thực cho phép tích hợp các thiết bị IoT khác nhau được triển khai trong các bối cảnh khác nhau. Vấn đề chi phí khiến việc triển khai khóa mật mã hiệu quả và quản lý khóa mật mã là một thách thức trong xác thực thiết bị IoT.
* Kiểm soát truy cập và ủy quyền: Tuỳ vào thiết bị IoT mà có thể chỉ hỗ trợ một vài cơ chế để xác minh truy cập. Do đó, việc triển khai và quản lý các cơ chế kiểm soát truy cập và ủy quyền được điều chỉnh phù hợp với khả năng của từng thiết bị khác nhau là một thách thức trong hệ sinh thái IoT.
* Riêng tư: Việc xác định các thiết bị có thể có quyền truy cập vào thông tin cá nhân của người dùng được hay không là một thách thức lớn đối với hệ sinh thái IoT.
* Kiến trúc: Vấn đề không đồng nhất của các kiến trúc IoT là một trong những thách thức lớn đối với hệ sinh thái IoT. Việc xây dựng một kiến trúc giải quyết được các vấn đề bảo mật đã biết cũng như có khả năng đối phó với các thách thức mới không hề dễ dàng. Việc phát hiện các tác nhân độc hại đối với IoT như mã độc là nhiệm vụ rất khó khăn đối với các hệ thống phát hiện và ngăn chặn xâm nhập.
* Thách thức trong điều tra số: Hiện nay các phương pháp và công cụ phục vụ công tác thu thập bằng chứng số cũng như bảo quản bằng chứng số chưa phổ biến. Bên cạnh đó do tính chất tương tác thời gian thực và tự động giữa các thiết bị IoT, do đó khó khăn trong việc xác định được phạm vi thoả hiệp khi có tấn công xảy ra. Ngoài ra do bộ nhớ hạn chế, nên hầu như các thiết bị IoT hầu như không lưu trữ các metadata, kể cả dữ liệu tạm thời như thời gian sửa đổi, truy cập… khiến gần như không thể thu thập được mối tương quan của các bằng chứng được thu thập từ các thiết bị IoT khác nhau và đây trở thành một thách thức lớn cho công tác điều tra số.

Giới hạn về tài nguyên như bộ nhớ, CPU, cơ chế bảo mật trên nhiều thiết bị IoT như IP camera, router rất kém. Do đó, các mạng botnet là một mối đe doạ tiềm tàng đối với các thiết bị IoT ngày nay. Do đó, IoT trở thành một mối đe doạ tiềm tàng của hacker để chèn vào mã độc hơn so với các máy tính thông thường (desktop, laptop): [7]

* Nhiều thiết bị IoT sử dụng 24/7 mà không có cập nhật bản vá an ninh.
* Bảo mật thường không được chú trọng để giảm thiểu giá thành trong vòng đời phát triển thiết bị IoT. Chẳng hạn 70% IoT đều có lỗ hổng.
* Hạn chế tài nguyên, dẫn đến việc mã hoá trở nên xa xỉ.
* Nhiều thiết bị IoT sử dụng thông tin đăng nhập mặc định / yếu của nhà sản xuất. theo báo cáo của ESET, trong số những router được kiểm tra thì có khoảng 15% router sử dụng username và password yếu hoặc mặc định từ nhà sản xuất như “admin”, khoảng 20% router phơi port Telnet trên Internet.
* Thường các thiết bị IoT sử dụng hệ điều hành Linux, do đó tập trung vào file thực thi ELF.

Mã độc trên các thiết bị IoT được thiết kể để gây ra các tác động không mong muốn hoặc gây hại đến hệ thống máy tính hoăc thiết bị IoT. Có rất nhiều loại mã độc IoT như Trojan, Worms, Rootkits, Botnet… Trong đó botnet là một trong những mối đe doạ lớn nhất đối với các thiết bị IoT. Tấn công IoT botnet lên các các thiết bị như camera giám sát CCTV, đầu ghi DVR, smartTV hay các hệ thống tự động trong nhà đang trở nên phổ biến đối với hacker [7] do sự hiệu quả tấn công mang lại khi hacker có thể tạo ra một mạng botnet có khả năng đánh sập mạng cả một quốc gia [8].

## Mã độc IoT Botnet

Trong những năm gần đây, botnet đã được coi là một trong những mối đe dọa bảo mật chính trong số tất cả các loại phần mềm độc hại đang hoạt động trên Internet. Botnet đã không ngừng phát triển trên Internet toàn cầu ở cả quy mô và sự tinh vi của các kỹ thuật tấn công.

Thuật ngữ bot được bắt nguồn từ từ “robot” được thiết kế để thực hiện các chức năng đã được xác định, được lặp đi lặp lại và tự động. Ban đầu, các chương trình được phát triển như một tính năng hữu ích để thực hiện lặp đi lặp lại các hoạt động tiêu tốn thời gian, nhưng sau đó khả năng của chúng đã bị lạm dụng.

Kẻ tấn công tạo ra một chương trình độc hại và tìm cách lây nhiễm vào các thiết bị, mục đích biến các thiết bị này thành các bot dưới sự kiểm soát của hắn ta. Sau đó các bot này sẽ tiếp tục tự động tìm kiếm trong mạng các thiết bị dễ bị tấn công và không được bảo vệ để tiếp tục lây nhiễm. Sau khi phát tán và lây nhiễm, chúng có thể ẩn náu trong thiết bị nạn nhân một thời gian dài mà người dùng không hay biết.

Một lượng lớn các bot kết nối với nhau thông qua Internet tạo thành một mạng các bot, gọi là mạng botnet, thường được sử dụng để truyền phần mềm độc hại, gửi thư rác, ăn cắp thông tin nhạy cảm, lừa đảo, tạo các nhấp chuột ảo hoặc nghiêm trọng hơn là để thực hiện các cuộc tấn công mạng quy mô lớn, chẳng hạn như các cuộc tấn công từ chối dịch vụ.

