ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Nghi Phú

NGHIÊN CỨU PHÁT TRIỂN PHƯƠNG PHÁP PHÁT HIỆN CÁC LỖI BẢO MẬT AN NINH CHO PHẦN MỀM NHÚNG VÀ CÁCH KHẮC PHỤC

TÓM TẮT LUẬN ÁN TIẾN SỸ CÔNG NGHỆ THÔNG TIN

Hà Nội - 2019

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Nghi Phú

NGHIÊN CỨU PHÁT TRIỂN PHƯƠNG PHÁP PHÁT HIỆN CÁC LỖI BẢO MẬT AN NINH CHO PHẦN MỀM NHÚNG VÀ CÁCH KHẮC PHỤC

Chuyên ngành: Kỹ thuật Phần mềm
Mã số: 9480103.01

TÓM TẮT LUẬN ÁN TIẾN SỸ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:
1. PGS.TS. Nguyễn Ngọc Bình
2. TS. Nguyễn Đại Thọ

Hà Nội - 2019

Mục lục

Chương 1

MỞ ĐẦU

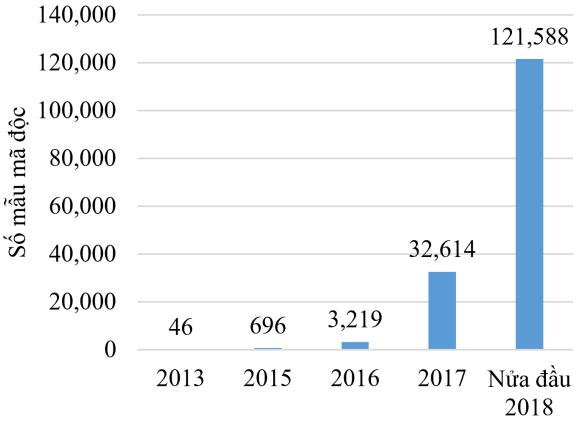
1.1 Bối cảnh

Internet vạn vật (Internet of Things - IoT) đang ngày càng phổ biến, tiếp tục phát triển mạnh và dần đóng vai trò quan trọng trong tương lai. Đây là một trong những thành phần cốt lõi tạo nên cuộc cách mạng công nghiệp lần thứ tư, xây dựng thành phố thông minh và trực tiếp mang lại nhiều tiện ích trong nhiều lĩnh vực như: y tế, nông nghiệp, giao thông thông minh, quốc phòng v.v. Theo thống kê và dự báo của Cisco, năm 2017 đã có khoảng 7 tỉ thiết bị IoT, đến năm 2020 ước tính sẽ có 50 tỷ thiết bị IoT sẽ được kết nối vào mạng Internet. Các thiết bị này sẽ có mặt ở mọi nơi, từ các địa điểm công cộng đến những điểm riêng tư, phổ biến là các thiết bị như CameraIP, VoIp, IpTV, thiết bị định tuyến v.v. Thiết bị IoT ở trong luận án được hiểu là các thiết bị nhúng có kết nối mạng Internet.

1.2 Đặt vấn đề

Cùng với lỗ hổng bảo mật, mã độc là mối đe doạ chính ảnh hưởng an ninh, an toàn của các thiết bị IoT. Các nguy cơ có thể ảnh hưởng tới thiết bị IoT bao gồm: chính sách quản lý (31%), lây nhiễm mã độc (26%), tấn công từ chối dịch vụ (13%) và tấn công phá hoại kết nối của các thiết bị IoT (12%). Mã độc trên IoT đang tăng nhanh về số lượng, sức phá hoại và kỹ thuật ngày càng phức tạp, chi tiết các năm trình bày trong Hình 1.1.

Thiết bị IoT có nhiều đặc thù khác với máy tính cá nhân như: hạn chế tài nguyên, thường xuyên kết nối với Internet, ít được nâng cấp v.v. Đặc biệt, các thiết bị sử dụng kiến trúc bộ vi xử lý cũng như hệ điều hành nhúng hoàn toàn khác. Do đó, không thể áp dụng các phương pháp về mã độc nói chung sang mã độc nhúng, mà cần có các phương pháp đặc thù, tối ưu và các công cụ, môi trường mới để hỗ trợ phân tích, phát hiện mã độc nhúng. Các nghiên cứu về mã độc trên IoT đang tập trung theo hướng xây dựng các công cụ và môi trường hỗ trợ, phương pháp tĩnh và động trong phát hiện mã độc. Các nghiên cứu về phát hiện mã độc trên IoT trước đây tập trung chủ yếu vào điện thoại đi động với hệ điều hành Android và kiến trúc vi xử lý ARM. Một lượng lớn các thiết bị IoT sử dụng hệ điều hành Linux nhúng với các kiến trúc rất phổ biến như MIPS chưa được đề cập nhiều. Phương pháp tĩnh bước đầu có nhiều lợi thế, do mã độc trên IoT ít sử dụng các biện pháp bảo vệ như mã hoá, nén, làm rối v.v. Nhưng các phương pháp



Hình 1.1: Số lượng mã độc IoT xuất hiện trong các năm.

phân tích tĩnh áp dụng hiệu quả cho máy tính cá nhân khi áp dụng sang IoT không hiệu quả do tốc độ xử lý và môi trường đa kiến trúc. Do đó, việc nghiên cứu xây dựng các cơ sở dữ liệu đầy đủ, các công cụ, môi trường và các thuật toán phát hiện hiệu quả mã độc trên các thiết bị IoT có vai trò quan trọng, có ý nghĩa lớn về lý thuyết và thực tiễn.

1.3 Mục tiêu nghiên cứu và các đóng góp chính của luận án

Mục tiêu của luận án là phát triển các công cụ, đề xuất các thuật toán mới có độ chính xác cao và tốc độ xử lý nhanh để phát hiện mã độc trong các thiết bị IoT sử dụng hệ điều hành Linux nhúng và các vi xử lý nhúng, đặc biệt là MIPS. Các đóng góp chính của luận án gồm:

Thứ nhất, xây dựng cơ sở dữ liệu về phần sụn của các thiết bị IoT và mã độc trên IoT, gọi là tập dữ liệu Firmware IoT (F-IoT). Đây là cơ sở dữ liệu có số lượng lớn và đầy đủ về các phần sụn của các thiết bị IoT, mã độc và mã sạch trên thiết bị IoT đến thời điểm hiện tại. Tập mã độc của F-IoT được xây dựng trên cơ sở kết hợp và tập dữ liệu đã có và tạo mới, sau đó được tiến hành phân tích, lọc nhiễu, gán nhãn theo nhiều tiêu chí phục vụ phần thực nghiệm. F-IoT là tập dữ liệu lớn, được sử dụng trong thực nghiệm của luận án và có ý nghĩa lớn cho các nghiên cứu của cộng đồng sau này về IoT. Đóng góp này của chúng tôi đã được công bố vắn tắt trong các báo cáo hội thảo [PP4, PP7, PP6, PP8] và sau đó được đăng thành phiên bản đầy đủ trong bài tạp chí [PP2,PP3].

Thứ hai, đề xuất quy trình phát hiện mã độc trên thiết bị IoT bằng phương pháp phân tích động F-Sandbox như một loại sandbox mới, chuyên dụng cho các thiết bị IoT dựa trên hệ điều hành Linux nhúng. F-Sandbox được phát triển dựa trên QEMU, kế thừa các kỹ thuật của Firmadyne, có khả năng mô phỏng đầy đủ các thành phần vật lý của bộ định tuyến, mạng Internet và trích xuất các lời gọi hệ thống dựa trên nhân Linux 2.6 được sửa đổi (instrumented kernel). Đóng góp này của chúng tôi đã được công bố vắn tắt trong báo cáo hội thảo [PP4] và sau đó tiếp tục phát triển theo hai hướng được đăng thành phiên bản đầy đủ trong các bài tạp chí [PP1, PP2].

Thứ ba, đề xuất thuật toán trích xuất đặc trưng của các tệp tin thực thi (chương trình chứa mã độc và chương trình sạch) dựa trên luồng điều khiển (Control flow-based feature) có hiệu quả và hiệu suất cao cho việc phát hiện mã độc trên thiết bị IoT, gọi là CFD. CFD là giải thuật quy hoạt động, có tốc độ xử lý nhanh, sử dụng ít bộ nhớ, cho phép xử lý các tệp tin lớn, có nhiều khối cơ bản (basic block). Về mặt lý luận và thực nghiệm đã chứng minh, CFD phù hợp cho việc trích chọn đặc trưng của các chương trình trên các thiết bị IoT. Đóng góp này của chúng tôi đã được công bố vắn tắt trong báo cáo hội thảo [PP5].

Cuối cùng, phát triển phương pháp trích chọn đặc trưng CFDVex phát hiện mã độc nhúng đa nền tảng vi xử lý, tức học các mã độc trên kiến trúc cũ để phát hiện các phiên bản mã độc trên kiến trúc vi xử lý mới, đây là một trong những xu thế xuất hiện của mã độc trên IoT. Phương pháp CFDVex sẽ dịch ngược chương trình về mã trung gian Vex, sau đó áp dụng thuật toán CFD để trích xuất thông tin đặc trưng dựa trên luồng điều khiển của đồ thị với các đỉnh là chuỗi các đại diện của lệnh Vex. Thực nghiệm cho thấy, CFDVex có khả năng phát hiện mã độc đa kiến trúc với độ chính xác cao. Đóng góp này của chúng tôi đã được công bố vắn tắt trong báo cáo hội thảo [PP6].

Các nghiên cứu của luận án nhằm xây dựng một phương pháp hoàn chỉnh gồm các phương pháp động và phương pháp tĩnh để phát hiện mã độc trong các thiết bị IoT sử dụng hệ điều hành Linux nhúng nói chung. Các thực nghiệm cho kết quả tốt và mở ra nhiều hướng phát triển tiếp theo.

1.4 Cấu trúc luận án

Luận án *Nghiên cứu phương pháp phát hiện lỗi bảo mật an ninh cho các phần mềm nhúng và cách khắc phục* bao gồm năm chương, cụ thể nội dung các chương như sau:

Chương 1. Mở đầu trình bày về lý do chọn đề tài và nội dung nghiên cứu.

Chương 2. Phân tích mã độc trong các hệ thống nhúng trình bày về các kiến thức nền được sử dụng trong luận án. Đầu tiên chương này trình bày tổng quan về hệ thống nhúng, phần mềm nhúng, các loại mã độc trên thiết bị nhúng.

