HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Trần Đình Tân

ỨNG DỤNG THUẬT TOÁN ONE-CLASS SVM TRONG PHÁT HIỆN BOTNET TRÊN CÁC THIẾT BỊ IOT

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

TÓM TẮT LUẬN VĂN THẠC SĨ

HÀ NỘI - 2019

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: TS. Ngô Quốc Dũng

(Ghi rõ học hàm, học vị)

Phản biện 1: PGS. TS Nguyễn Hà Nam

Phản biện 2: PGS.TS Nguyễn Mạnh Hùng.

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 9 giờ 00 ngày 11 tháng 01 .. năm 2020

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông

MỞ ĐẦU

1. Lý do chọn đề tài

Trong cuộc cách mạng công nghiệp 4.0, Internet của vạn vật (Internet of Things - IoT) là một xu hướng công nghệ mới đang được phát triển rất mạnh mẽ làm thay đổi cách sống và cách làm việc của con người. Tuy nhiên, càng nhiều thiết bị được kết nối với nhau để chia sẻ thông tin thì đồng nghĩa với việc càng xuất hiện thêm nhiều lỗ hổng bảo mật đe dọa sự an toàn của chính các thiết bị IoT. Bên cạnh đó, nhiều chuyên gia an ninh mạng đánh giá các cuộc tấn công mạng vào các thiết bị IoT sẽ để lại hậu quả nghiêm trọng hơn so với các cuộc tấn công vào hệ thống máy tính thông thường. Theo số liệu tính đến đầu năm 2018 của Kaspersky Lab cho biết tổng số mẫu phần mềm độc hại nhắm đến các thiết bị IoT được họ phát hiện đã lên tới hơn 7.000, trong đó hơn một nửa xuất hiện chỉ trong năm 2017. Hầu hết các cuộc tấn công nhắm vào máy ghi hình kỹ thuật số hoặc máy quay IP (chiếm 63%), và 20% là vào các thiết bị mạng, gồm router, modem ... Khoảng 1% mục tiêu là các thiết bị quen thuộc nhất của người dùng như máy in và thiết bị gia đình thông minh khác.

Các mã độc nói chung và mã độc trên các thiết bị IoT nói riêng đều có rất nhiều biến thể vì vậy việc phát hiện rất khó khăn. Việc thu thập mã độc đã và đang được thực hiện thông qua các hệ thống HoneyPot cho các thiết bị IoT như IoTPot, Detux... Tuy nhiên, việc thu thập các tệp tin lành tính để từ đó áp dụng các thuật toán học máy nhằm phân biệt, phát hiện các tệp tin mã độc lại chưa có nhiều. Để thực

hiện việc phân biệt giữa các tệp tin mã độc và lành tính trong điều kiện số lượng tệp tin giữa hai lớp mã độc/lành tính chênh lệch lớn thì việc sử dụng các thuật toán học máy 1 lớp trở nên cần thiết. Thuật toán One-class SVM đã được ứng dụng nhiều vào các bài toán phân lớp mã độc và cũng đã được chứng minh có hiệu quả trong việc phát hiện các mã độc thông thường. Từ lý đó và thực tiễn đảm bảo an ninh mạng cho các thiết bị IoT em đề xuất đề tài luận văn: "**Ứng dụng thuật toán One-class SVM trong phát hiện botnet trên các thiết bị IoT**".

2. Tổng quan về vấn đề nghiên cứu

Hiện nay, trên thế giới đã có nhiều công trình nghiên cứu về botnet trên các thiết bị IoT, trong đó điển hình là công trình nghiên cứu của nhóm tác giả Vitor Hugo Bezerra và các thành viên công bố vào năm 2018, với tiêu đề: One-class Classification to Detect Botnets in Iot a devices. Trong công trình nghiên cứu này nhóm tác giả đã xây dựng mô hình phát hiện botnet và chạy thử nghiệm trên thiết thị Rasperrypi, các bước tiến hành như sau: cài đặt công cụ thu thập dữ liệu trên thiết bị IoT; thu thập dữ liệu; chuẩn hóa dữ liệu thu thập; trích xuất đặc trưng; training model; vận hành thử nghiệm. Các kết quả đạt được rất khả quan, tuy nhiên tập dataset của nhóm tác giả chỉ có các mẫu mã độc, không có các mẫu sạch nên tập dataset bị lệch dẫn đến kết quả nhận diện không được cao. Bên cạnh đó tác giả chỉ mới thử nghiệm mô hình trên thiết bị Raspberry pi.