Số lượng thiết bị trong mạng và sự phân tán là khác nhau trong các trường hợp. Nhiều thiết bị bị nhiễm mà không có nhận thức của chủ sở hữu, vì thiết bị đó vẫn có thể hoạt động được mặc dù nó là một phần của botnet.

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

A close up of text on a black background

Description automatically generated

* + - * 1. Các thành phần cơ bản của mạng IoT botnet

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.

### Mô hình hoạt động của một mạng IoT botnet cơ bản

Các IoT botnet có thể có kiến trúc, kích thước khác nhau nhưng có chung các giai đoạn trong vòng đời của mình.

Đặc trưng của một IoT botnet thường diễn ra theo một quy trình cơ bản gồm 04 giai đoạn như sau [7]:

A screenshot of a cell phone

Description automatically generated

* + - * 1. Vòng đời của một IoT botnet

Giai đoạn 1: Rò quét, tìm kiếm thiết bị tổn thương

* Một khi thiết bị IoT bị nhiễm mã độc và trở thành một bot dưới sự kiểm soát của hacker, nó sẽ thực hiện rò quét, tìm kiếm các thiết bị IoT khác trong mạng.
* Liên tục 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 2: Tấn công

* Cố gắng tấn công vét cạn thiết bị IoT sử dụng một bộ từ điển thông tin đăng nhập mặc định như root/root, admin/123,… Một khi đăng nhập thành công, nó sẽ gửi các thông tin về thiết bị IoT khai thác thành công này bao gồm địa chỉ IP, thông tin đăng nhập đến cho hacker.

Giai đoạn 3: Lây nhiễm

* Trong giai đoạn này, sau khi đã đăng nhập thành công vào thiết bị IoT, 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,…).

Giai đoạn 4: Bảo mạng botnet

Đảm bảo an toàn cho các bot cũng như mạng botnet là việc quan trọng và phải được duy trì suốt vòng đời 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).
* Sau đó cố gắng kích hoạt mã độc đã tải về ở giai đoạn 3 rên bộ nhớ tạm (ví dụ bộ nhớ RAM) của các thiết bị IoT và tạm dừng hoạt động để ẩn náu cho đến khi nhận được lệnh từ kẻ tấn công.

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 mạng IoT botnet: Mirai botnet [9]

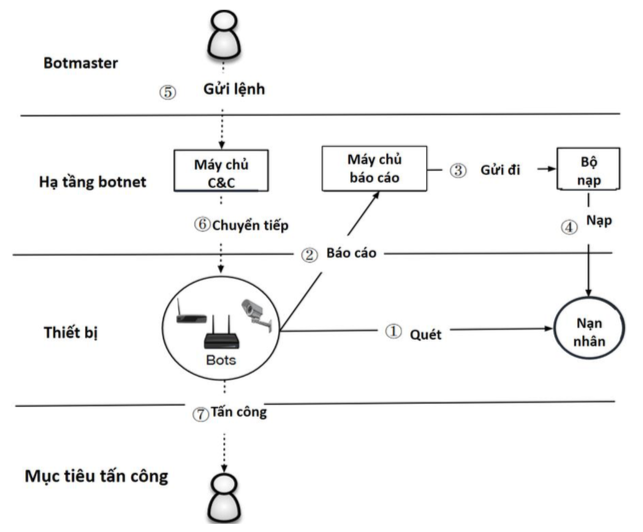
A screenshot of a social media post

Description automatically generated

* + - * 1. Các sự kiện chính của Mirai botnet

Đây là loại botnet được phát hiện khoảng tháng 8/2016 được xây dựng để hướng đến các thiết bị IoT (Internet of Things) chứa lỗ hổng hoặc bảo mật kém hoặc sử dụng mật khẩu/tài khoản quản trị mặc định. Có thể nói đây là mạng mã độc lớn nhất trên thế giới hình thành từ các thiết bị IoT, chuyên thực hiện các cuộc tấn công DDoS nhắm mục tiêu vào các tổ chức lớn như Dyn, Deutsche Telekom... gây ra các sự cố mất điện trên toàn diện rộng, gây thiệt hại lớn về tài chính của các cơ quan, tổ chức và quốc gia bị ảnh hưởng.

Mirai thực hiện rà quét dải mạng IPv4 nhằm tìm kiếm các thiết bị IoT sử dụng Telnet hoặc SSH và cố gắng đăng nhập bằng cách sử dụng một từ điển các tài khoản mặc định, tài khoản dễ đoán.



* + - * 1. Mô hình hoạt động của Mirai botnet

Mạng botnet Mirai hoạt động bao gồm 7 hoạt động chính:

1. Các bot trong mạng botnet sẽ thực thiện rà quét các thiết bị IoT mới và thực hiện tấn công từ điển vào các thiết bị nếu không telnet trực tiếp được. Trong mã nguồn của Mirai cho thấy Mirai có sử dụng 60 tài khoản, mật khẩu mặc định để thực hiện tấn công như root:root, admin:admin1234, guest:guest…

2. Sau khi đăng nhập thành công thiết bị IoT, bot gửi các thông tin gồm IP, cổng, tài khoản đăng nhập về máy chủ C&C để báo cáo.

3+4 . Lấy thông tin từ máy chủ báo cáo cấp cho bộ nạp để thực hiện đăng nhập tới thiết bị IoT, nạp và thực thi mã độc để lây nhiễm thiết bị mới. Sau khi lây nhiễm thành công, Mirai sẽ che giấu sự hiện diện của nó bằng cách xóa bỏ tệp tin nhị phân đã tải xuống và làm xáo trộn tên tiến trình thành một chuỗi ký tự chữ và số ngẫu nhiên. Do đó Mirai sẽ không tồn tại sau khi khởi động lại hệ thống.