Chương 3. Một giải pháp phân tích động đối với các mã độc nhúng trình bày kiến trúc F-Sandbox và hai quy trình sử dụng F-Sandbox cho phát hiện, phân lớp mã độc trên thiết bị IoT bằng phương pháp động, sử dụng đặc trưng về lời gọi hệ thống.

Chương 4. Phương pháp đặc trưng dòng điều khiển cho phát hiện các mã độc nhúng trình bày hai phương pháp trích xuất đặc trưng dòng điều khiển mới áp dụng hiệu quả cho phát hiện mã độc trên thiết bị IoT là CFD và CFDVex.

Chương 5. phân tích về các đóng góp chính của luận án và thảo luận về các nghiên cứu trong tương lai từ các kết quả ban đầu đã đạt được.

Chương 2

PHÂN TÍCH MÃ ĐỘC TRONG CÁC HỆ THỐNG NHÚNG

Chương này trình bày các kiến thức cơ sở liên quan đến phân tích mã độc nói chung, các thiết bị nhúng và đi sâu vào phân tích mã độc nhúng. Đầu tiên, chúng tôi trình bày tổng quan về bài toán phân tích mã độc và hướng vào bài toán sử dụng học máy trong phân tích mã độc. Tiếp theo, chúng tôi trình bày các kiến thức cơ sở về các hệ thống nhúng gồm thiết bị nhúng, hệ điều hành nhúng, hệ điều hành nhúng cụ thể là Linux nhúng. Phần cuối chương trình bày phân tích mã độc nhúng với các nội dung khái niệm, phân loại, đặc điểm, các môi trường và công cụ phân tích đã có, các phương pháp phân tích mã độc nhúng trong thời gian gần đây.

2.1 Tổng quan về phân tích mã độc

2.1.1 Khái niệm mã độc

Theo Aycock, mã độc (malware viết tắt của Malicious software) là một chương trình hoặc đoạn mã được chèn một cách bí mật vào hệ thống với mục đích làm tổn hại đến tính tin cậy, tính bảo mật, tính toàn vẹn hoặc tính sẵn sàng của hệ thống.

Đa số các mã độc trong quá khứ được sinh ra tập trung cho máy tính cá nhân sử dụng hệ điều hành Windows của Microsoft. Tuy nhiên, thời gian gần đây, với sự phát triển mạnh mẽ của IoT, mã độc IoT tăng lên nhanh chóng. Ngoài ra các thiết bị di động sử dụng hệ điều hành Android, một lượng lớn các thiết IoT sử dụng hệ điều hành Linux nhúng (Embedded Linux), đó là cơ sở phát triển các mã độc trên Linux nhúng, gọi tắt là mã độc nhúng.

2.1.2 Bài toán phân tích tự động mã độc

Bài toán phân tích mã độc hướng tới ba mục đích: phát hiện mã độc (malware detection), phân tích độ tương tự của mã độc (malware similarity analysis) và phát hiện loại mã độc (malware category detection). Phát hiện mã độc có mục đích xác định mẫu đầu vào có phải là mã độc (malicous) hay không. Phân tích độ tương tự của mã độc là chỉ ra những điểm giống nhau giữa các mã độc, như tìm ra những điểm tương tự giữa các mã độc mới xuất hiện với các mã độc đã có từ trước. Phát hiện loại mã độc là việc

dựa trên các hành vi nổi bật và mục đích để phân mã độc thành các loại khác nhau như gián điệp (spy), điều khiển từ xa (remote) v.v.

Phát hiện dựa trên hành vi (Behavior-based detection) sẽ xác định hành vi bất thường dựa trên các hành vi bình thường mà chúng thu thập được dựa trên hai pha huấn luyện và nhận dạng. Phát hiện dựa trên chữ ký (Signature-based detection) là phương pháp xây dựng mô hình của hành vi độc hại và sử dụng mô hình này để phát hiện mã độc.

Trích chọn đặc trưng (feature extraction) là tiến trình trích xuất thông tin từ mẫu để thu các đặc trưng của mẫu phục vụ mục đích phân tích mẫu. Quá trình này có thể được thực hiện bằng phương pháp tích pháp tĩnh, phương pháp phân tích động hay kết hợp cả hai phương pháp. Phân tích tĩnh là phương pháp phân tích mẫu dựa trên những đặc trưng của các mẫu mà không cần thực thi chúng. Phương pháp phân tích tĩnh cũng còn tồn tại nhiều vấn đề như tệp tin có dung lượng lớn, phức tạp thì phương pháp phân tích động đóng một vai trò quan trọng. Phương pháp phân tích động là phương pháp phân tích các hoạt động của mẫu khi mẫu được thực thi thực sự (trong môi trường vật lý hoặc mô phỏng).

2.1.3 Các đặc trưng của mã độc và phương pháp trích rút

Giả mã thực thi (Operation code - Opcode) là lệnh cơ bản được thực hiện trên bộ vi xử lý. Nó là mã máy trừu tượng mức thấp hay còn gọi là ngôn ngữ máy để thực thi các hoạt động của một loại vi xử lý nhất định, mỗi loại vi xử lý khác nhau lại sử dụng một tệp lệnh khác nhau. Chuỗi opcode được trích xuất từ việc dịch ngược (disassembly) từ tệp tin nhị phân có thể thực thi (executable) mô tả các hành vi của một chương trình và có thể được trích xuất thông qua phân tích tĩnh. Kết quả thu được là tên lệnh như sau: POP, PUSH, MOVE, ADD, SUB v.v.

Định nghĩa 1 (*Đồ thị dòng điều khiển*): Đồ thị dòng điều khiển (CFG) của một tập tin thực thi là một đồ thị có hướng $G = (V, E, r, L)$, trong đó:

— V là tập các đỉnh, mỗi đỉnh là một khối cơ bản (basic block) trong tập tin thực thi sau khi được dịch ngược, gồm chuỗi các câu lệnh liên tiếp nhau mà ở đó dòng điều khiển đi vào tại điểm khởi đầu và đi ra tại điểm kết thúc mà không bị dừng đột ngột hoặc xử lý rẽ nhánh trước khi kết thúc.

— E là tập các cạnh, được tạo ra bởi các lệnh jumps/calls/rets lệnh thực thi giữa các khối cơ bản của chương trình thực thi sau khi được dịch ngược. Với mỗi cạnh có hướng (u,v), u được gọi là đỉnh đầu và v được gọi là đỉnh cuối của cạnh. Khi duyệt đồ thị, u được gọi là cha của v và v được gọi là con của u;

— r là đỉnh đầu hay nút gốc của đồ thị, có bậc vào là 0 (tức không có cạnh nào đi đến r), nó là khối cơ bản chứa điểm bắt đầu (Entry block) của chương trình;

— L là tập các lá, tức bậc ra của đỉnh là 0 (hay từ đỉnh này không có cạnh nối đến các đỉnh khác), nó là các khối cơ bản kết thúc chương trình (Exit block).

Lời gọi hệ thống (System call - Syscall) là lệnh mà chương trình yêu cầu các dịch vụ từ nhân hệ điều hành, bao gồm các dịch vụ liên quan đến phần cứng như truy cập đĩa hay cạc mạng, tạo hay xử lý các tiến trình. Khi chương trình thực thi sẽ gọi các lời gọi hệ thống, do đo chuỗi các lời gọi hệ thống được gọi khi thực thi chương trình là chuỗi hành vi thể hiện đặc trưng của chương trình đó khi hoạt động.

Để trích rút các đặc trưng từ một chuỗi byte, ký tự, lời gọi hệ thống v.v. người ta thường sử dụng phương pháp n-gram. Phương pháp n-gram sẽ đếm tần suất xuất hiện của n phần tử liên tiếp để làm đặc trưng, số lần xuất hiện này được lưu trong một véc tơ để làm đặc trưng cho chuỗi đó. Ví dụ tính 3-gram của chuỗi ($ABACABACACA$) được mô tả như trong Bảng 2.1.

Bảng 2.1: Tần suất 3-gram của chuỗi.

3-gram	ABA	BAC	ACA	CAB	CAC
Tần suất	2	2	3	1	1

2.1.4 Sử dụng học máy trong phát hiện/phân loại mã độc

Các phương pháp học máy trong phát hiện mã độc phổ biến là Máy véc tơ hỗ trợ (Support Vector Machine-SVM) với 29% và đạt được độ chính xác cao nhất, tiếp theo là Cây quyết định (Decision Tree - DT) chiếm 14%, Naive Bayes - NB chiếm 10% và Rừng ngẫu nhiên (Random Forest - RF) chiếm 5%. SVM là phương pháp xây dựng một siêu phẳng (hyperplane) để phân lớp (classify) tập dữ liệu thành 2 lớp riêng biệt sao cho độ rộng từ các điểm đến siêu phẳng là lớn nhất. SVM chỉ có khả năng phân hai lớp, do đó khi áp dụng SVM cho bài toán phân lớp có nhiều lớp cần một số kỹ thuật như One vs One (OvO) hoặc One vs Rest (OvR). NB là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. NB thuộc nhóm học có giám sát, thường được sử dụng trong các bài toán phân lớp văn bản, có thời gian huấn luyện và phân lớp nhanh. DT là một kiểu mô hình dự báo (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. RF là phương pháp xây dựng một tập hợp rất nhiều cây quyết định và sử dụng phương pháp bầu (voting) để đưa ra quyết định về biến mục tiêu (target) cần được dự báo. Phân loại một lớp là phương pháp cho phép huấn luyện mô hình phân lớp với dữ liệu huấn luyện chỉ có một nhãn.

Khi số chiều của véc tơ đặc trưng quá lớn sẽ ảnh hưởng đến tốc độ tính toán. Người ta thường áp dụng các phương pháp giảm chiều, tức tạo ra véc tơ đặc trưng K chiều thay thế cho véc tơ ban đầu có Q chiều (với $K < Q$) nhưng vẫn đảm bảo giữ nguyên thông tin như véc tơ Q chiều ban đầu hoặc có thay đổi nhưng vẫn đảm bảo độ chính xác cho phép. Các phương pháp không giám sát lọc thông tin dựa trên mối tương quan giữa các đặc trưng với nhau, không sử dụng các thông tin về nhãn. Phương pháp giảm chiều tiêu biểu với trường hợp không có nhãn là PCA. Ngược lại, các phương pháp giảm chiều dựa trên nhãn (có giám sát) xác định mối tương quan giữa các đặc trưng và nhãn

lớp dựa trên các độ đo khoảng cách, độ phụ thuộc thông tin. Phương pháp giảm chiều có sử dụng nhãn phổ biến, cho hiệu quả cao và được chúng tôi sử dụng trong luận án là Chi-Square (CHI).