Tại Hội thảo quốc gia lần thứ XX: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông năm 2017 diễn ra tại Quy Nhơn,

nhóm tác giả Lê Hải Việt và các thành viên đã công bố bài báo: Xây dựng mô hình phát hiện mã độc trên thiết bị định tuyến bằng tác tử. Trong bài báo này, nhóm tác giả mới chỉ đề xuất giải pháp phát hiện botnet trong các thiết bị router mà chưa đề cập đến các thiết bị IoT khác.

3. Mục đích nghiên cứu

Xây dựng và thử nghiệm mô hình phát hiện botnet trên các thiết bị IoT bằng thuật toán One-class SVM.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

- Thuật toán one-class, SVM, one-class SVM;

- Các thiết bị IoT;

- Botnet trên các thiết bị IoT.

Phạm vi nghiên cứu:

- Hiện nay, có rất nhiều chủng loại thiết bị IoT, tuy nhiên, trong phạm vi nghiên cứu của đề tài này chỉ tập trung vào các thiết bị IoT dân dụng. Các thuật toán học máy sẽ sử dụng đặc trưng System-call Graph với bộ dữ liệu từ bộ IoTPot gồm 4000 mẫu IoT botnet và thu thập thêm từ các nguồn khác như: Virusshare,…

5. Phương pháp nghiên cứu

- Phương pháp nghiên cứu lý thuyết: Đọc và phân tích tài liệu về các thuật toán học máy;

- Phương pháp thực nghiệm: Xây dựng và thử nghiệm mô hình áp dụng thuật toán one-class SVM trong phát hiện botnet trên các thiết bị IoT.

6. Nội dung

Cấu trúc của luận văn sẽ bao gồm 3 chương, cụ thể như sau:

CHƯƠNG 1: TỔNG QUAN VỀ MÃ ĐỘC IOT BOTNET VÀ CÁC BIỆN PHÁP PHÁT HIỆN

Chương này sẽ trình bày kiến thức tổng quan về các thuật toán học máy: one-class; SVM; one-class SVM và trình bày về phát hiện botnet trong các thiết bị IoT.

CHƯƠNG 2: XÂY DỰNG MÔ HÌNH PHÁT HIỆN IOT BOTNET

Chương này trình bày về việc áp dụng thuật toán học máy one-class SVM vào trong việc xây dựng mô hình phát hiện botnet trong các thiết bị IoT.

CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương này trình bày về các bước cài đặt mô hình, thử nghiệm, từ kết quả thu được đưa ra những nhận xét, đánh giá.

CHƯƠNG 1: TỔNG QUAN MÃ ĐỘC IOT BOTNET VÀ CÁC BIỆN PHÁP PHÁT HIỆN

1.1. Tổng quan về mã độc IoT Botnet

1.1.1. Tổng quan về thiết bị IoT dân dụng

Các thiết bị IoT dân dụng hiện nay phần lớn bao gồm các thiết bị định tuyến, IP Camera và các thiết bị Smartbox-TV. Tác giả lựa chọn trình bày chi tiết về kiến trúc của thiết bị định tuyến. Thiết bị định tuyến là thiết bị mạng lớp 3 của mô hình OSI (Network Layer) với cấu tạo các phần cụ thể bao gồm: CPU, ROM, RAM, FLASH, NVRAM.

Cấu trúc của firmware rất đa dạng, phụ thuộc vào chức năng và thiết kế của từng nhà sản xuất. Các firmware có thể được chia thành các kiểu như sau:

- Integrated (apps + OS-as-a-lib);

- Partial updates (apps or libs or resources or support).

1.1.2. Tổng quan về mã độc Botnet trên thiết bị IoT dân dụng

Hiện nay, mã độc botnet trên các thiết bị IoT dân dụng có các loại sau đây:

- **Linux.Hydra**: Là mã độc đầu tiên lây nhiễm trên các thiết bị IoT (gọi tắt là mã độc IoT). Linux.Hydra xuất hiện vào năm 2008.

- **Psyb0t**: Tương tự như mã độc Linux.Hydra, mã độc Psyb0t được phát hiện lây nhiễm trên các thiết bị định tuyến, modem DSL có vi xử lý MIPS litteendian chạy firmware Mipsel Linux vào năm 2009.

- **Chuck Norris**: Ngay khi mã độc botnet Psyb0t được tạo ra, một mẫu mã độc mới đã được phát triển và trở thành đối thủ cạnh tranh trong năm 2010, được gọi là mã độc Chuck Norris.

- **Tsunami/Kaiten**: Tsunami còn có thể thực hiện tấn công bằng một số kỹ thuật phức tạp như HTTP Layer 7 Flood, TCP XMASS.

- **Aidra/LightAidra/Zendran**: Xuất hiện trong khoảng 2012, đây là 3 loại mã độc có nhiều phần mã nguồn tương tự nhau.