5+6+7. Thiết bị sau khi bị lây nhiễm sẽ lắng nghe và nhận lệnh từ máy chủ C&C và thực thi lệnh đã được giao như thực hiện gửi thư rác, tạo các nhấp chuột ảo hay phát động chiến dịch tấn công từ chối dịch vụ vào mục tiêu xác định.

## Tổng quan về các phương pháp phát hiện IoT Botnet

### Phương pháp phân tích động

Trong phân tích động, các hành vi của tệp tin được theo dõi trong khi đang thực thi chương trình và phát hiện các mã độc chưa biết đến.

Một số công trình nghiên cứu phát hiện mã độc IoT botnet dựa trên phương pháp phân tích tĩnh được [10] kể đến như:

Celeda và cộng sự [11] 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 [12] 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 [13] 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 [14] 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 tuỳ 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 tuỳ 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 thâp dữ liệu làm đầu vào cho quá trình phân tích là rất phức tạp [10].

### Phương pháp phân tích tĩnh

Phân tích tĩnh hay còn gọi là phân tích dựa trên đặc trưng bao gồm phân tích, phát hiện mã độc và lỗ hổng bảo mật trong mã nguồn firmware hoặc các tập tin thực thi mà không cần thực thi chương trình.

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 [10] kể đến là của Costin và cộng sự [15], 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 [16].

### Phát hiện sử dụng phương pháp học sâu (Deep Learning)

Bên cạnh việc sử dụng các phương pháp phân tích tĩnh và phân tích động, phương pháp học sâu được sử dụng nhiều những năm gần đây, giúp giải quyết được các vấn đề phức tạp hơn, như phân tích ảnh hoặc hiểu được ngôn ngữ tự nhiên, trong khi học nông không thể đáp ứng đầy đủ.

Thay vì tạo thủ công một bộ quy tắc và thuật toán để trích xuất các đặc trưng từ dữ liệu thô, deep learning bao gồm việc học các đặc trưng này một cách tự động trong quá trình training. Trong deep learning bao gồm nhiều lớp (layers), dữ liệu từ lớp đầu vào (input layer) sẽ được đưa qua các lớp hidden layers, trong đó các lớp hidden layers cơ bản xử lý các khía cạnh cơ bản của vấn đề, các lớp hidden layers cao hơn dựa trên đầu ra của các lớp hidden layers cơ bản và xử lý các khía cạnh phức tạp hơn. Cuối cùng, lớp đầu ra (output layer) phân loại dữ liệu và đưa ra quyết định - đầu ra thu được ở output layer bị ảnh hưởng trực tiếp bởi mọi lớp khác có sẵn trong mạng. Quá trình này có thể được xem như là học phân cấp – mỗi lớp trong mạng sử dụng đầu ra của các lớp trước để làm đầu vào. [17]

Số lượng layers được sử dụng để xác định độ sâu của mô hình. Deep learning hiện nay thường bao gồm 10 hoặc thậm chí hàng trăm lớp hidden layers để học tự động.

Học sâu đã thành công trong các vấn đề chưa được giải quyết trước đây, điều này khá khó giải quyết bằng cách sử dụng máy học cũng như các mạng học nông khác. Điều đó đã thúc đẩy nhiều nghiên cứu ứng dụng học sâu, đặc biệt là ứng dụng học sâu trong phân tích phát hiện mã độc IoT botnet.

Một số công trình nghiên cứu phát hiện mã độc IoT botnet dựa trên phương pháp học sâu được [10] kể đến như:

Yuan và cộng sự [18] sử dụng hơn 200 đặc trưng từ quá trình phân tích tĩnh động làm đầu vào cho mạng học sâu DBN cho phép đạt được độ chính xác lên tới 96% trong việc phân loại mã độc và tệp tin lành tính.

Saxe và Berlin [19] đề xuất mô hình dựa trên mạng nơ-ron truyền thẳng để trích xuất các đặc trưng từ hơn 40000 tập tin nhị phân ứng dụng Windows, kết quả đạt được độ chính xác 95% với tỷ lệ dương tính giả (false positive rate) là 0.1%.

Hamed và cộng sự [20] đã đề xuất giải pháp sử dụng cấu trúc LSTM với RNN (Recurrent Neural Network) trong phát hiện mã độc trên thiết bị IoT dựa trên đặc trưng Opcode trích xuất từ các ứng dụng thực thi nền tảng ARM, độ chính xác đạt 98%.

Tuy nhiên nhược điểm của các nghiên cứu này mới áp dụng phương pháp học sâu vào phân tích dữ liệu thu thập được từ quá trình hoạt động của hệ thống, mà chưa khai thác những đặc thù của mã độc Botnet, lớp mã độc phổ biến nhất trên các thiết bị IoT [10].

## Tổng kết chương

Theo [7] và [21], 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. Bên cạnh đó, một đồ thị được sử dụng để hiện các hành vi của ứng dụng một cách hiệu quả, bởi vì các đồ thị, đặc biệt là các đồ thị có nhãn, có thể được sử dụng để mô hình hoá các mối quan hệ phức tạp giữa các dữ liệu.

Việc kết hợp phân tích tĩnh xác định mã độc IoT trong đó các đặc trưng được đồ thị hoá, với các gốc đồ thị là các đặc trưng - biểu diễn các hành vi của IoT botnet trong vòng đời IoT botnet, là cách tiếp cận đầy hứa hẹn. Các đặc trưng này được định nghĩa theo cả hành vi của IoT botnet và cả topo mạng của đồ thị, nó được sinh ra và xử lý thông qua deep learning [7].

# Ứng dụng thuật toán DGCNN và đồ thị cấu trúc PSI-graph trong phát hiện IoT botnet

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, thuật toán trong việc sinh ra đồ thị PSI, một cách tiếp cận mới trong việc mô tả cấu trúc của các tập tin thực thi ELF. Từ đồ thị PSI, thuật toán DGCNN sẽ được sử dụng nhằm huấn luyện trong việc phân tách giữa các tập tin lành tính và tập tin mã độc.