Đánh giá hiệu quả các mô hình phân lớp mã độc

Sau khi xây dựng một mô hình học máy, chúng ta cần đánh giá để xác định mức độ hiệu quả của mô hình cũng như so sánh khả năng của các mô hình để chọn ra mô hình có hiệu quả cao nhất. Hiệu năng của một mô hình được đánh giá dựa trên tập dữ liệu kiểm thử (test data). Với bài toán phân lớp có hai lớp có bốn độ đo cơ bản trong mô hình phân lớp hai lớp này, được thể hiện trong ma trận nhầm lẫn (confusion matrix) như Bảng 2.2.

Bảng 2.2: Ma trận nhầm lẫn.

Số mẫu	Dự đoán là Dương tính	Dự đoán là Âm tính
Thực tế là Dương tính	Dương tính thật (TP)	Âm tính giả (FN)
Thực tế là Âm tính	Dương tính giả (FP)	Âm tính thật (TN)

Từ bốn độ đo cơ bản trên, xây dựng nên các độ đo kết hợp khác. Một số độ đo phổ biến: độ chính xác (Accuracy - AC), Tỉ lệ Âm tính giả (False Negative Rate - FNR) và Tỉ lệ Dương tính giả (False Positive Rate - FPR), Precision (tính chính xác), Recall (tính hữu dụng), F1-score.

2.2 Các hệ thống nhúng

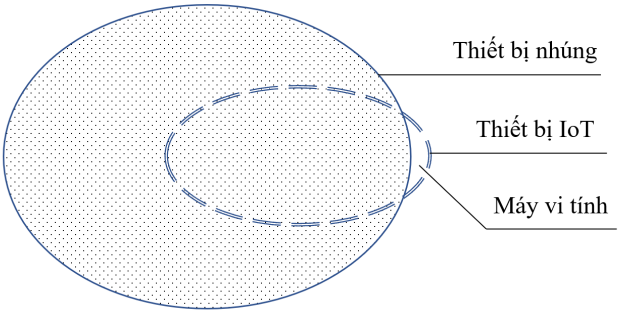
2.2.1 Các thiết bị nhúng

Hệ nhúng (Embedded System) là hệ thống máy tính được thiết kế cho chức năng điều khiển và được đặt (nhúng, cài, gắn) trong một hệ thống lớn hơn.

Một khái niệm được sử dụng phổ biến ngày nay là các thiết bị Internet vạn vật (Internet of things - IoT). IoT là tập hợp các cảm biến và bộ điều khiển nhúng trong các thiết bị được liên kết thông qua mạng có dây và không dây. Theo nghĩa rộng IoT là tất cả các thiết bị kết nối với nhau thông qua mạng Internet. Như vậy, các thiết bị IoT gồm thiết bị nhúng và máy vi tính nhưng về số lượng thì đa phần là các thiết bị nhúng và IoT là một trong những ứng dụng lớn của các hệ thống nhúng. Sự giao thoa của thiết bị IoT và thiết bị nhúng được hiểu như trong Hình 2.1. Trong phạm vi luận án này, đề cập đến các thiết bị IoT hay thiết bị nhúng để chỉ các thiết bị nhúng có kết nối Internet.

2.2.2 Các hệ điều hành nhúng

Phần mềm nhúng là phần mềm máy tính được phát triển chuyên biệt nhúng vào hệ thống mà nó điều khiển nhằm đảm bảo tương tác giữa hệ thống đó với môi trường bên



Hình 2.1: Sự giao thoa của thiết bị nhúng, thiết bị IoT và máy vi tính.

ngoài. Hệ nhúng có chức năng chuyên biệt nên không sử dụng các hệ điều hành vạn năng của máy vi tính như Windows, Linux mà sử dụng các hệ điều hành nhúng (Embedded OS). Hiện nay có nhiều hệ điều hành nhúng khác nhau như Embedded Linux, Android, Windows CE, VxWorks, uOS/C, QNX v.v.

Hệ điều hành Linux nhúng hiện chiếm đa số trong phần sụn các thiết bị nhúng, phiên bản nhân Linux được sử dụng phổ biến là 2.6 và 3.0. Hệ điều hành hoàn chỉnh nhưng tối giản trong các phần sụn loại này thông thường chứa các tập tin hệ thống như: Bộ tải (Boot loader), nhân hệ điều hành (kernel), bản ảnh tệp tin hệ thống, tệp tin tài nguyên và hỗ trợ (Resources and support files), giao diện web và máy chủ web (Web-server/web-interface).

Phần sụn là phần mềm được tích hợp vào một sản phẩm hệ thống nhúng và được lưu trữ trong bộ nhớ không biến đổi (nonvolatile memory), chẳng hạn như ROM, EPROM, E2PROM hoặc FLASH. Trong công nghiệp, phần sụn được hiểu như là phần mềm nhúng (embedded software) hoặc phần mềm mức thấp (low-level software). Trong luận án này, loại phần sụn được nghiên cứu là phần sụn đầy đủ.

2.3 Phân tích mã độc trong các hệ thống nhúng

2.3.1 Mã độc trong các hệ thống nhúng

Chúng tôi đề xuất khái niệm như sau: mã độc trong các hệ thống nhúng (mã độc nhúng) là những chương trình được cài cắm sẵn vào phần sụn một cách không chính thống hoặc lây lan vào thiết bị nhúng trong quá trình hoạt động với mục đích thu thập thông tin hoặc lợi dụng các hệ thống nhúng này lây lan, khai thác, tấn công các hệ thống khác.

Các đặc tính của mã độc trong các hệ thống nhúng là không thường trú, đa số là mã độc botnet, ít sử dụng các kỹ thuật gây rối và có các phiên bản mã độc hoạt động đa nền tảng. Các mã độc khi lây nhiễm sẽ tồn tại trên các bộ nhớ tạm thời, khi khởi động lại thiết bị sẽ bị xoá. Mã độc trên thiết bị nhúng không đa dạng như trên máy tính, đa số mã độc phổ biến trên thiết bị nhúng hiện nay có cơ chế hoạt động tương tự mã độc botnet như Mirai. Các thiết bị IoT có đặc điểm là hạn chế về tài nguyên, giảm thiểu tối

đa các chức năng phụ và hoạt động trong môi trường tương đối độc lập, đóng gói, ít bị giám sát nên các mã độc trên IoT cũng theo hướng đơn giản, không phức tạp như mã độc truyền thống. Các thiết bị IoT sử dụng nhiều loại kiến trúc vi xử lý khác nhau như Intel 80386, MIPS, ARM v.v. do đó mã độc trên IoT cũng có xu hướng đa kiến trúc phát triển mạnh trong thời gian gần đây.

2.3.2 Môi trường và công cụ phân tích mã độc trên hệ thống nhúng

Angr là bộ công cụ phân tích nhị phân mã nguồn mở, hỗ trợ phân tích các tập tin nhị phân đa kiến trúc như x86, AMD64, MIPS, PowerPC, ARM. Nó có các chức năng thực hiện một loạt các kỹ thuật phân tích tĩnh như: dịch ngược mã máy và biểu diễn ở dạng ngôn ngữ trung gian các tập tin nhị phân, phân tích dòng điều khiển, mô hình hóa dòng thực thi, mô hình hóa dữ liệu và thực hiện xử lý biểu tượng.

QEMU là phần mềm mô phỏng và ảo hoá máy tính nguồn mở, có khả năng hỗ trợ mô phỏng 26 kiến trúc vi xử lý (CPU) khác nhau, đặc biệt các kiến trúc vi xử lý nhúng cho IoT như MIPS, ARM v.v. và hỗ trợ các hệ điều hành Windows, Linux.

Firmadyne là hệ thống mô phỏng dựa trên phần sụn để phân tích động quy mô lớn và tự động cho các thiết bị nhúng. Nghiên cứu này cũng giải quyết việc mô phỏng thiết bị ngoại vi đặc trưng của thiết bị IoT là thiết bị lưu trữ cấu hình trong bộ nhớ không ổn định (NVRAM) dựa trên mô phỏng. Firmadyne sử dụng các tệp phần sụn được phân phối của các nhà cung cấp, tự động giải nén nội dung để xác định nhân (kernel) và giải nén hệ thống tập tin trong phần sụn.

Sandbox được hiểu là một hệ thống phân tích phần mềm độc hại sử dụng môi trường thử nghiệm có kiểm soát, nơi mà có thể nghiên cứu và phân tích được các hành vi của mã độc. Sandbox ảo dựa trên công nghệ mô phỏng, ảo hoá được sử dụng phổ biến. Nhiều giải pháp sandbox tương đối hoàn chỉnh cho phân tích mã độc trên máy tính thông thường đã được xây dựng như Cuckoo, Detux, IoTPoT.

2.3.3 Các phương pháp phân tích mã độc trong hệ thống nhúng

Dovom đề xuất phương pháp cho phát hiện mã độc trên IoT dựa trên phương pháp cây mẫu mờ (Fuzzy Pattern Tree). Phương pháp trích chọn đặc trưng được sử dụng là dựa trên opcode kết hợp với đồ thị dòng điều khiển, bằng cách lựa chọn các véc tơ riêng của đồ thị dòng điều khiển. Sau đó, nghiên cứu này đề xuất một số cách cài đặt cho cây mẫu mờ, cây mẫu mờ nhanh và đánh giá độ chính xác, thời gian thực hiện. Tuy nhiên, thiết bị IoT được đề cập là các thiết bị biên hay đầu cuối trong mạng IoT (Edge computing), bao gồm cả máy tính truyền thống, di động và các thiết bị nhúng khác.

Các nghiên cứu về mã độc trên thiết bị nhúng thời gian qua tập trung mã độc trên thiết bị sử dụng hệ điều hành Android, số các nghiên cứu này chiếm hơn 40% các nghiên cứu liên quan đến phân tích mã độc dựa trên các kỹ thuật học máy so với các chủ

đề khác. Các phương pháp phân tích tĩnh được nghiên cứu như: phát hiện dựa trên trên chữ ký, phát hiện dựa trên quyền (Permission based detection), phát hiện dựa trên Dalvik Bytecode. Các phương pháp động tiếp cận dưới các hướng phát hiện dựa trên bất thường (Anomaly based detection), phân tích theo vết (Taint anslysis) và phát hiện dựa trên mô phỏng (Emulation based detection).