- **Spike/Dofloo/MrBlack/Wrkatk/Sotdas/AES.DdoS**: năm 2014 một dòng mã độc mới đã xuất hiện với nhiều loại mã độc như Spike, Dofloo, nhưng rất khó có thể phân biệt giữa các mã độc đó.

- **Bashlite/Lizkebab/Torlus/Gafgyt**: Xuất hiện vào năm 2014, có nhiều đặc điểm tương tự như dòng mã độc Spike.

- **Elknot/BillGates Botnet**: được phát hiện vào năm 2015, đây là mã độc được sử dụng khá phổ biến tại Trung quốc để thực hiện tấn công từ chối dịch vụ phân tán (DRDOS).

- **XOR.DdoS**: Trong năm 2015, một làn sóng mã độc khai thác lỗ hổng Shellshock có tên là XOR.DdoS đã âm thầm lây nhiễm nhiều thiết bị IoT.

- **Remaiten/KTN-RM**: Xuất hiện trong năm 2015 và được biết đến khá rộng rãi như mã độc Mirai. Remaiten kết hợp các đặc điểm chính của hai loại mã độc là Tsunami và BASHLITE.

- **NewAidra/IRCTelnet**: được biết đến với tên gọi là Linux.IRCTelnet, mã độc này được kết hợp dựa trên mã nguồn gốc Aidra, giao thức của Kaiten IRC based, mã dò quét/mã lây nhiễm của BASHLITE và bộ từ điển tấn công của Mirai.

- **Darlloz**: Hãng Symantec đã phát hiện ra một sâu mã độc có tên gọi là Darlloz, năm 2013, mã độc này khai thác lỗ hổng PHP có mã CVE-20121823.

- **Mirai**: Là mã độc khá nổi bật trong những năm qua, được sử dụng để thực hiện các vụ tấn công từ chối dịch vụ phân tán có tính quy mô rất lớn.

1.2. Tổng quan các phương pháp phát hiện mã độc

1.2.1. Phân tích tĩnh

Phương pháp phân tích tĩnh mã độc dựa trên những đặc trưng của các tập tin mà không cần thực thi chúng để phát hiện mã độc. Ưu điểm: Các phương pháp tĩnh có ưu điểm lớn nhất là có thể phân tích một cách chi tiết các tập tin và đưa ra được cái nhìn tổng quát về tất cả các khả năng kích hoạt của chúng. Hạn chế: Hạn chế lớn nhất của phân tích tĩnh là khi mã độc sử dụng các kỹ thuật gây rối phức tạp (obfuscations) như sắp xếp lại câu lệnh, chèn mã lệnh vô nghĩa, khi đó

rất khó có thể thu thập được thông tin ngữ nghĩa chính xác cho việc phân tích.

1.2.2. Phân tích động

Kỹ thuật phân tích động thông qua việc thực thi các mã chương trình và quan sát các hành vi của nó để phát hiện có hay không mã độc. Ưu điểm: Phương pháp phân tích động có hiệu quả và độ chính xác, cho phép xác định nhanh chóng và tổng quát về mã độc được phân tích thông qua các hành vị bộc lộ của chúng. Hạn chế: Mặc dù có những ưu điểm như đã nêu, phân tích động chỉ có thể giám sát đơn luồng thực thi. Ngoài ra phân tích động cũng có thể gây nguy cơ mất an toàn cho mạng và hệ thống.

1.2.3. Phân tích lai

Cả phân tích tĩnh và phân tích động đều có những hạn chế nhất định. Vì thế phân tích tĩnh và phân tích động đều có thể bổ trợ lẫn nhau. Vì vậy, các nghiên cứu phương pháp lai hiện nay tiếp cận theo hướng phân tích tĩnh trước sau đó tiến hành dò quét động để bổ sung các thông tin nhằm xác định mã độc hoặc sử dụng phân tích động để thực hiện bóc tách mã độc rồi sử dụng phân tích tĩnh.

1.3. Tổng quan về học máy

1.3.1. Các khái niệm cơ bản

1.3.1.1. Khái niệm học máy

Khái niệm chính thức về học máy và đã được trích dẫn rộng rãi trong lĩnh vực học máy: “Một chương trình máy tính được gọi là học từ kinh nghiệm E để hoàn thành nhiệm vụ T, với hiệu quả được đo bằng phép đánh giá P, nếu hiệu quả của nó khi thực hiện nhiệm vụ T, khi được đánh giá bởi P, cải thiện theo kinh nghiệm E”.

1.3.1.2. Phân loại các thuật toán học máy

Việc huấn luyện các mô hình học máy có thể xem là việc cho chúng trải nghiệm trên các tập dữ liệu – tập huấn luyện. Các dữ liệu khác nhau thì sẽ cho các mô hình các trải nghiệm khác nhau. Chất lượng của các tập dữ liệu này cũng ảnh hưởng tới hiệu năng của mô hình. Dựa trên tín chất của các tập dữ liệu, các thuật toán học máy có thể được phân thành hai loại: học không giám sát và học có giám sát.