## Tổng quan

### Mạng nơ ron tích chập - Convolutional Neural Network (CNN) [17]

Trong một mạng nơ-ron truyền thống, các nơ-ron được kết nối đầy đủ giữa các lớp khác nhau. Các lớp nằm giữa lớp đầu vào (input layer) và lớp đầu ra (output layer) được gọi là các lớp ẩn (hidden layers). Mỗi hidden layer được tạo thành từ một số nơ-ron, trong đó mỗi nơ-ron được kết nối đầy đủ với tất cả các nơ-ron đứng trước. Tuy nhiên vấn đề đối với mạng nơ-ron này là kiến trúc không phù hợp với đầu vào là các dữ liệu lớn. Khi đó, mạng nơ-ron tích chập CNN được đề xuất giúp giải quyết vấn đề này.

CNN là được thiết kế để xử lý cá dữ liệu như dữ liệu hình ảnh, tín hiệu giọng nói và video. Một mạng CNN điển hình bao gồm các lớp dưới đây, trong đó mỗi lớp thực hiện các nhiệm vụ khác nhau:

* Convolutional layer (lớp tích chập),
* Activation function layer,
* Pooling layer,
* Fully connected layer (lớp kết nối đầy đủ),
* Dropout layer.

Các lớp Convolutional layer và Activation function layer thường được xếp chồng lên nhau, theo sau là lớp Pooling layer (tuỳ chọn). Lớp Fully connected layer là lớp cuối cùng của mạng và đầu ra của lớp Fully connected layer cuối cùng tạo ra điểm số (score) để đưa ra quyết định đối với của dữ liệu đầu vào.

#### Convolutional layer

Convolution layer được coi là thành phần cốt lõi của một mạng nơ ron tích chập. Convolution layer sử dụng các phép toán chập thay cho phép nhân ma trận tổng quát. Các tham số của nó bao gồm một tập hợp các bộ lọc (filter) có thể learning. Nhiệm vụ chính của Convolution layer là xác định các đặc trưng (features) được tìm thấy trong các vùng cục bộ (local regions) của hình ảnh đầu vào (input image). Các đặc trưng này sau đó sẽ được áp dụng trên toàn bộ tập dữ liệu (dataset) và ánh xạ sự xuất hiện của chúng vào một bản đồ đặc trưng (feature map).

Một bản đồ đặc trưng (feature map) thu được bằng cách áp dụng lặp đi lặp lại filter trên các tiểu vùng subregions của hình ảnh đầu vào, tức là, tích chập filter với input image, thêm một bias, sau đó áp dụng activation function. Vùng đầu vào mà filter được áp dụng được gọi là trường tiếp nhận cục bộ (local receptive field). Kích thước của trường local receptive field giống như kích thước của filter.

A picture containing crossword

Description automatically generated

* + - * 1. Ví dụ phép tích chập

Hìnhtrên đâycho thấy cách một bộ lọc (hình chữ T) được tích chập với đầu vào để có được bản đồ tính năng (feature map).

Bản đồ đặc trưng đạt được sau khi thêm một bias và sau đó áp dụng hàm phi tuyến nonlinear function cho đầu ra của phép toán chập. Mục đích của hàm phi tuyến là giới thiệu tính phi tuyến trong mô hình CNN và có một số hàm phi tuyến có sẵn được giải thích ngắn gọn trong phần tiếp theo.

#### Activation function layer

Đầu ra của mỗi lớp chập được đưa đến một lớp Activation function layer. Activation function layer bao gồm một hàm activation với đầu vào là feature map được tạo bởi lớp chập và đầu ra là activation map, nhằm đảm bảo tính chất phi tuyến trong mô hình. Dưới đây là các hàm activation phổ biến:

* Sigmoid Activation Function

|  |  |
| --- | --- |
| A picture containing text, sitting, black, white  Description automatically generated | A picture containing clock  Description automatically generated |

* + - * 1. Hàm Sigmoid Activation Function
  + Kết quả cuối cùng được biến đổi về khoảng giá trị [0,1]
  + Có thể gặp trường hợp “vanishing gradient” khi huấn luyện mạng
* Tanh activation function

|  |  |
| --- | --- |
| A picture containing text, sitting, white, black  Description automatically generated | A picture containing clock  Description automatically generated |

* + - * 1. Hàm Tanh Activation Function
  + Kết quả cuối cùng được biến đổi về khoảng giá trị [-1,1]
  + Có thể gặp trường hợp “vanishing gradient” khi huấn luyện mạng
* Softmax activation function

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated | A picture containing clock  Description automatically generated |

* + - * 1. Hàm Softmax Activation Function
  + Mỗi giá trị này trong khoảng [0,1] và tổng tất cả giá trị bằng 1 nên được sử dụng để mô hình các phân phối xác suất.
  + Nó thường được sử dụng trong lớp đầu ra của mạng nơ-ron nhằm mục đích phân loại.
* ReLU activation function

ReLU là hàm phổ biến nhất cho các mạng nơ ron sâu (deep neural network). Qúa trình training mạng nơ-ron với ReLU nhanh hơn nhiều so với sử dụng các hàm activation khác như sigmoid và tanh. ReLU chỉ đơn giản là tính toán bằng cách đặt ngưỡng đầu vào bằng 0. Nói cách khác, nếu input nhỏ hơn 0 thì sẽ được chỉnh về giá trị 0, ngược lại sử dụng giá trị thô.

|  |  |
| --- | --- |
| A picture containing object, antenna  Description automatically generated | A picture containing knife  Description automatically generated |

* + - * 1. Hàm ReLU Activation Function
  + Range: 0 to infinity
  + Kết quả cuối cùng có thể rất lớn
  + Vì nó lọc giá trị âm nên có thể bỏ qua những thông tin quan trọng.
  + Gradient có thể tiến về 0 nên các giá trị weights không được cập nhật (“dying ReLU”)

#### Pooling layer

Trong mạng CNN, trình tự của lớp convolution và lớp activation được theo sau bởi một lớp pooling để giảm kích thước không gian của đầu vào và do đó giảm số lượng tham số trong mạng.