Ngoài các thiết bị sử dụng hệ điều hành Android, một số lượng lớn các thiết bị nhúng sử dụng hệ điều hành Linux nhúng. Một trong các tập dữ liệu IoT đặc trưng cho mã độc trong môi trường này được thu thập bởi Honeypot chuyên dụng của IoT là IoTPOT. Tập dữ liệu này gồm hơn 4.000 mẫu thuộc nhiều họ khác nhau như Linux.Gafgyt, Mirai, Trojan.Linux.Fgt v.v. Nhiều nghiên cứu đề xuất các thuật toán và cũng sử dụng tập mẫu này để đánh giá theo tập mã độc trên thiết bị IoT. Pa và cộng sự cũng đã xây dựng một IoT sandbox đặt tên là IoTBox để phân tích động các cuộc tấn công của phần mềm độc hại khai thác dịch vụ Telnet trên các thiết bị IoT chạy trên các nền tảng kiến trúc CPU khác nhau. Su và cộng sự đã đề xuất phương pháp pháp phân lớp mã độc nhẹ (Lightweight Classification) dựa trên nhận dạng ảnh. Từ các mẫu nhị phân ban đầu được chuyển thành ảnh xám, sau đó áp dụng các kỹ thuật nhận dạng ảnh để phát hiện ảnh đó được tạo ra từ mã độc nào hay mã sạch. Nguyen và cộng sự đề xuất phương pháp trích xuất đặc trưng cải tiến dựa trên PSI và CFG. Sau đó xây dựng mạng nơ ron tích chập CNN để phân lớp các họ mã độc.

2.4 Tổng kết chương

Trong Chương 2, luận án đã trình bày tổng quan về phân tích mã độc nhúng. Các nội dung cụ chính được đề cập gồm tổng quan về phân tích mã độc, các thiết bị nhúng và phân tích mã độc nhúng. Trong quá trình nghiên cứu của luận án, khái niệm thiết bị nhúng và thiết bị IoT được hiểu đồng nhất. Thêm vào đó, phần sụn với vai trò là thiết bị nhúng đặc biệt cũng được đề cập chi tiết hơn về cấu trúc, đặc điểm. Đối tượng nghiên cứu của luận án là mã độc hoạt động trên các thiết bị IoT sử dụng hệ điều hành Linux nhúng (mã độc nhúng).

Chương 3

MỘT GIẢI PHÁP PHÂN TÍCH ĐỘNG ĐỐI VỚI CÁC MÃ ĐỘC NHÚNG

Trong chương này, chúng tôi trình bày môi trường phân tích động các phần mềm nhúng F-Sandbox (Firmware based sandbox). [1] Đây là môi trường phân tích động chuyên dụng cho hệ thống nhúng kiểu mới được xây dựng nhằm tạo môi trường thích hợp nhất để mô phỏng môi trường hoạt động đa dạng của thiết bị nhúng, giúp các phần mềm nhúng có thể kích hoạt đầy đủ các chức năng. Môi trường thích hợp này được tạo ra dựa trên việc kế thừa môi trường của phần sụn các thiết bị thực tế thay cho việc sử dụng các môi trường chuẩn duy nhất như các nghiên cứu trước đây. Trên cơ sở F-Sandbox, chúng tôi đề xuất hai quy trình phát hiện và phân lớp mã độc dựa trên lời gọi hệ thống được trích xuất từ F-Sandbox.

3.1 Đặt vấn đề

3.1.1 Mục tiêu và các yêu cầu đặt ra

Mục tiêu đặt ra là phát triển sandbox có khả năng kích hoạt tối đa các chức năng của các mẫu phần mềm nhúng được thực thi trong sandbox và thu thập đầy đủ các dữ liệu về hành vi của các phần mềm này. Từ dữ liệu thu được của sandbox, phải xây dựng được quy trình và lựa chọn các phương pháp phù hợp để tiền xử lý mẫu, thu thập hành vi qua sandbox, trích chọn đặc trưng từ dữ liệu các hành vi và huấn luyện/phát hiện và có khả năng phát hiện, phân lớp mã độc với độ chính xác cao.

3.1.2 Các thách thức cần giải quyết

Thứ nhất, xây dựng một môi trường tương tự như các thiết bị nhúng là thách thức để kích hoạt mã độc nhúng gặp nhiều khó khăn như việc xây dựng sandbox cho môi trường máy tính truyền thống. Thứ hai, các thông tin cấu hình và môi trường (thư viện, tệp tin hỗ trợ, tệp tin tài nguyên) được hỗ trợ trong phần sụn các thiết bị nhúng rất đa dạng, do đó cần phát triển chức năng điều khiển sandbox có khả năng tự động thích nghi tương ứng để điều khiển hiệu quả sandbox. Việc mô phỏng NVRAM đã mang lại hiệu quả tốt, song tỉ lệ thành công vẫn còn hạn chế, lý do là một số phần sụn yêu cầu

1. Các phương pháp, kỹ thuật, giải thuật và dữ liệu được xây dựng trong luận án này có tiếp đầu ngữ là F, biểu thị dùng cho xử lý phần sụn (Firrmware)

thông tin chính xác từ NVRAM của đúng thiết bị vật lý. Thứ ba, *Strace* có sẵn nhiều chức năng cho giám sát hành vi hệ thống và được sử dụng phổ biến, song có hạn chế là không giám sát được tất cả tiến trình hệ thống, công cụ này bị một số mã độc phát hiện và chuyển sang chế độ phòng vệ, không kích hoạt các chức năng mã độc. Thứ tư, tập mẫu mã độc và tập sạch để phục vụ cho nghiên cứu còn nhiều hạn chế.

3.1.3 Các công cụ, bộ dữ liệu đã có

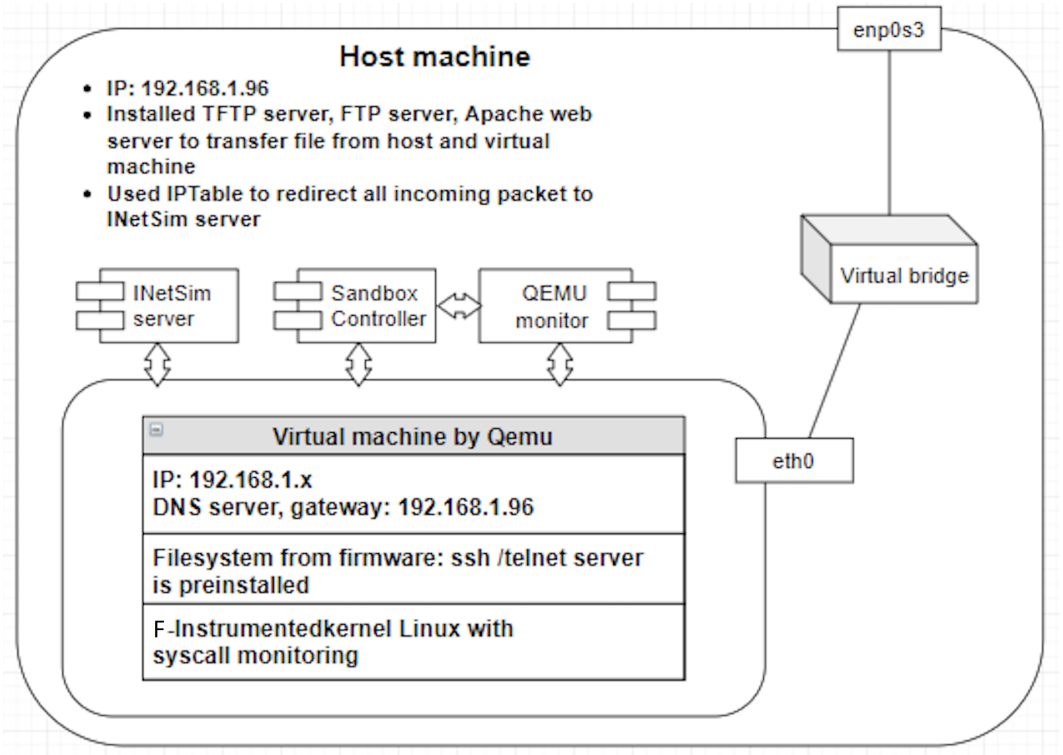
Các giải pháp sandbox cho phân tích mã độc trên máy tính thông thường như Limon, CwSandbox không áp dụng cho việc phân tích mã độc trên các thiết bị IoT do khác về kiến trúc vi xử lý và hệ điều hành. Sandbox cho các thiết bị IoT chủ yếu sử dụng nền tảng mô phỏng QEMU như Cuckoo, Detux, IoTBox. IoTBox là sandbox phân tích các hành mã độc dựa trên giao thức Telnet để thực hiện các cuộc tấn cong DDoS. Chang và cộng sự đề xuất xây dựng IoT sandbox dựa trên QEMU hỗ trợ 9 loại kiến trúc nhúng khác nhau. Tuy nhiên, các sandbox này tập trung phân tích mã độc dựa trên thông tin mạng, không giám sát hành vi hệ thống nên sẽ không pháp hiện được các mã độc trên IoT có hành vi tác động hệ thống nhiều như Linux/theMoon. Các sandbox này cũng không đề cập đến mô phỏng NVRAM. Các nghiên cứu phát hiện mã độc qua lời gọi hệ thống cho kết quả phát hiện tốt trên nhiều nền tảng, song chưa có nghiên cứu áp dụng cho MIPS ELF. Hiện này, chưa có sandbox nào hỗ trợ các thiết bị IoT sử dụng vi xử lý MIPS có chức năng thu thập syscall và giả lập Internet phục vụ phân tích mã độc.

3.1.4 Cách tiếp cận của chúng tôi

Về mô phỏng mạng Internet, chúng tôi kế thừa cách tiếp cận các nghiên cứu trước trong mô phỏng mạng Internet bằng InetSim. Về công cụ giám sát, luận án là sử dụng *Kprobe* để giám sát các lời gọi hệ thống bằng cách sửa nhân Linux như Firmadyne. Chúng tôi phát triển nhân tích hợp *Kprobe* để có khả năng nghe 30 lời gọi hệ thống có tần suất xuất hiện nhiều nhất trong khi thực thi các mẫu mã độc để phục vụ cho mục đích phát hiện/phân lớp mã độc. Về mô phỏng thiết bị ngoại vi, thành phần NVRAM được mô phỏng dựa trên Firmadyne hoặc kết hợp với thông tin NVRAM thực tế. Một số phần sụn không hoạt động được với NVRAM mô phỏng, do đó chúng tôi phát triển công cụ F-Extractor để thu thập phần sụn trực tiếp từ thiết bị, trong đó có NVRAM. Về việc xây dựng môi trường thích nghi, các nghiên cứu trước đây thường sử dụng bản ảnh Linux chuẩn, một số nghiên cứu sử dụng môi trường chuyên dụng hơn cho thiết bị nhúng là OpenWrt, các bản ảnh QEMU sử dụng trong F-Sandbox chứa hệ thống tệp tin được trích xuất từ các bản phần sụn thực tế được bóc tách bằng công cụ F-Extractor (FE) do chúng tôi phát triển. Cách tiếp cận này sẽ giúp F-Sandbox tạo được môi trường như các phần sụn đã có. Về bộ điều khiển sandbox, chúng tôi xây dựng bộ điều khiển có khả năng hỗ trợ nhiều giao thức có sẵn trong các phần sụn để có khả năng điều khiển, truyền nhận tệp tin với máy ảo một cách hiệu quả. Các thành phần mô phỏng mạng Internet và điều khiển sandbox được phát triển để tự động các bước phân tích.