1.3.1.3. Hàm mất mát và tham số mô hình

Quan hệ giữa một phép đánh giá và các tham số mô hình thường được mô tả thông qua một hàm số được gọi là hàm mất mát (loss function). Hàm mất mát này thường có giá trị nhỏ khi phép đánh giá cho kết quả tốt và ngược lại. Việc đi tìm các tham số mô hình sao cho phép đánh giá trả về kết quả tốt tương đương với việc tối thiểu hàm mất mát. Như vậy, việc xây dựng một mô hình học máy chính là việc đi giải một bài toán tối ưu. Quá trình đó có thể được coi là quá trình học của máy.

1.3.1.4. Mô hình các thuật toán học máy

Trong mỗi bài toán của học máy sẽ có hai bước (phase) lớn là bước huấn luyện và bước kiểm thử. Dữ liệu cũng vậy sẽ được chia thành dữ liệu huấn luyện và dữ liệu kiểm thử. Bước huấn luyện chỉ sử dụng dữ liệu huấn luyện, bước kiểm thử chỉ sử dụng kiểm thử.

1.3.2. Support vector machines

Support vector machines (SVM) là một trong những thuật toán phân lớp phổ biến và hiệu quả. SVM hoạt động dựa trên 2 nguyên tắc:

- SVM tìm cách phân lớp bằng một siêu phẳng sao cho khoảng cách từ siêu phẳng tới các lớp là lớn nhất. Nguyên tắc này được gọi là nguyên tắc lề cực đại (max margin);

- Trong trường hợp, không thể phân lớp bằng một siêu phẳng thì SVM sẽ ánh xạ không gian ban đầu sang một không gian khác (thường là có số chiều nhiều hơn), sau đó tìm lề cực đại trong không gian mới này.

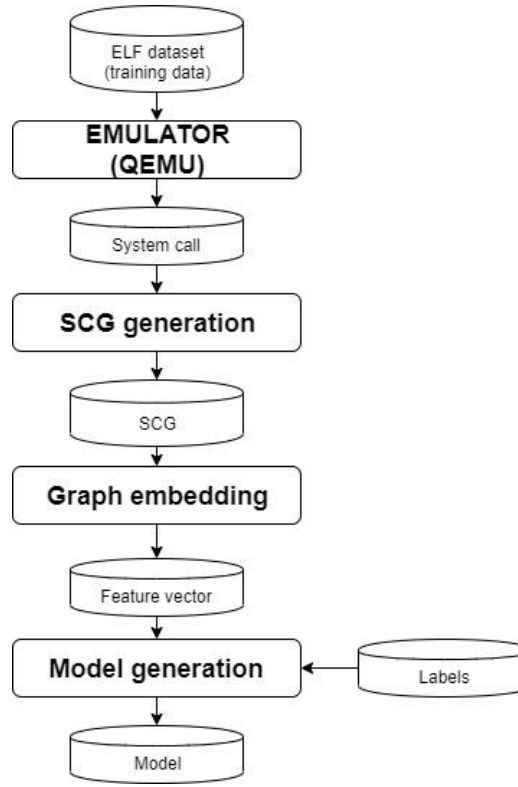
1.4. Kết luận chương

Trong nội dung chương 1 tác giả đã tập trung vào việc trình bày: kiến thức tổng quan về IoT; mã độc IoT botnet; tổng quan về các phương pháp phát hiện mã độc; tổng quan về học máy và thuật toán học SVM. Các kiến thức này sẽ là kiến thức nền tảng để tác giả vận dụng đề xuất nên mô hình phát hiện mã độc IoT botnet sẽ được trình bày trong chương 2.