Từ feature map – output của lớp convolution, lớp pooling sẽ cho ra một feature map mới với kích thước nhỏ hơn nhưng vẫn giữ được các đặc trưng nhất của feature map convolution.

Có một số kỹ thuật pooling có sẵn và kỹ thuật pooling phổ biến nhất là max-pooling.

A picture containing clock

Description automatically generated

* + - * 1. Max-pooling

Max-pooling đơn giản chỉ xuất ra giá trị tối đa trong vùng đầu vào (input region). Vùng đầu vào là tập con của đầu vào. Ví dụ: nếu vùng đầu vào có kích thước 2 × 2, Max-pooling sẽ xuất ra 4 bốn giá trị lớn nhất như trong hình.

Lớp pooling nhằm loại bỏ các dữ liệu ít quan trọng hơn nhưng vẫn giữ được các đặc trưng. Lý do đằng sau hoạt động pooling là việc phát hiện đặc trưng quan trọng hơn vị trí chính xác của đặc trưng.

#### Fully connected layer

CNN gồm hai giai đoạn: giai đoạn trích xuất đặc trưng và giai đoạn phân loại. Trong CNN, stack của các lớp convolution và pooling đóng vai trò là giai đoạn trích xuất đặc trưng, trong khi giai đoạn phân loại bao gồm một hoặc nhiều lớp kết nối đầy đủ (fully connected layer) theo sau là hàm softmax.

Quá trình convolution và pooling tiếp tục cho đến khi phát hiện đủ các đặc trưng. Bước tiếp theo là đưa ra quyết định dựa trên các đặc trưng được phát hiện này. Điều này được thực hiện bằng cách thêm một hoặc nhiều fully connected layers vào cuối. Trong fully connected layer, mỗi nơ-ron từ lớp trước (lớp convolution hoặc lớp pooling hoặc lớp fully connected layer) được kết nối với mọi nơ-ron ở lớp tiếp theo và mọi giá trị đều góp phần đưa ra quyết định cuối cùng.

Hình dưới thể hiện liên kết giữa một lớp convolution và một lớp fully connected layer. Giống như các lớp convolution, các lớp fully connected layer có thể được xếp chồng lên nhau để tìm hiểu các tổ hợp đặc trưng phức tạp hơn nữa. Đầu ra của lớp fully connected layer cuối cùng được đưa đến một bộ phân loại để tính điểm số cho từng lớp (class score). Softmax và Support Vector Machines (SVM) là hai phân loại chính được sử dụng trong CNN. Softmax tạo xác suất cho mỗi lớp với tổng xác suất là 1; SVM tạo ra điểm số và lớp có điểm cao nhất được coi là lớp đúng.

A close up of a logo

Description automatically generated

*Hình: Liên kết giữa một lớp convolution và một lớp fully connected layer*

#### Dropout

Deep CNN bao gồm nhiều lớp ẩn (hidden layers) cho phép nó tìm hiểu các đặc trưng phức tạp hơn. Nó được theo sau bởi các lớp được kết nối đầy đủ để đưa ra quyết định. Một lớp được kết nối đầy đủ được kết nối với tất cả các đặc trưng và nó có xu hướng bị overfitting. Overfitting cập đến vấn đề khi một mô hình được đào tạo và nó hoạt động rất tốt trên dữ liệu đào tạo đến mức nó ảnh hưởng tiêu cực đến hiệu suất của mô hình trên dữ liệu mới. Để khắc phục vấn đề overfitting, một lớp dropout có thể được đưa vào mô hình trong đó một số tế bào thần kinh cùng với các kết nối của chúng được drop ngẫu nhiên khỏi mạng trong quá trình đào tạo. Một mạng giảm (reduced network) được để lại; các cạnh đến và đi đến một nút bị loại bỏ cũng bị loại bỏ. Chỉ có mạng giảm được đào tạo về dữ liệu trong giai đoạn (stage) đó. Các nút bị loại bỏ sau đó được gắn lại (reinsert) vào mạng với trọng số ban đầu (original weight) của chúng. Dropout làm giảm overfitting và cải thiện việc khái quát hóa/tổng quan hoá (generalization) mô hình.

|  |  |
| --- | --- |
| A picture containing scissors, table  Description automatically generated | |
| * + - * 1. Mạng nơ-ron đơn giản | * + - * 1. Mạng nơ-ron sau  khi dropout |

### Đồ thị cấu trúc PSI-graph

Theo [7] và [21], 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.

#### Đồ thị đặc trưng phi cấu trúc

Phát hiện mã độc bằng cách phân tích đồ thị các đặc trưng phi cấu trúc như Processor Information, opcodes.string, system call (thể hiện trong file header)…..

Tuy nhiên việc phát hiện mã độc dựa trên các đặc trưng này phụ thuộc vào giá trị của các đặc trưng này (ví dụ: system call inet\_toa) và các đặc trưng này không thể mô tả các thông tin có ngữ nghĩa phức tạp (như các dữ liệu lệ thuộc vào trong vòng đời của IoT botnet). Ngoài ra, phương pháp đồ thị hoá các đặc trưng phi cấu trúc thường không hiệu quả đối với các mã độc có sử dụng các kỹ thuật obfuscate như mã hóa, garbage insertion. [7]

#### Đồ thị đặc trưng có cấu trúc

Là việc tập trung vào đồ thị các đặc trưng có cấu trúc như: control flow graph, call graph, code graph để biểu diễn hành vi của các tệp thực thi do đồ thị là phương pháp hiệu quả để mô hình hoá các mối quan hệ phức tạp của dữ liệu.