3.2 Cấu trúc môi trường phân tích động F-Sandbox

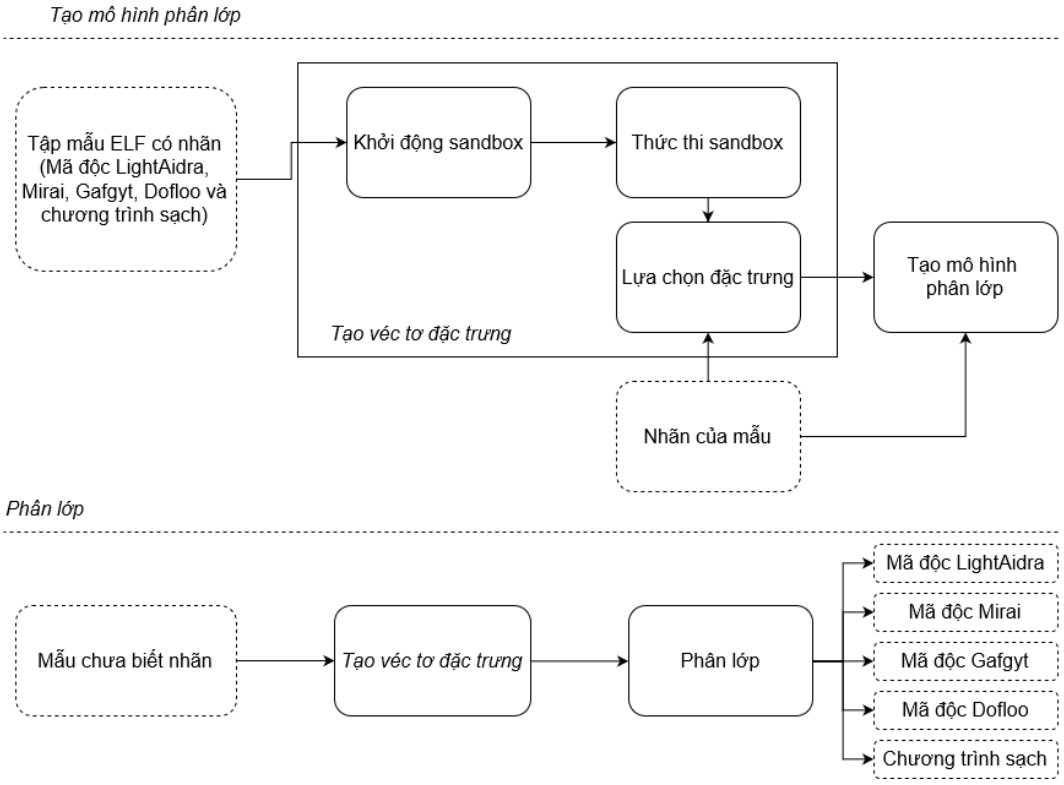
F-Sandbox thu thập hành vi hệ thống syscall thông qua công cụ Kprobes được tích hợp trong nhân hệ điều hành Linux. INetSim/pyNetsim để cung cấp mô phỏng dịch vụ mạng giúp cho mã độc tương tác bộc lộ các hành vi. Kết quả thu thập được của F-Sandbox sẽ là các thông tin phân tích tĩnh của mẫu thử, dữ liệu mạng do mẫu sinh ra dưới dạng tệp tin .pcap, log iNetsim/pyNetsim về các thông tin tương tác với Internet và các hành vi hệ thống syscall dưới dạng tệp tin syscall log. Cấu trúc F-Sandbox có ba thành phần chính là Điều khiển sandbox(Sandbox Controller), máy ảo (virtual machine) và máy chủ giả lập Internet (INetSim server) được thể hiện như trong Hình 3.1.



Hình 3.1: Cấu trúc F-Sandbox.

— Bộ điều khiển sandbox (Sandbox Controller) tương tác với QEMU monitor thông qua việc gọi các lệnh hiển thị cấu hình mạng, restore snapshot. Và sanbox controller tương tác với máy ảo thông qua việc gọi các thủ tục SSH/Telnet tới máy ảo, truyền tệp tin thực thi từ máy thật vào trong, cấp quyền thực thi, yêu cầu thực thi file, tải tệp tin dữ liệu thu được sau khi chạy mẫu từ máy ảo. *Máy ảo* dựa trên QEMU, bao gồm hai thành phần bản ảnh QEMU và nhân Linux. Nhân Linux được sử dụng trong F-Sandbox không phải là nhân Linux chuẩn mà sử dụng F-Kernel.

— Máy chủ mô phỏng Internet: Máy ảo tương tác với INetSim server thông qua việc gửi các request (http,ftp,dns...) và nhận lại các phản hồi giả (fake respond) từ INetSim. Các thành phần enp0s3, br0, tap0 là các giao diện kết nối mạng.



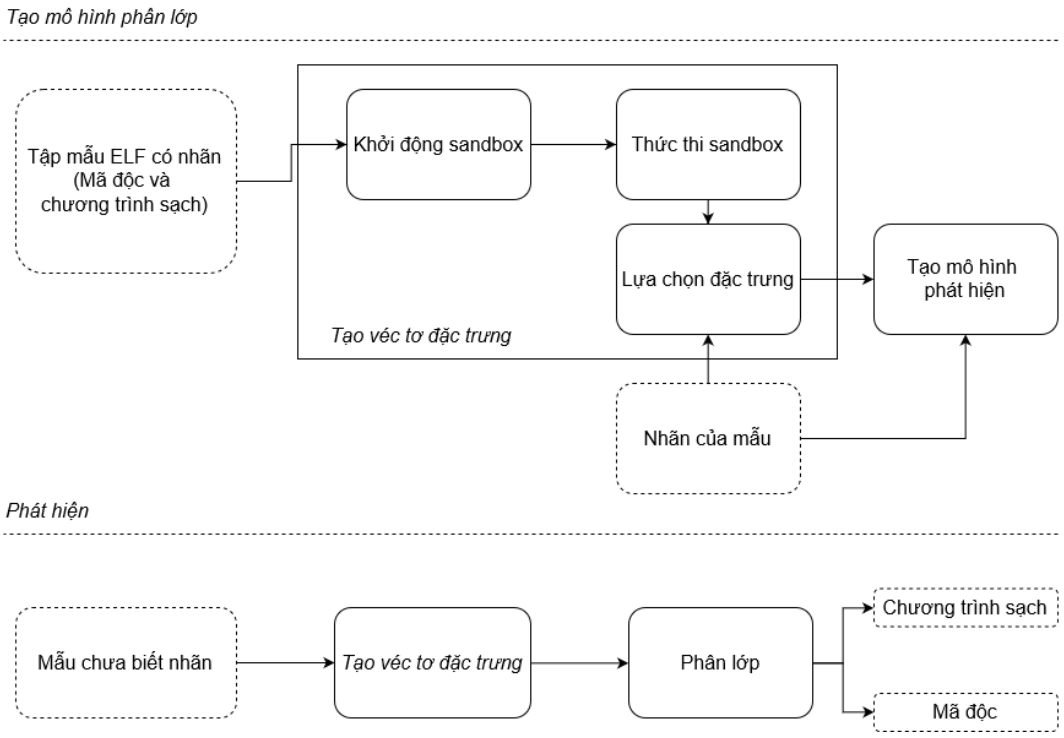
Hình 3.2: Tiến trình đề xuất để phân lớp mã độc dựa trên F-Sandbox và học máy.

3.3 Quy trình phân lớp mã độc dựa trên F-Sandbox

Dựa trên F-Sandbox, chúng tôi đề xuất quy trình phân lớp mã độc trên thiết bị nhúng gồm các bước như trong Hình 3.2. Quy trình này gồm hai pha: pha tạo mô hình và pha phân lớp. Trong pha tạo mô hình, các mẫu ELF đã gán nhãn được trích xuất đặc trưng bằng chức năng tạo véc tơ đặc trưng, sau đó những véc tơ đặc trưng này cùng với nhãn của nó sẽ là đầu vào để huấn luyện các mô hình học máy. Chức năng tạo véc tơ đặc trưng gồm ba bước: khởi tạo F-Sandbox, thực thi sandbox và trích xuất đặc trưng. Trong pha phân lớp, những mẫu chưa biết nhãn sẽ được trích xuất đặc trưng bởi chức năng tạo véc tơ đặc trưng với các tham số đã được sinh ra từ pha trước, sau đó những véc tơ này sẽ được đưa vào mô hình học máy đã được tạo ra từ pha trước và xác định nhãn của mẫu đưa vào.

3.4 Quy trình phát hiện mã độc dựa trên F-sandbox

Chúng tôi đề xuất quy trình gồm bốn bước để phát hiện mã độc như trong Hình 3.3. Quy trình phát hiện mã độc trong các thiết bị nhúng sử dụng kiến trúc MIPS dựa trên lời gọi hệ thống bằng kỹ thuật phân loại một lớp SVM. Các chương trình được chạy trong F-Sandbox để thu thập các syscall được gọi, các syscall được biểu diễn dưới dạng đặc trưng n-gram, sau đó sử dụng phương pháp PCA để giảm số chiều véc tơ đặc trưng.



Hình 3.3: Quy trình phát hiện mã độc trên thiết bị IoT dựa trên phân tích động.

Do đặc thù tập dữ liệu thực nghiệm có số lượng dữ liệu giữa tập sạch và tập mã độc lệch nhau nên chúng tôi sử dụng phân loại một lớp SVM.