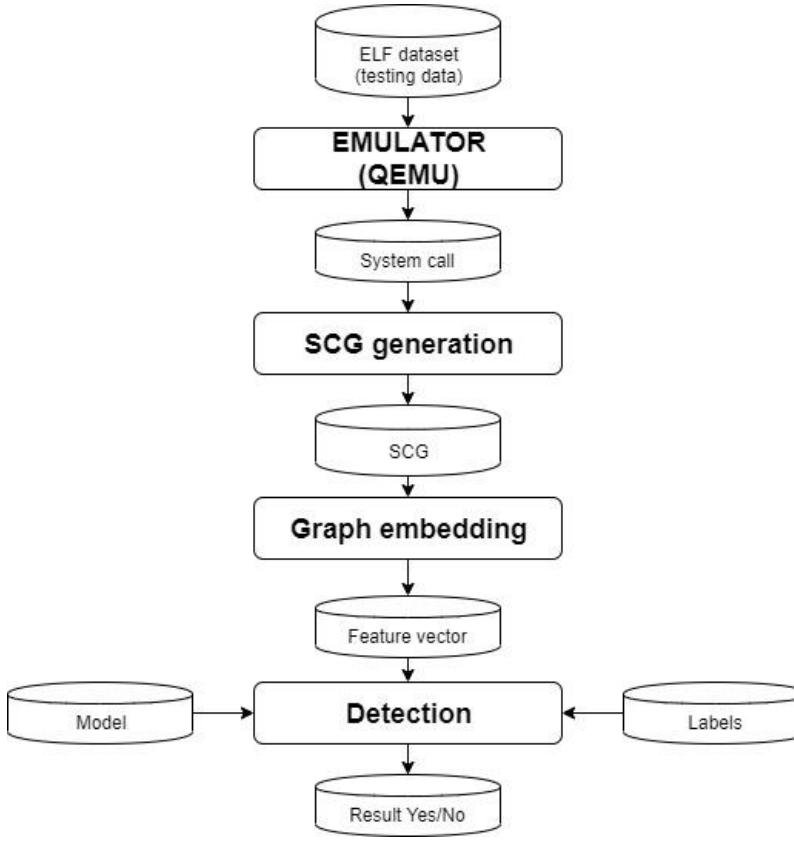
CHƯƠNG 2: XÂY DỰNG MÔ HÌNH PHÁT HIỆN MÃ ĐỘC IOT BOTNET

2.1. Mô hình tổng quan

Mô hình tổng quan của bài toán áp dụng học máy trong phát hiện mã độc IoT botnet sẽ có hai pha: pha huấn luyện (trainning) và pha kiểm thử (testing).



Hình 2.1. Pha huấn luyện trong hô hình phát hiện botnet trong các thiết bị IOT



Hình 2.1. Pha kiểm thử trong mô hình phát hiện mã độc IoT botnet

Hai pha huấn luyện và kiểm thử sẽ có các bước tiền xử lý dữ liệu giống nhau đó là:

- Thu thập dữ liệu bằng Emulator (QEMU);
- Xây dựng đồ thị SCG (SCG generation);
- Xây dựng đồ thị nhúng (graph embedding).

Sau khi có vector đặc trưng từ bước xây dựng đồ thị nhúng, trong pha huấn luyện sẽ sử dụng thêm dữ liệu nhãn đánh dấu mã độc/lành tính của tập dữ liệu huấn luyện để đưa vào kỹ thuật học máy để sinh model. Trong khi đó pha huấn luyện sẽ sử dụng model được xây dựng trong pha huấn luyện, cùng với dữ liệu kiểm thử để đưa ra kết quả phát hiện. Trong phần tiếp theo, tác giả sẽ mô tả chi tiết chức năng cũng như cấu trúc của từng thành phần cụ thể.

2.2. Thu thập dữ liệu

Trong giai đoạn này, mô hình sẽ sử dụng QEMU để giả lập kiến trúc chíp ARM, MIPS kiến trúc được sử dụng phổ biến trên các thiết bị IOT dân dụng, sau đó chạy bản ảnh QEMU Debian VM dành riêng cho kiến trúc chíp ARM, MIPS. Công việc tiếp theo là sử dụng strace tool để có thể trích xuất ra được các lời gọi hệ thống của các tệp tin mã độc. Đầu vào của giai đoạn này là các tệp tin mã độc và kết quả đầu ra là tập các tệp tin log lưu trữ chi tiết các lời gọi hệ thống của tệp tin mã độc.

Nhiều giải pháp Sandbox tương đối hoàn chỉnh cho phân tích mã độc đã được xây dựng như Cuckoo sandbox. Tuy nhiên, đối với các kiến trúc chíp ARM, MIPS Cuckoo sandbox cho kết quả nhật ký các lời gọi hệ thống không được đầy đủ và chi tiết. Vì vậy, trong luận văn này tác giả sử dụng bộ giả lập QEMU để mô phỏng kiến trúc chíp ARM, MIPS sử dụng mã nguồn mở Strace tool để thu thập log và điều quan trọng là sử dụng bản ảnh QEMU Debian VM dành cho kiến trúc ARM, MIPS do Aurel đăng tải lên.

QEMU là một phần mềm mã nguồn mở được viết bởi Fabrice Bellard dùng để giả lập và ảo hóa phần cứng. QEMU mô phỏng bộ vi xử lý của máy thông qua dịch nhị phân động và cung cấp một bộ các mô hình phân cứng và các thiết bị khác nhau điều đó cho phép QEMU có thể chạy với nhiều hệ điều hành khác nhau. Khi được sử dụng như một trình ảo hóa, QEMU có thể đạt được hiệu năng gần như máy thật bằng cách thực thi mã khách trực tiếp trên CPU của máy chủ. Bên cạnh đó, QEMU có thể lưu trữ và khôi phục lại trạng thái làm việc của hệ điều hành với tất cả các chương trình đang chạy.