[7] đã kể ra một số nghiên cứu đem lại kết quả phát hiện tốt như: Zhao và cộng sự đề xuất phương pháp phát hiện dựa trên đồ thị CFG-graph (accuracy = 96,8% với Random Forest); Zhiwu Xu và cộng sự đề xuất một phương pháp để phát hiện phần mềm độc hại Android dựa trên đồ thị kết hợp CFG-DFG-graph và CNN trong phân lớp (accuracy = 99,82%, F1= 99,191%). Tuy nhiên, đồ thị các đặc trưng không phải lúc nào cũng là single connected graph, vì vậy nhiều nhà nghiên cứu đã đề xuất sử dụng các đồ thị con (subgraph) để khai thác sâu hơn về hành vi của mã độc. Karbalaie và cộng sự đã đề xuất xây dựng một đồ thị biểu thị hành vi của mã độc và thực hiện khai thác sơ đồ con của nó trước khi sử dụng các thuật toán phân lớp (accuracy = 96,6%) . Từ đồ thị CFG-graph, Silvio Cesare và Yang Xiang đã loại bỏ các cycle (do mã độc cần phải thực hiện các tác vụ lặp đi lặp lại như kết nối C&C, tấn công,… nên đối với các sơ đồ con mà xuất hiện 1 lần trong đồ thị – cycle – không phải là hành vi độc hại) và thực hiện duyệt qua tất cả các đường dẫn của đồ thị để tạo ra các đồ thị con, sau đó thực hiện vector hoá đồ thị con này.

Hầu hết các nghiên cứu được đề cập ở trên là để phát hiện mã độc trên nền tảng PC và nền tảng di động mà không phát hiện các thiết bị IoT. Bên cạnh đó các nghiên cứu thường gặp một số hạn chế về tài nguyên và với các mã độc có sử dụng biện pháp obfuscate.

Thừa kế các nghiên cứu đã có, để cải thiện độ chính xác trong phát hiện mã độc [7][16] và [22] đã đề xuất tập trung vào kết hợp PSI-graph và CNN.

## Phương pháp đề xuất

IoT botnet thực thi step-by-step và phần lớn chúng thường sử dụng các thông tin quan trọng như địa chỉ IP address, URL, Domain Name,… Các chuỗi thông tin này, có thể ở dạng mã hoá, hoặc ở dạng bản rõ hoặc bị làm rối, được gọi là PSI (Printable String informaton). Các PSI này thường chứa các thông tin quan trọng từ đó phản ánh ý đồ và mục tiêu của hacker như chuỗi PSI *“/dev/watchdog; /dev/misc/watchdog”* luôn có trong mã độc Linux.Mirai cho thấy botnet đang cố gắng để ngăn cản việc reboot lại thiết bị. Do nó PSI thường được sử dụng trong phân tích tĩnh để xác định liệu một file ELF có là file độc hại hay không. [16]

A picture containing screenshot

Description automatically generated

* + - * 1. Mô hình đề xuất

Mô hình [7] đề xuất bao gồm 4 tác vụ:

1. Trích xuất đồ thị có cấu trúc PSI-graph từ các tập tin ELF đầu vào.
2. Từ đồ thị có cấu trúc PSI-graph, sử dụng thuật toán duyệt đồ thị để trích xuất ra các đồ thị con PSI-rooted.
3. Tiền xử lý dữ liệu thô từ đồ thị con PSI-rooted:
   * Trích xuất các đặc trưng để thực hiện vector hoá dữ liệu
   * Chuẩn hóa dữ liệu
   * Lựa chọn ra các đặc trưng thích hợp nhất để tối ưu phương pháp đề xuất.
4. Sử dụng một số thuật toán phân lớp trong học máy để đưa ra quyết định tập tin đầu vào là lành tính hay độc hại.

### Trích xuất đồ thị có cấu trúc PSI-graph từ các tập tin ELF đầu vào

[16] và [10] đề xuất sử dụng công cụ IDA pro để trích xuất đồ thị CFG, bởi IDA (Interactive Disassembler) là công cụ phân tách có khả năng thực hiện dịch ngược và tự động phân tích các các file thực thi nhị phân. Tuy nhiên, đồ thị CFG thu được luôn có cấu trúc phức tạp và sự liên kết giữa các giá trị dạng chuỗi trong các hàm của tập tin nhị phân đầu vào khó quan sát, đồng thời việc áp dụng các kỹ thuật học sâu cũng mất nhiều thời gian. Chính vì vậy, [1] và [4] đã đề xuất sử dụng đồ thị PSI thay vì sử dụng đồ thị CFG.

Từ file ELF đầu vào, khám phá tất cả các hàm liên quan có chứa PSI. Từ đó làm cơ sở để trích xuất đồ thị có hướng PSI-graph: *GPSI (V, E),* trong đó:

- *V* là một tập hợp các đỉnh, với mỗi đỉnh là một phần tử của tập PSI.

- *E* là một tập hợp các cạnh có hướng.

Tuy nhiên, [16] chỉ tập trung vào cấu trúc của đồ thị PSI mà không đi vào khai thác các hành vi độc hại trong vòng đời của IoT botnet. Do đó, để khai thác được các hành vi liên quan đến vòng đời của mã độc IoT botnet, [7] đã đề xuất sinh đồ thị con PSI-rooted với mỗi nút gốc là một phần tử của đồ thị PSI-Graph, và mỗi hành vi tương ứng với một đồ thị con PSI-rooted.

### Trích xuất các đồ thị con PSI-rooted

Phương pháp dựa trên đồ thị PSI-graph cho thấy các tệp ELF thực sự được thực thi như thế nào. Nhưng không phải tất cả các hành vi của một tệp tin độc hại đều là độc hại mà sẽ bao gồm những hoạt động rất bình thường như bất kỳ chương trình lành tính nào khác.