3.5 Thử nghiệm

Chúng tôi sử dụng tập dữ liệu thực nghiệm gồm 3.773 mẫu được phát hiện với tỉ lệ cao (tức tối thiểu 19/67 phần mềm, trong đó cả bốn phần mềm uy tín là Kaspersky, Avast, Avg, Symantec đều nhận diện đây là mã độc), gọi là tập F-IoT. Các mẫu được lấy từ Detux, VirusShare, IoTPOT. Tập mã sạch được thu thập từ các chương trình cơ bản có sẵn trên Linux nhúng, được tích hợp sẵn trong *busybox* và một số ứng dụng cơ bản trên nền MIPS. Số lượng mẫu sạch thu được là 258 mẫu.

3.5.1 Kết quả thử nghiệm quy trình phân lớp mã độc

Hình 3.4 trình bày kết quả các độ đo F1-Macro, F1-Micro và F1-weight với các phương pháp học máy khác nhau và ngưỡng độ dài của nhật ký lời gọi hệ thống khác nhau. Nhìn chung, kết quả cao đồng đều ở các chỉ số và với các thuật toán. Từ thực nghiệm rút ra một số nhận định: Với các mẫu MIPS ELF, quy trình của chúng tôi đạt kết quả tốt nhất với phương pháp trích chọn đặc trưng 2-gram và phương pháp phân lớp RF, tại ngưỡng độ dài nhật ký lời gọi hệ thống là 400; Kết quả các phương pháp đều cao, chứng tỏ F-Sandbox hoạt động hiệu quả và các bước xây dựng quy trình là đúng.

Len	F1 - Macro			F1 - Micro			F1 - Weight		
	RF	SVM	NB	RF	SVM	NB	RF	SVM	NB
50	85.53	80.31	67.71	92.89	91.69	68.87	91.98	89.63	67.06
100	86.10	85.15	69.51	93.03	92.02	69.46	92.23	91.32	67.85
150	85.45	83.17	70.69	93.9	92.45	70.49	92.79	91.33	69.26
200	85.98	83.04	78.51	93.95	92.71	84.96	92.95	91.52	83.88
250	85.58	82.83	78.29	94.00	92.63	84.90	92.88	91.44	83.78
300	85.53	81.28	78.21	94.15	92.83	84.95	93.01	91.13	83.91
350	85.05	81.30	73.86	93.73	92.49	79.79	92.65	90.93	78.50
400	**92.55**	88.49	79.24	**97.56**	96.13	93.67	**97.44**	96.01	93.40
450	92.22	90.48	**80.09**	97.53	96.08	**94.17**	97.42	96.04	**94.00**
500	92.41	**90.94**	79.79	97.51	**96.63**	93.97	97.39	**96.64**	93.76

Hình 3.4: F1-Macro, F1-Micro và F1-weight với trích chọn đặc trưng 2-gram.

3.5.2 Kết quả thử nghiệm quy trình phát hiện mã độc

Thử nghiệm với 5 tập dữ liệu sinh ra từ tập F-IoT do chọn 5 ngưỡng của độ dài syscall log khác nhau là 50, 100, 200, 300, 400, 500. Với đặc trưng 1-gram, 2-gram thu được, chúng tôi thử nghiệm giảm chiều bằng PCA với K lần lượt là 20, 40, 80. Riêng với 1-gram, do số chiều là 345 nên chúng tôi tiến hành thêm thử nghiệm không áp dụng giảm chiều bằng PCA.

Từ kết quả thực nghiệm với độ đo F1, AP, AC cho thấy: Các chỉ số mô hình tốt với giá trị F1 cao nhất đạt ngưỡng 0,977; trích xuất các syscall log có độ dài 200-300 cho khả năng phát hiện tốt nhất; sử dụng phương pháp 1-gram đơn giản, không gian đặc trưng ít nhưng vẫn đạt hiệu quả rất cao trong trường hợp này.

3.6 Tổng kết chương

Chương này đã trình bày về môi trường phân tích động kiểu mới cho thiết bị nhúng F-Sandbox. Nó có khả năng tự động tạo các môi trường phong phú dựa trên các phần sụn của thiết bị nhúng, qua đó kích hoạt tốt các chức năng của các phần mềm nhúng, thu thập hành vi hệ thống và hành vi mạng phục vụ phát hiện, phân lớp mã độc nhúng. Từ F-Sandbox, chúng tôi đề xuất hai quy trình phân tích và phát hiện mã độc trên thiết bị nhúng sử dụng kiến trúc MIPS và hệ điều hành Linux (MIPS ELF). Các quy trình đều cho kết quả tốt khi thực nghiệm với bộ dữ liệu lớn F-IoT do chúng tôi đề xuất, chứng tỏ F-Sandbox hoạt động hiệu quả, các phương pháp sử dụng khi đề xuất các quy trình là đúng đắn.

Kết quả nghiên cứu trên được công bố tại Tạp chí quốc tế [PP1, PP2], Tạp chí Thông tin và truyền thông [PP3], hội nghị quốc tế IEEE [PP4], hội nghị trong nước [PP8].

Chương 4

PHƯƠNG PHÁP TRÍCH XUẤT ĐẶC TRƯNG DÒNG ĐIỀU KHIỂN CHO PHÁT HIỆN CÁC MÃ ĐỘC NHÚNG

Chương này đề xuất hai thuật toán trích xuất đặc trưng hiệu quả cho việc phát hiện mã độc trên thiết bị nhúng giải quyết hai thách thức trên: phương pháp trích xuất đặc trưng dòng điều khiển dựa trên mã thực thi bằng thuật toán quy hoạch động (CFD) và phương pháp trích xuất đặc trưng dòng điều khiển dựa trên ngôn ngữ trung gian Vex (CFDVex). Thuật toán CFD cho khả năng phát hiện tốt mã độc nhúng trên một kiến trúc vi xử lý và CFDVex có thể phát hiện mã độc đa kiến trúc.

4.1 Đặt vấn đề

4.1.1 Mục tiêu và các yêu cầu đặt ra

Thứ nhất, cần cải tiến các thuật toán trích xuất đặc trưng dòng điều khiển dựa trên mã thực thi có khả năng phát hiện mã độc đảm bảo độ chính xác cao, giảm độ phức tạp và tiêu thụ ít tài nguyên. Thứ hai, phát triển thuật toán trích xuất đặc trưng hiệu quả để phát hiện mã độc đa kiến trúc, với tốc độ nhanh, độ chính xác cao với các kịch bản đánh giá đầy đủ trên tập dữ liệu lớn để chứng minh tính hiệu quả. Các kịch bản đánh giá phải bao gồm phát hiện hỗn hợp, so sánh với các phương pháp khác cùng một kiến trúc vi xử lý, đặc biệt khi mô hình được huấn luyện từ dữ liệu của một kiến trúc cũ và có khả năng phát hiện mã độc trên kiến trúc mới. Thứ ba, cần thu thập tập dữ liệu đầy đủ các loại kiến trúc và số lượng đủ lớn là cần thiết cho việc nghiên cứu mã độc đa kiến trúc và đánh giá chính xác thuật toán đề xuất.

4.1.2 Các thách thức cần giải quyết

Thứ nhất, cần xây dựng thuật toán trích xuất dòng điều khiển có hiệu quả (độ chính xác cao và độ dương tính giả thấp), có độ phức tạp đa thức và sử dụng ít tài nguyên hơn các phương pháp cũ. Các phương pháp xử lý trên đồ thị có hướng thường phải bỏ chu trình để đưa về đồ thị có hướng không có chu trình. Thứ hai, lựa chọn loại đặc trưng có khả năng phát hiện mã độc đa kiến trúc hiệu quả, có các công cụ hỗ trợ đầy đủ, đồng

bộ. Thứ ba, thu thập được số lượng mẫu mã độc nhúng, đặc biệt là mẫu sạch trên các thiết bị nhúng với số lượng lớn và đầy đủ chủng loại.

4.1.3 Các công cụ, bộ dữ liệu đã có

Sử dụng mã thực thi dựa trên văn bản (Text-based extraction methods) làm đặc trưng để phát hiện mã độc đã được nhiều nghiên cứu đề cập. Ding và cộng sự đề xuất trích xuất đặc trưng dòng điều khiển dựa trên giải lệnh thực thi (Control flow-based opcode - CFO) đạt được độ chính xác cao hơn các phương pháp dựa trên văn bản trong phát hiện mã độc. Tuy nhiên, họ phải đối mặt với bài toán có độ phức tạp NP-hard trong duyệt đồ thị là liệt kê tất cả đường đi từ đỉnh u đến đỉnh v trong đồ thị. Do đó, phương pháp này chỉ có thể áp dụng với các tập tin thực thi đơn giản, không phù hợp với các tập tin phức tạp, có kích thước lớn.

Costin và cộng sự đã thu thập hơn 32.000 bản ảnh phần sụn và tiến hành phân tích tĩnh chúng. Pa và cộng sự thu thập tập dữ liệu mã độc trên IoT gọi là IoTPOT, với hơn 4.000 mã độc IoT, phổ biến như Tsunami, Mirai, Bashlite. Detux thu thập được hơn 9.000 mẫu mã độc nhúng. Brash thu thập được 1.078 mẫu sạch và 128 mẫu mã độc cho ARM.

Nhiều phương pháp phát hiện mã độc đa nền tảng cũng đã được đề cập như phương pháp tạo chữ ký mới cho mã độc IoT đa nền tảng của Alhanahnah, sử dụng ngôn ngữ trung gian chuyên phân tích tệp tin nhị phân của Kim, đề xuất ngôn ngữ REIL với khả năng sử dụng thêm các cờ (flags) trong tính toán của Sepp hay đề xuất một ngôn ngữ trung gian MAIL chuyên cho phân tích mã độc siêu đa hình của Alam. Tuy nhiên, chúng tôi nhận thấy chưa có nghiên cứu nào giải quyết bài toán phát hiện mã độc đa kiến trúc vi xử lý với tập kịch bản đánh giá đầy đủ trên tập dữ liệu đủ lớn.

Vex là ngôn ngữ trung gian được sử dụng trong Valgrind, một công cụ phân tích nhị phân phổ biến và nổi tiếng. Đây là ngôn ngữ độc lập, không bị tác động bởi ngôn ngữ đích, được hỗ trợ tốt trong quá trình dịch mã. Vex được sử dụng trong BitBlaze đặc biệt là công cụ chuyên phân tích tĩnh IoT là Angr.