Strace là một tiện ích được sử dụng để chẩn đoán, gỡ lỗi trong hệ điều hành linux. Strace được viết bởi Paul Kranenburg vào năm 1991 và dành cho SunOS. Nó được sử dụng để giám sát và giả mạo các tương tác giữa các quy trình và nhân Linux, bao gồm các lời gọi hệ thống, phân phối tín hiệu và thay đổi trạng thái của quy trình.

2.3. Xây dựng đồ thị SCG

Thuật toán xây dựng đồ thị lời gọi hệ thống có thể được biểu diễn như sau:

```
1: function scg_generation(S[], id)
   Input:
    - S[]: a list of system-calls log for each
process
    - id: PID of the process
   Output: system call graph(SCG)
2: s_log = S[id]
3: V=[], E=[]
4: SCG = (V,E)
5: V = V ∪ {s_log[0]}
```

```
     6: for(i=1;i<|s_log|;i++) do
     7:  if s_log[i]<>s_log[i-1] then
     8:       if s_log[i] is "fork" then
     9:           child_pid = Get PID of process
which is opened by "fork"
     10:          SCG = SCG U scg_generation(S,
child_pid)
     11:      endif
     12:      if s_log[i] not in V then
     13:          V = V U {s_log[i]}
     14:      endif
     15:      E = E U {edge(s_log[i-1] connect to
s_log[i])}
     16: endif
     17: endfor
     18: return SCG
```

2.4. Đồ thị nhúng

Đồ thị nhúng là sự chuyển đổ các thuộc tính của đồ thị thành một vector hoặc một tập vector. Việc chuyển đổi này vẫn lưu giữ được các thông tin của đồ thị như: cấu trúc liên kết của đồ thị, mối liên quan giữa các đỉnh, sơ đồ con và các thông tin liên quan đến đồ thị. Có 2 phương pháp chính trong đồ thị nhúng đó là: vertex embeddings và graph embeddings.

- Vertex embeddings: phương pháp sẽ chuyển đổi mỗi đỉnh trong đồ thị thành một vector. Phương pháp này thường được áp dụng trong các trường muốn trực quan hóa hoặc dự đoán cấp độ của đỉnh. Ví dụ như: trực quán hóa các đỉnh trong mặt

phẳng 2D; dự đoán liên kết mới dựa trên sự tương đồng về đỉnh.

- Graph embeddings: phương pháp sẽ chuyển đổi đồ thị về một vector duy nhất. Phương pháp này thường được áp dụng vào việc muốn dự đoán về mức độ của đồ thị hoặc cần so sánh và trực quan hóa toán bộ đồ thị. Ví dụ như so sánh các cấu trúc hóa học.

3.5. Thiết lập mô hình học máy

Bài toán phân lớp truyền thống không thể hoạt động tốt trong trường hợp lực lượng giữa 2 lớp chênh lệch quá lớn hoặc thậm chí chỉ có dữ liệu của một tập dữ liệu. Ví dụ: bài toán phân loại trạng thái hoạt động bình thường của nhà máy hạt nhân, thì trong kịch bản này, có rất ít các trạng thái hệ thống bị lỗi mà chỉ có số liệu thống kê hoạt động bình thường của nhà máy. Trong trường hợp này, người ta đề xuất mô hình phân loại một lớp (One class classification – OCC).

Ý tưởng phân loại một lớp dựa trên SVM (OSVM) là việc xác định một siêu cầu nhỏ nhất (có tâm c và bán kính r) có thể bao lấy tất cả các điểm dữ liệu. OSVM có thể được xác định như sau:

$$\min_{r,c} r^2 \ sao\ cho\ ||\Phi(x_i) - c|| \leq r^2\ \forall i = 1,2,\dots,n$$

Tuy nhiên công thức trên sẽ rất hạn chế và nhạy cảm với nhiễu. Do đó một công thức linh hoạt hơn được xây dưng để có thể giảm nhiễu và có thể chấp nhận được như sau:

$$\min_{r,c,\zeta} r^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\zeta_i$$

Sao cho

$$||\Phi(x_i) - c||^2 \leq r^2 + \zeta_i \quad \forall i = 1, 2, \ldots, n$$

Từ điều kiện tối ưu của Karush-Kuhn-Tucker (KKT), chúng ta có được

$$c = \sum_{i=1}^{n}\alpha_i \Phi(x_i),$$

khi đó $\alpha_i' s$là giải pháp cho vấn đề tối ưu hóa:

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i \kappa(x_i, x_i) - \sum_{i,j=1}^{n}\alpha_i \alpha_j \kappa(x_i, x_j)$$

Sao cho

$$\sum_{i=1}^{n}\alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq \frac{1}{\nu n} \text{for all } i = 1, 2, \ldots, n.$$

Một số phương pháp đã được đề xuất để giải quyết vấn đề phân loại một lớp. Các cách tiếp cận có thể được phân loại thành ba loại chính: ước tính mật độ, phương pháp biên và phương pháp tái thiết.