Do đó, việc phân tích các đường đi và nút của đồ thị PSI-graph được coi là một công việc quan trọng để xác định được hành vi đó có thực sự độc hại hay không. Để thực hiện việc phân tích các đường đi và nút của đồ thị PSI-graph, [7] đã đề xuất đồ thị con PSI-rooted.

*Gsg = (V’, E’, v, d)* đại diện cho một đồ thị con có hướng PSI-rooted được sinh ra từ đồ thị có hướng PSI-graph *GPSI=(V, E, λ)* trong đó:

* + *v* là nút gốc của đồ thị con PSI-rooted
  + *d* là độ sâu (degree) của đồ thị con PSI-rooted
  + *V’* ⊂ *V*: là tập các đỉnh của đồ thị con PSI-rooted thoả mãn *0 ≤ (v, Vi’) ≤ d*
  + *E’* ⊂ *E* : là tập các cạnh có hướng giữa các nút trong *V’*.

[7] đề xuất thuật toán để sinh đồ thị PSI-rooted như sau:

A screenshot of a cell phone

Description automatically generated

Trong đó, hàm *GETWLSUBGRAPH* sử dụng thuật toán được thừa hưởng từ đề xuất của Annamalai Narayanan, để duyệt tất cả nhánh của đồ thị PSI-graph nhằm trích xuất đồ thị con PSI-rooted theo dạng BFS (Breadth First Search). BSF thực hiện duyệt cây đồ thị bắt đầu từ gốc đồ thị và đi qua tất cả các nút liền kề ở độ sâu hiện tại trước khi duyệt sang các nút ở độ sâu tiếp theo, trong khi DFS (Deepth First Search) duyệt cây đồ thị bắt đầu từ gốc đồ thị và khai thác càng sâu càng tốt mỗi nhánh đồ thị trước khi quay lại.

|  |  |
| --- | --- |
| Order in which the nodes get expanded | Order in which the nodes get expanded |
| * + - * 1. Duyệt cây đồ thị BFS | * + - * 1. Duyệt cây đồ thị DFS |

Với độ sâu cố định của đồ thị con, BSF sẽ phù hợp hơn trong việc trích xuất đồ thị con PSI-graph so với DSF. Từ một đồ thị PSI-graph, ta sẽ có được đồ thị con PSI-rooted, trong đó đồ thị con PSI-rooted với đỉnh là một phần tử của đồ thị PSI-graph ban đầu. Một ví dụ trực quan được đưa ra như dưới đây, thực hiện duyệt đồ thị PSI-graph để trích xuất ra đồ thị con PSI-rooted với đỉnh 11, độ sâu bằng 2.

|  |  |
| --- | --- |
| A close up of a piece of paper  Description automatically generated | A screenshot of a cell phone  Description automatically generated |

Qua quá trình duyệt đồ thị PSI-graph, tại nhánh đồ thị với đỉnh 11, thu được đồ thị con PSI-rooted bao gồm { 11, 0, 8, 10, 7, 9, 18, 0, 0, 7, 0, 5, 6, 15, 16}. Thực hiện tương tự với toàn bộ đồ thị PSI-graph để thu được một danh sách các đồ thị con PSI-rooted với gốc đồ thị là các phần tử khác của đồ thị PSI-graph.

Sau khi thu được danh sách các đồ thị con PSI-rooted, tiến hành lựa chọn ra các đồ thị con có chứa các hành vi đặc trưng trong vòng đời của IoT botnet bằng tác vụ phân lớp.

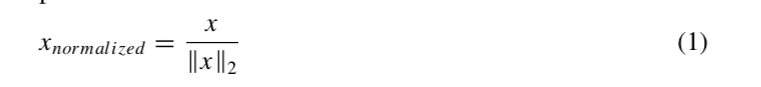
### Tiền xử lý dữ liệu - Data preprocessing phase

Đầu ra của tác vụ trích xuất đồ thị con PSI-rooted là một đồ thị, do đó tác vụ tiền xử lý dữ liệu giúp vector hoá các dữ liệu phức tạp của đồ thị con PSI-rooted.

Để phục vụ tác vụ này, một số phương pháp chuyển đổi được kể đến như Word2vec, Doc2vec, Graph2vec và Subgraph2vec. Những phương pháp đó đều liên quan đến một mô hình ngôn ngữ gọi là Skipgram. Tuy nhiên, Skipgram mất nhiều thời gian và yêu cầu nhiều tài nguyên phục vụ cho quá trình learning. Do đó, [7] đề xuất lược đồ chuyển đổi đồ thị thành các vector dạng số học, trong đó coi mỗi đồ thị là một tài liệu (document) và đồ thị con gốc của nó là một từ (word) của tài liệu này, khi đó có thể thực hiện đếm số lần xuất hiện của các word trong mỗi document. Và mỗi khi đồ thị con PSI-rooted xuất hiện thì PSI-rooted đó được coi là một đặc trưng độc hại tương ứng với hành vi độc hại trong vòng đời của mã độc. Do đó, dữ liệu có thể được biểu diễn bằng một ma trận với hàng dựa trên đồ thị PSI-graph và cột dựa trên đồ thị con PSI-rooted tương ứng.

Kết quả thử nghiệm của [7] cho thấy lược đồ giảm đáng kể thời gian (giảm thời gian learning xuống khoảng 30 phút so với 12-15h) so với thuật toán của Annamalai Narayanan (sử dụng Skipgram), không yêu cầu quá nhiều tài nguyên tính toán và độ tin cậy giảm không đáng kể.