4.2 Phương pháp CFD

4.2.1 Cách tiếp cận của chúng tôi

Về vấn đề trích xuất đặc trưng dòng điều khiển dựa trên mã thực thi, chúng tôi đề xuất thuật toán trích xuất đặc trưng dựa trên ý tưởng của quy hoạch động, gọi là CFD. Đầu tiên, từ CFG của mẫu, chúng tôi đề xuất thuật toán trích tạo đồ thị không chu trình AG (Acyclic Graph). Tiếp theo, đồ thị thực thi EDAG (Execution AG) được xây dựng từ AG. Cuối cùng từ AG xây dựng thuật toán trích xuất đặc trưng DFO của mẫu.

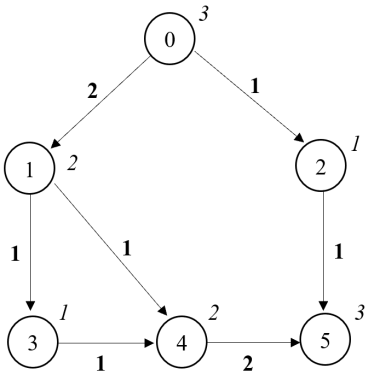
Về tập dữ liệu cho thiết bị nhúng, chúng tôi hướng tới xây dựng bộ dữ liệu đầy đủ các kiến trúc thịnh hành, số lượng lớn, đầy đủ cả mã độc và chương trình sạch. Mã độc được thu thập và lọc từ các nguồn có sẵn, chương trình sạch sẽ được trích xuất từ các phần sụn bằng công cụ F-Toolkit.

4.2.2 Thuật toán trích xuất đặc trưng CFD

Thuật toán 4.1 Thuật toán trích xuất đặc trưng dòng điều khiển - CFD.

```
   Input: Tệp tin thực thi: exeFile
   Output: feature_vector: Đặc trưng dựa trên dòng điều khiển với
n-gram
 1: graph = GetCfgFromFile (exeFile)
 2: dag = ConstructDag(graph)
 3: edag = ConstrucEdag(dag)
 4: feature_vector = ExtractControlFlowBasedFeature(edag)
   Return: feature_vector
```

Chúng tôi đề xuất một phương pháp dựa trên quy hoạch động để xây dựng EDAG với nhãn của đỉnh và trọng số của cạnh chứa số lượng đường thực thi đi qua. Các bước được trình bày trong Thuật toán 4.1. Đầu tiên, một DAG được tạo ra từ CFG của một chương trình thực thi. Sau đó, một EDAG được xây dựng từ DAG. Khi xây dựng EDAG số đường đi hiện tại được xây dựng dựa trên các kết quả trước đó, vì vậy thuật toán này thực hiện nhanh hơn phương pháp của Ding và cộng sự đề xuất. Cuối cùng, các đặc trưng dựa trên dòng điều khiển được trích xuất từ EDAG. Độ phức tạp của thuật toán CFD là O(N^2).



Hình 4.1: Ví dụ về EDAG

Xây dựng đồ thị có hướng không chu trình DAG dựa trên ý tưởng DFS. Xuất phát từ đỉnh gốc, thuật toán sẽ thăm các đỉnh kề với đỉnh đó, lựa chọn các cạnh không tạo thành chu trình để xây dựng DAG.

EDAG có D chứa các trọng số của cạnh. Với cạnh *(u,v)*, *D[u,v]* là số lượng các đường thực thi có chứa cạnh *(u,v)* và *D[v,u]* là số đường thực thi ngược (tức đường đi ngược chiều cạnh từ lá về đỉnh gốc) chứa cạnh *(u,v)*. Thuật toán xây dựng EDAG của tệp tin

thực thi gồm 2 pha: pha đi ngược để tính số đường thực thi ngược từ các lá về đỉnh gốc, và pha đi xuôi để gán trọng số cho các cạnh là số đường thực thi chứa cạnh đó.

4.3 Phương pháp CFDVex

4.3.1 Cách tiếp cận của chúng tôi

Về bài toán phát hiện mã độc đa kiến trúc, trên cơ sở ý tưởng thuật toán CFD, chúng tôi đề xuất sử dụng đặc trưng dòng điều khiển dựa trên ngôn ngữ trung gian Vex (FDV) và phương pháp trích xuất đặc trưng này là CFDVex. Tập các lệnh thực thi là khác nhau với các kiến trúc khác nhau, nên chúng tôi thay lệnh thực thi bằng ngôn ngữ trung gian Vex để dùng chung cho các kiến trúc vi xử lý khác nhau. CFDVex sẽ tính n-gram của chuỗi các Vex tạo thành từ các đường thực thi của chương trình. Phương pháp đề xuất được thực nghiệm đánh giá bằng ba kịch bản khác nhau, trong đó có kịch bản phát hiện chéo trên các kiến trúc.

4.3.2 Thuật toán trích xuất đặc trưng CFDVex

Phương pháp trích chọn đặc trưng trên dòng điều khiển mức ngôn ngữ trung gian Vex (CFDVex) theo ý tưởng của thuật toán CFD, nhưng mỗi đỉnh của đồ thị dòng điều khiển của tập tin thực thi là khối cơ bản gồm dãy các đại diện Vex thay cho dãy các mã thực thi. Câu lệnh Vex có nhiều dạng khác nhau so với câu lệnh thực thi. Chúng có nhiều kiểu, mỗi kiểu có nhiều mẫu khác nhau, do đó chúng ta phải tìm cách chọn các đại diện của câu lệnh Vex. Chúng tôi đề xuất hai cách chọn đại diện cho câu lệnh Vex gọi là CFDVex mức 1 và mức 2. Với CFDVex mức 1, chúng tôi chọn kiểu của câu lệnh Vex làm đại diện cho câu lệnh đó. Với CFDVex mức 2, biểu thức chính của mỗi câu lệnh sẽ được lựa chọn như cột *Đại diện đề xuất* trong Bảng 4.1. Tương tự với CFD, độ phức tạp của thuật toán CFDVex là O(N^2).

Bảng 4.1: Kiểu câu lệnh Vex IR, mẫu các câu lệnh và đề xuất đại diện.

Kiểu câu lệnh	**Các mẫu câu**	**Đề xuất đại diện**
Put	Put(add) = tmp	Put
	Put(add) = constant	Put.cons
PutI	PutI(add) = tmp	PutI
	PutI(add) = constant	PutI.cons
WrTmp	tmp = GET (add)	Get
Store	STle (add) = tmp	STle

4.4 Công cụ giải nén phần sụn

Chúng tôi phát triển công cụ F-Toolkit cho mục đích này nhằm nâng cao khả năng dịch ngược, phân tích các phần sụn thành các chương trình sạch cho thực nghiệm. Quy trình dịch ngược sử dụng bộ công cụ F-Toolkit gồm bốn bước: thu thập phần sụn, giải nén phần sụn, tìm kiếm tệp tin phần sụn nhị phân và dịch ngược phần sụn.

4.5 Kết quả thực nghiệm

4.5.1 Thực nghiệm bóc tách phần sụn

Bộ công cụ F-Toolkit thử nghiệm với 13.674 phần sụn, chúng tôi đã thống kê được các định dạng tập tin hệ thống thông dụng trong thiết bị nhúng, đa phần là SquashFS. F-Toolkit đã dịch ngược thành công 6.623 tập tin nhị phân thành mã nguồn tường minh. Kết quả dịch ngược của F-Toolkit cũng đạt kết quả tốt hơn Fmk, Firmadyne hay Binwalk như trong Bảng 4.2.

Bảng 4.2: Kết quả thử nghiệm dịch ngược bản ảnh phần sụn của các hãng cung cấp thiết bị.

Firmware Mod Kit	18 %
Firmadyne	37 %
F-Toolkits	48 %

4.5.2 Thực nghiệm đánh giá phương pháp CFD

Chúng tôi thu thập tập mã độc IoT từ các nguồn IoTPoT, Detux, và VirrusShare. Sau khi thu thập, các file thực thi ELF được chọn và kiểm tra lại trên Virus Total. Mẫu sạch được thu thập từ hai nguồn: bản ảnh phần sụn và các máy tính cá nhân. Số lượng chi tiết các kiến trúc được đề cập trong Bảng 4.3. Kiến trúc MIPS là một trong những

Bảng 4.3: Thông tin thống kê về tập dữ liệu F-IoT.

	Virus -Share	**IoTPot**	**Detux**	**Shared**	**F -Mal**	**F -Benign**
MIPS	1.603	935	3.282	798	5.022	1.899
ARM	2.117	912	35	26	3.038	530
Intel 80386	5.492	570	29	5	6.086	1.438
X86-64	586	320	11	3	914	180

kiến trúc phổ biến nhất trong các kiến trúc nhúng và kiến trúc Intel 80386 là phổ biến trong PC. Vì vậy, tập dữ liệu thử nghiệm của chúng tôi tập trung vào các tập tin định dạng có thể thực thi trong hệ điều hành Linux trên MIPS và Intel. Các kết quả thử

nghiệm so sánh với phương pháp của Ding và phương pháp CFD về độ chính xác và F1-Score cho thấy CFD hiệu quả hơn tại tất cả các ngưỡng như Hình 4.4.

Bảng 4.4 so sánh giữa phương pháp của Ding và phương pháp CFD về độ phức tạp của thuật toán, tiêu thụ RAM, trung bình của các đường đi được hình thành và tỷ lệ trích xuất, kết quả thể hiện CFD vượt trội về tất cả các tiêu chí.

Bảng 4.4: So sánh giữa phương pháp cũ và mới.

	Các của Ding và cộng sự	**CFD**
Độ phức tạp thuật toán	NP-Hard	O(N^2)
Tiêu thụ RAM	6,0 GB	0,4 GB
Số đường trung bình được tìm thấy	10^4	10^{303}
Khả năng trích xuất được đặc trưng	86,2%	100%

4.5.3 Thực nghiệm đánh giá phương pháp CFDVex

Chuỗi Vex của khối cơ bản trong chương trình cũng được trích xuất bằng Angr bởi nó hỗ trợ cả Vex mức 1 và mức 2. Chi-square được sử dụng cho việc giảm chiều véc tơ đặc trưng với $K = 350$ và sử dụng phương pháp 2-gram để trích rút đặc trưng. Các độ đo sử dụng để đánh giá mô hình gồm ACC, FPR, FNR và F1-Score.

Phương pháp của Ding quá chậm để trích xuất tất cả mẫu MIPS trong tập F-IoT, so đó chỉ tập T1 là tập con của F-IoT được sử dụng. Tập này có 844 mẫu MIPS, gồm 300 mẫu sạch và 544 mẫu mã độc. Kết quả cho thấy, CFDVex cho kết quả ACC và F1-Score tốt hơn cách của Ding và cộng sự nhưng thấp hơn so với CFD dựa trên lệnh thực thi.