3.6. Kết luận chương

Nội dung chương 2 căn cứ vào nghiên cứu lý thuyết tại chương 1 đưa ra mô hình đề xuất cho phương pháp phát hiện mã độc IoT botnet. Mô hình phát hiện bao gồm hai pha: Huấn luyện và kiểm thử. Các biến tiền xử lý của hai pha giống nhau bao gồm các bước:

- Thu thập dữ liệu bằng Emulator (QEMU);
- Xây dựng đồ thị SCG (SCG generation);
- Xây dựng đồ thị nhúng (graph embedding).

Pha huấn luyện sẽ sử dụng thêm dữ liệu nhãn đánh dấu mã độc/lành tính của tập dữ liệu huấn luyện để đưa vào kỹ thuật học máy để sinh model. Trong khi đó pha huấn luyện sẽ sử dụng model được xây dựng trong pha huấn luyện, cùng với dữ liệu kiểm thử để đưa ra kết quả phát hiện. Trong chương 3, tác giả sẽ mô tả các bước cài đặt môi trường, cũng như các công cụ sử dụng để xây dựng và kiểm thử mô hình học máy này.

CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ

Nội dung chương 3 sẽ tiến hành áp dụng mô hình phát hiện botnet trên các thiết bị IOT đã được đề xuất ở chương 2 vào tập dữ liệu mẫu. Sau đó kết quả sẽ được đưa ra nhật xét và đánh giá chất lượng của mô hình.

3.1. Thu thập và tiền xử lý dữ liệu

3.1.1. Dữ liệu mẫu

Luận văn đã thu thập được 1,326 tệp tin dữ liệu mẫu, trong đó có 1,248 tệp tin có chứa mã độc và 78 tệp tin lành tính không chưa mã độc. Tất cả các dữ liệu mẫu này đều là ứng dụng chạy trên hệ điều hành linux. Các tệp tin mã độc được thu thập từ nguồn VirusShare và IoTPOT.

3.1.3. Thu thập dữ liệu

Để có thể thu thập được dữ liệu một cách tự động, tác giả đã sao chép các tệp tin mẫu mã độc vào máy khách sau đó chạy đoạn mã GetLog.sh để có thể lấy được log lời gọi hàm hệ thống. Với mỗi log của tệp tin mẫu mã độc sẽ được lưu trữ trong một tệp tin text.

3.1.4. Xây dựng đồ thị SCG và đồ thị nhúng

Bước tiếp theo tác giả sẽ sử dụng mã nguồn mở graph2vec để có thể chuyển đổi đồ thị lời gọi hàm hệ thống thành đồ thị nhúng. Đầu vào của bước này là các tệp tin đồ thị *.gexf và đầu sẽ là một ma trận

2 chiều với mỗi hàng là một vector của một đồ thị. Ma trận vector này sẽ được lưu trữ trong một tệp tin *.txt.

3.2. Thử nghiệm

Trong bước này tác giả sẽ cài đặt thuật toán học máy one-class SVM bằng thư viện scikit-learn. Chia dữ liệu thành 2 phần dữ liệu huấn luyện và dữ liệu kiểm thử:

- Dữ liệu huấn luyện: 936 mẫu (chiếm 75% số lượng mã độc);
- Dữ liệu kiểm thử: 312 mẫu (chiếm 25% số lượng mã độc) và 78 mẫu lành tính.

3.3. Nhận xét đánh giá

Kết quả chạy được với bộ dữ liệu mẫu thu được như sau:

Confusion matrix:

Bảng 3.1. Kết quả Confusion matrix

	Được phân loại bởi hệ thống	
	Mã độc	Lành tính
Mã độc	300	12
Lành tính	8	70

Dựa vào Confusion matrix ta có thể thấy:

- Với 312 mẫu mã độc để kiểm thử thì mô hình phân lớp đúng 300 mẫu đúng vào lớp mã độc và 12 mẫu bị phân lớp sai vào lớp lành tính;