Ma trận nhận được bao gồm các các đặc trưng có kích thước khác nhau, do đó cần chuẩn hóa (normalization) các đặc trưng với một định mức chuẩn (unit norm). Normalization là quá trình mở rộng (scaling) các đặc trưng riêng lẻ để có một định mức chuẩn. Scaling đầu vào thành một định mức chuẩn (unit norm) là một hoạt động phổ biến trong phân lớp văn bản hoặc phân cụm. Do coi đồ thị con PSI-rooted là một word trong document, do đó normalization là việc cần thiết để tránh kết quả không mong muốn khi dữ liệu được đưa đến task phân lớp. Công thức normalization được mô tả như sau:



Trong đó x là phần tử của bộ dữ liệu đồ thị con PSI-rooted.

Tuy nhiên số lượng đồ thị con PSI-rooted khác nhau trong toàn bộ tập dữ liệu PSI-graph là rất lớn (530155 đồ thị con PSI-rooted khác nhau được trích xuất trong tập dữ liệu của [7]), nhưng không phải mọi đồ thị con PSI-rooted đều cần thiết cho tác vụ phân lớp. Chẳng hạn, có nhiều đồ thị con PSI-rooted liên quan đến hành vi độc hại và lành tính. Do đó, việc tối giản tập dữ liệu thành một tập hợp các đặc trưng có giá trị là cần thiết trong giảm dung lượng bộ nhớ và khả năng tính toán cần thiết cho task phân lớp và giúp tiết kiệm thời gian cho quá trình training.

Sau bước này, nhận được dữ liệu chuẩn bị sẵn sàng cho các nhiệm vụ phân lớp.

### Tác vụ phân lớp

Tác vụ tiền xử lý dữ liệu cho ra một bộ dữ liệu với các đặc trưng có giá trị. Tập dữ liệu này sẵn sàng để được làm đầu vào cho tác vụ phân lớp.

[16] đề xuất sử dụng DGCNN trong phân lớp các dữ liệu PSI-graph bao gồm 3 phần:

1. Lớp tích chập: trích xuất các đỉnh đặc trưng và xác định thứ tự các đỉnh sao cho nhất quán.

2. Lớp SortPooling: sắp xếp các đỉnh đặc trưng theo thứ tự đã được xác định và thống nhất kích thước đầu vào.

3. Đưa ra quyết định phân lớp.

## Tổng kết chương

# Thực nghiệm kết quả

Chương 3 trình bày cấu hình môi trường và đánh giá kết quả thử nghiệm và đánh giá độ chính xác và hiệu quả của phương pháp đề xuất trong việc phân tích phát hiện mã độc IoT botnet.

## Giới thiệu bộ dữ liệu IoT botnet

## Kết quả thực nghiệm, đánh giá độ chính xác và hiệu quả

## Tổng kết chương

# KẾT LUẬN

* Tóm tắt những kết quả đã làm được
* Đánh giá, đề xuất hướng phát triển

# TÀI LIỆU THAM KHẢO

[1] P. Liu and S. Fabre, “Market Trends : Strategies Communications Service Providers Can Use to Address Key 5G Security Challenges,” no. October, 2019.

[2] J. Guth, U. Breitenbucher, M. Falkenthal, F. Leymann, and L. Reinfurt, “Comparison of IoT platform architectures: A field study based on a reference architecture,” *2016 Cloudification Internet Things, CIoT 2016*, 2017, doi: 10.1109/CIOT.2016.7872918.

[3] M. Bauer and S. Lange, *Enabling Things to Talk*. 2013.

[4] P. Fremantle, “A reference architecture for the internet of things,” *WSO2 White Pap.*, no. October 2015, p. 21, 2014, doi: 10.13140/RG.2.2.20158.89922.

[5] Cisco, “The Internet of Things Reference Model,” *Nat. Phys.*, vol. 10, no. 7, p. 538, 2014, doi: 10.1038/nphys3028.

[6] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things security and forensics: Challenges and opportunities,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 544–546, 2018, doi: 10.1016/j.future.2017.07.060.

[7] H. T. Nguyen, Q. D. Ngo, D. H. Nguyen, and V. H. Le, “PSI-rooted subgraph: A novel feature for IoT botnet detection using classifier algorithms,” *ICT Express*, no. xxxx, 2020, doi: 10.1016/j.icte.2019.12.001.

[8] Swati Khandelwal, “Someone is Using Mirai Botnet to Shut Down Internet for an Entire Country,” 2016. [Online]. Available: https://thehackernews.com/2016/11/ddos-attack-mirai-botnet.html.

[9] M. Antonakakis *et al.*, “Understanding the Mirai Botnet This paper is included in the Proceedings of the Understanding the Mirai Botnet,” *USENIX Secur.*, pp. 1093–1110, 2017.

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

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

[12] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti, “Avatar: A Framework to Support Dynamic Security Analysis of Embedded Systems’ Firmwares,” no. February, pp. 23–26, 2014, doi: 10.14722/ndss.2014.23229.

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

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

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

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

[17] M. Defferrard, *Advances in Deep Learning on Graphs*. 2018.

[18] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, “Droid-Sec: Deep learning in android malware detection,” *Comput. Commun. Rev.*, vol. 44, no. 4, pp. 371–372, 2015, doi: 10.1145/2619239.2631434.

[19] J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” *2015 10th Int. Conf. Malicious Unwanted Software, MALWARE 2015*, pp. 11–20, 2016, doi: 10.1109/MALWARE.2015.7413680.

[20] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, “A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting,” *Futur. Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018, doi: 10.1016/j.future.2018.03.007.

[21] S. Chowdhury *et al.*, “Botnet detection using graph-based feature clustering,” *J. Big Data*, vol. 4, no. 1, 2017, doi: 10.1186/s40537-017-0074-7.

[22] H. T. Nguyen, D. H. Nguyen, Q. D. Ngo, V. H. Tran, and V. H. Le, “Towards a rooted subgraph classifier for IoT botnet detection,” *ACM Int. Conf. Proceeding Ser.*, pp. 247–251, 2019, doi: 10.1145/3348445.3348474.