Khi huấn luyện bằng tập dữ liệu Intel 80386 và đánh giá bằng tập dữ liệu MIPS, tức học trên Intel và phát hiện mã độc cho MIPS, kết quả đạt độ chính xác khá cao (95,72%) với tỉ lệ dương tính giả FPR chấp nhận được (3,2%). Nhưng điều ngược lại không đúng, nếu huấn luyện bằng MIPS và phát hiện cho Intel thì kết quả phát hiện rất kém, mặc dù số lượng của tập MIPS lớn.

4.6 Tổng kết chương

Chương này đã trình bày hai phương pháp do chúng tôi đề xuất là trích xuất đặc trưng theo dòng điều khiển CFD và CFDVex. Hai phương pháp đề xuất được thiết kế để phát hiện hiệu quả mã độc nhúng, trên cơ sở thuật toán quy hoạch động, với độ phức tạp thuật toán là O(N^2). Thực nghiệm cho thấy các phương pháp đề xuất có tốc độ thực thi nhanh, độ chính xác cao, sử dụng ít bộ nhớ.

Kết quả nghiên cứu trên được công bố tại Tạp chí quốc tế [PP2], kỷ yếu hội thảo quốc tế [PP5, PP6].

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

IoT đang trở thành một thành phần quan trọng trong thế giới ngày nay. Phát hiện mã độc trên các thiết bị IoT sử dụng hệ điều hành Linux có vai trò quan trọng, đảm bảo sự phát triển bền vững và an toàn. Tuy nhiên, đây là bài toán khó do có nhiều thách thức như môi trường đóng gói, đặc thù về hạn chế tài nguyên phần cứng nhưng lại rất phong phú về chủng loại, các công cụ hỗ trợ và tập dữ liệu. Từ những thách thức trên yêu cầu phải có cách tiếp cận mới để giải quyết bài toán phát hiện mã độc trên thiết bị IoT nói chung và thiết bị IoT sử dụng hệ điều hành Linux nói riêng.

Chúng tôi trình bày mô hình tổng thể để giải quyết các tình huống liên quan đến mã độc trên thiết bị IoT, xử lý cả các mã độc được cài cắm và lây lan trong quá trình hoạt động, áp dụng cả kỹ thuật phân tích tĩnh và phân tích động trong giải quyết bài toán phát hiện và phân lớp mã độc trên từng kiến trúc vi xử lý IoT đến phát hiện mã độc đa kiến trúc, phát hiện mã độc chéo kiến trúc. Song song với việc nghiên cứu đề xuất mô hình, thuật toán mới, chúng tôi tập trung xây dựng bộ công cụ F-Toolkit và tập dữ liệu F-IoT phục vụ thực nghiệm và các nghiên cứu lâu dài.

5.1 Các đóng góp của chúng tôi

Sau một thời gian nghiên cứu và giải quyết bài toán phát hiện mã độc trên thiết bị IoT sử dụng hệ điều hành Linux, chúng tôi đã có một số những đóng góp chính như sau:

Thứ nhất, chúng tôi xây dựng tập dữ liệu F-IoT, là tập dữ liệu mã độc IoT lớn nhất thời điểm hiện tại, phục vụ trực tiếp cho các đánh giá thực nghiệm của luận án cũng như đóng góp quan trọng cho các nghiên cứu sau này về IoT.

Thứ hai, chúng tôi xây dựng F-Sandbox là sandbox chuyên dụng cho các thiết bị IoT dựa trên hệ điều hành Linux nhúng. F-Sandbox có khả năng mô phỏng đầy đủ các thành phần và thông tin của thiết bị định tuyến, tạo nhiều phiên bản khác nhau tương ứng với môi trường đa dạng của thiết bị IoT, giúp thực thi và giám sát hiệu quả các mã độc. Thực nghiệm chứng minh, dữ liệu thu được từ F-Sandbox có khả năng phát hiện mã độc với độ chính xác cao.

Thứ ba, đề xuất hai thuật toán trích chọn đặc trưng mới để phát hiện mã độc trên thiết bị IoT. Hai thuật toán này trích xuất đặc trưng theo dòng điều khiển (Control

flow-based feature) dựa trên mã thực thi và ngôn ngữ trung gian Vex. Thuật toán CFD dựa trên opcode để phát hiện mã độc trên từng kiến trúc, còn CFDVex dựa trên ngôn ngữ trung gian Vex để phát hiện mã độc đa nền tảng. Thực nghiệm chứng minh, hai thuật toán trên có tốc độ xử lý nhanh, độ phức tạp chỉ $O(N^2)$, với N là số khối cơ bản của tệp tin thực thi khi được dịch ngược. CFDVex cho phép phát hiện mã độc trên kiến trúc mới từ mô hình được huấn luyện trên kiến trúc cũ.

Các nghiên cứu của chúng tôi nhằm xây dựng một phương pháp hoàn chỉnh gồm các phương pháp động và phương pháp tĩnh để phát hiện mã độc trong các thiết bị định tuyến nói riêng và thiết bị IoT sử dụng hệ điều hành Linux nhúng nói chung. Các thực nghiệm cho kết quả tốt và mở ra nhiều hướng phát triển tiếp theo. Các kết quả của chúng tôi đã công bố trong các công trình khoa học được đăng tải trên các hội nghị, tạp chí chuyên ngành trong nước và quốc tế có phản biện.

5.2 Hướng phát triển

Một số hướng phát triển tiếp theo của chúng tôi:

Thứ nhất, đối với F-Sandbox, vẫn còn một số mẫu mã độc chưa được kích hoạt do thiếu môi trường hoặc phát hiện cơ chế giám sát. Nghiên cứu tiếp theo sẽ hướng tới từng kỹ thuật cụ thể của mã độc để cải tiến môi trường thực thi. F-Sandbox có khả năng mở rộng cho đa kiến trúc, hiện tại mới cài đặt triển khai cho môi trường MIPS, sẽ tiếp tục hoàn thiện cho các môi trường khác.

Thứ hai, các phần sụn nghiên cứu trong luận án đang tập trung đa số là các bản ảnh dạng đầy đủ. Với các bản ảnh không đầy đủ, hay bản cập nhật của phần sụn cần có nghiên cứu thử nghiệm, đánh giá và đề xuất phương pháp bóc tách, đánh giá hợp lý. Đây là mảng nghiên cứu mới chưa được đề cập nhiều.

Thứ ba, phát hiện mã độc đa kiến trúc là đặc trưng của mã độc trên thiết bị IoT, do đó chúng tôi sẽ tiếp tục cải tiến phương pháp trích chọn đặc trưng đa nền tảng để nâng cao khả năng phát hiện.

Thứ tư, pháp triển các phương pháp loại bỏ mã độc khỏi phần sụn của thiết bị IoT. Mã độc nhúng tồn tại dưới hình thức lây lan trong quá trình hoạt động là mã độc không thường trú nên việc loại bỏ đơn giản là khởi động lại thiết bị, tất nhiên trong nhiều trường hợp không cho phép khởi động lại. Các mã độc cài cắm trong thiết bị sau khi bị loại bỏ cần có phương pháp đảm bảo hệ thống hoạt động ổn định. Đây là những nội dung nghiên cứu sẽ tiếp tục được hoàn thiện trong thời gian tiếp theo.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN ÁN

(PP1). Tran Nghi Phu, Nguyen Ngoc Binh, Hoang Dag Kien, Ngo Quoc Dung, Nguyen Dai Tho. *A Novel Framework to Classify Malware in MIPS Architecture-based IoT Devices*. Security and Communication Networks. 2019 (SCIE index, Q2, IF = 1,32) (Accepted).

(PP2). Tran Nghi Phu, Ngo Quoc Dung, Le Van Hoang, Nguyen Dai Tho, Nguyen Ngoc Binh. *A System Emulation for Malware Detection in Routers*. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue-11, Septemper 2019, pp. 32-40 (Scopus index)

(PP3). Trần Nghi Phú, Ngô Quốc Dũng, Nguyễn Đại Thọ, Nguyễn Ngọc Bình, and Hoàng Đăng Kiên. *Phát Hiện Mã Độc Trên Các Thiết Bị IoT Dựa Trên Lời Gọi Syscall và Phân Loại Một Lớp SVM.* Tạp chí Thông tin và Truyền thông, ISSN 1859-3550, 2018, pp. 150-158.

(PP4). Tran Nghi Phu, Nguyen Ngoc Binh, Ngo Quoc Dung, and Le Van Hoang. *Towards Malware Detection in Routers with C500-Toolkit.*. In 2017 5th International Conference on Information and Communication Technology (ICoIC7), pp. 1-5, 2017. https://doi.org/10.1109/ICoICT.2017.8074691. (Scopus index)

(PP5). Tran Nghi Phu, Nguyen Ngoc Toan, Le Hoang, Nguyen Dai Tho, and Nguyen Ngoc Binh. *C500-CFG: A Novel Algorithm to Extract Control Flow-based Features for IoT Malware Detection.*19th International Symposium on Communications and Information Technologies (ISCIT), 2019, Hochiminh, Vietnam, pp. 568-573.

(PP6). Tran Nghi Phu, Nguyen Ngoc Binh, Nguyen Dai Tho, Nguyen Ngoc Toan, and Le Huy Hoang. *CFDVex: A Novel Feature Extraction Method for Detecting Cross-Architecture IoT Malware*. The tenth international Symposium on Information and Communication Technology (SoICT 2019), Dec-2019, Hanoi, Vietnam. (Accepted, Scopus index)

(PP7). Trần Nghi Phú, Ngô Quốc Dũng, Nguyễn Huy Trung, Nguyễn Ngọc Bình. *Mô Hình Phát Hiện Mã Độc Trong Phần Mềm Nhúng Trên Thiết Bị Định Tuyến.* Hội Nghị Khoa Học Hội Thảo Quốc Gia Lần Thứ XIX: Một Số Vấn Đề Chọn Lọc của CNTT&TT - Hà Nội, 2016, pp. 206-212.

(PP8). Trần Nghi Phú, Nguyễn Huy Trung, Ngô Quốc Dũng, Nguyễn Ngọc Bình, and Nguyễn Đại Thọ. *Phát Triển Công Cụ Dịch Ngược Firmware Trên Thiết Bị Định Tuyến.* Hội Nghị Khoa Học Hội Thảo Lần Thứ I: Một Số Vấn Đề Chọn Lọc về an Toàn Thông Tin, 9-2016, pp. 108-115.