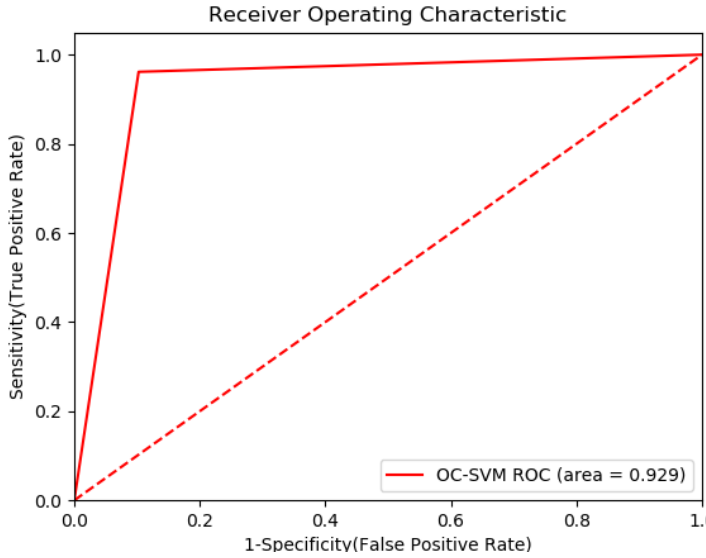
- Với 78 mẫu lành tính mô hình phân lớp đúng 70 mẫu vào lớp lành tình và 8 mẫu bị phân lớp sai vào lớp mã độc.

	Precision	Recall	F1- measure	support
Lành tính	0.85	0.90	0.88	78
Mã độc	0.97	0.96	0.97	312

Accuracy: 0.949

ROC AUC: 0.929

Falase Positive rate: 0.103



Hình 3.1. Đường ROC của mô hình đề xuất trong kiểm thử

Dựa vào kết quả kiểm thử mô hình đề xuất, có thể đưa ra một số nhận xét:

- Accuracy của mô hình đề xuất là: 94.9%, đối với một bài toán phân loại một lớp thì acc=94.9% là kết quả khá tốt;
- Đường ROC với giá trị AUC=0.929 có thể thấy hiệu quả của thuật toán với độ chính xác cao khi mà giá trị AUC tiến rất gần với giá trị lý tưởng là 1;

3.4. Kết luận chương

Chương 3 đã trình bày về quá trình thử nghiệm mô hình phát hiện mã độc IoT botnet, bao gồm các bước thực hiện, kết quả thực nghiệm, đánh giá và nhận xét về mô hình. Với 1,326 mẫu dữ liệu được chia thành 75% dành cho dữ liệu huấn luyện và 25% dành cho dữ liệu kiểm thử đưa vào mô hình huấn luyện và kiểm thử. Kết quả đạt được khá tốt đó là acc = 94.9%.

KẾT LUẬN

1. Những đóng góp của luận văn

Luận văn đã nghiên cứu về phương pháp phát hiện mã độc bonet trên các thiết bị IoT dân dụng vào thuật toán học máy one-lass SVM và đã đạt được một số kết quả như sau:

- Giới thiệu tổng quan về thiết bị IoT dân dụng;
- Giới thiệu tổng quan về mã độc IoT botnet và các phương pháp phát hiện mã độc;
- Trình bày cơ sở lý thuyết và đưa ra mô hình phát hiện mã độc IoT botnet bằng kỹ thuật học máy one-class SVM;
- Tiến hành thực nghiệm, đánh giá kết quả thu được.

Tổng kết lại, quá trình thử nghiệm mô hình được thực hiện như sau: Sử dụng QEMU để giả lập kiến trúc chíp MIPS, ARM và sử dụng công cụ Strace tool để thu thập nhật ký lời gọi hàm hệ thống của tệp dữ liệu mẫu, sau đó xây dựng đồ thị lời gọi hàm hệ thống, bước tiếp là sử dụng thư viện graph2vec để trích chọn đặc trưng từ đồ thị lời gọi hàm hệ thống và cuối cùng là đẩy các vector đặc trưng vào để huấn luyện thuật toán học máy one-class SVM dựa vào thư viện mã nguồn mở Sklearn.

2. Hướng phát triển tiếp theo

Hiện tại, luận văn chỉ mới sử dụng thuật toán học máy one-class SVM để áp dụng học máy và chỉ mới các thiết bị IoT dân dụng. Hướng tiếp theo sẽ là nghiên cứu các thuật toán học sâu để có thể áp

dụng vào bài toán phát hiện mã độc botnet trên các thiết bị IoT. Bên cạnh đó có thể mở rộng phạm vi nghiên cứu thêm các loại thiết bị IoT khác từ đó có thể xây dựng được một hệ thống sandbox đa nền tảng cho các thiết bị IoT. Ngoài ra luận văn cũng sẽ thu thập thêm tập dữ liệu mẫu để có thể tăng độ chính xác cho kết quả phát hiện, bởi tập dữ liệu hiện tại chỉ mới có một số lượng nhỏ các tệp tin mã độc botnet.