卒 業 論 文

IoT 機器を想定した実行環境における マルウェアの影響評価

101910020　　大羽 俊輔

名古屋大学 情報学部
コンピュータ科学科 情報システム系
2023 年 2 月

概要

IoT 機器の増加に伴い、マルウェアによる IoT 機器への攻撃リスクが高まっている。IoT とは、実世界のモノをインターネットに接続し、インターネットを介して情報を相互にやり取りするという考え方である。マルウェアはコンピュータの普及に伴い盛んに研究されてきた分野であるが、IoT 機器を攻撃対象としたマルウェアが盛んに研究されるようになったのは、IoT 機器が普及してきた 2018 年頃のことである。パーソナルコンピュータの多くが x86_64 プロセッサで動作している一方で、IoT 機器の多くは ARM プロセッサで動作している。従来のマルウェアの解析・検出手法の多くは x86_64 アーキテクチャに特化しており、命令セットが異なる ARM アーキテクチャに対して容易に転用できない。そのため、ARM アーキテクチャを攻撃対象としたマルウェアの更なる研究が求められている。昨今、IoT 機器の実行環境としてコンテナやユニカーネルなどの環境が提案されており、実行環境は多様化してきている。特に、コンテナは近年注目されている環境である。

本研究では、IoT 機器の実環境を想定した環境においてマルウェアがどのような影響を与えるのかを定量的に示すことを目的として、評価手法の提案とその手法を用いた評価実験を行う。アプリケーション実行環境として、ラズビアン環境、コンテナ環境、ユニカーネル環境を対象に、それぞれの環境においてマルウェアを実行した際に実行環境やサービスにどのような影響を与えるのかを評価する。いずれの環境も、IoT 機器の機能としてウェブサーバを想定し、ウェブサーバアプリケーションをインストールする。評価指標として、マルウェアの実行可否および脅威スコアを用い、マルウェアの影響を定量的に推定する。実行の可否は、マルウェアが実行可能かどうかを示す指標であり、マルウェアを実行した結果が正常終了または一定時間実行状態であったものを実行可能、異常終了したものを実行不可と判定する。脅威スコアは、実行したマルウェアがどの程度脅威となるかを示す値であり、マルウェアの実行の可否とそのマルウェアの分類に応じて算出される。これらの指標を用いて各環境におけるマルウェアの影響を比較することで、3 つの環境の特性を示すことができる。ラズビアン環境の評価環境として、エミュレータ型仮想化ソフトウェアである QEMU を用いて Raspberry Pi OS を動作させ、マルウェアの実行、実行結果の取得、脅威スコア算出を自動化する環境を構築した。

評価実験の結果、実行の可否および脅威スコアを求めることに成功し、提案手法を用い

ることでマルウェアの影響を定量的に示すことが出来ることが分かった。加えて、実行に失敗したマルウェアの終了ステータスを分析することで、実行に失敗するマルウェアの多くが不正なメモリ参照または存在しないコマンドの参照により実行に失敗していることが明らかになり、使用可能なコマンドを制限することでマルウェアの影響を抑制できることを示唆する結果となった。自動化を行うことで、1 つのマルウェアにつき 180 秒程度掛かっていた評価を 100 秒程度で行うことができ、提案手法の有用性が示された。

本研究では、IoT 機器の実環境を想定した 3 つの実行環境においてマルウェアがどのような影響を与えるのかを定量的に示す手法を提案し、ラズビアン環境において実際に評価実験を行うことでその有効性を示した。今後の課題として、残る 2 つの環境でも評価実験を行うことや、脅威スコアを厳密に算出するために実際のマルウェアの振る舞いから脅威の程度を評価すること、評価にかかる時間をさらに短縮することなどが挙げられる。

目次

図目次

表目次

第 1 章

はじめに

1.1　研究背景

1.1.1　昨今のマルウェア情勢

近年、マルウェアによる被害が増加している。特に、攻撃対象のシステムを暗号化し、復号と引き換えに金銭の支払いを求めてくるランサムウェアと呼ばれるマルウェアの活動が活発化している。ランサムウェアに感染すると、事業停止に陥ったり、機密情報が流出するなどの被害を被る可能性がある。警視庁の公開する「令和 4 年上半期におけるサイバー空間をめぐる脅威の情勢等について」[1] によると、国内におけるランサムウェアによる被害の報告件数は 2020 年から年々増加しており、2022 年には 2020 年のおよそ 5 倍の 114 件の被害が報告されている。また、情報処理推進機構（IPA）が公開している「情報セキュリティ 10 大脅威 2022」[2] では、組織部門において 2021 年に引き続きランサムウェアが脅威の 1 位として位置付けられており、現在のサイバー空間における大きな脅威となっている。

IoT 機器においてもマルウェアの被害は報告されている。IoT とは、実世界のモノをインターネットに接続し、インターネットを介して情報を相互にやり取りするという概念である。IoT 機器におけるマルウェアによる被害の多くは、ボット型マルウェアによるものであり、感染すると DDoS 攻撃の踏み台に利用されるなどの被害が生じる。特に、2016 年頃に大規模な DDoS 攻撃を行ったことで知られるボット Mirai は、そのソースコードが公開されて以降、その亜種が出現し続けており IoT 機器にとって大きな脅威となっている [3]。IoT 機器がマルウェアに感染する原因の多くが、初期パスワードを変更していなかったりシステムの更新を行なっていないためだと言われている。情報通信研究機構（NICT）

では、感染リスクのある機器を調査し利用者への注意喚起を行う取組、NOTICE[4] を通じてマルウェアに対する対策を講じている。

1.1.2　マルウェア研究における研究対象の偏り

現在、既存のマルウェアに関する研究の多くが、Windows を研究対象としており、Linux といったその他の OS を研究対象にしている研究は比較的少ない [3]。Windows を研究対象とする研究が多い背景としては、Windows は他の OS に比べて利用ユーザが多く、それを狙うマルウェアが多く存在することから、積極的に研究が行われているためであると考えられる。近年、IoT 機器を狙ったマルウェアの脅威の拡大に伴い、Linux などの OS を研究対象とする研究も報告されてきているが、Windows を対象とする研究に比べて十分な議論はなされていない。

既存のマルウェアに関する研究の中で、ARM アーキテクチャ上で動作するマルウェアを研究対象とした研究は少ない。パーソナルコンピュータでは一般的に x86_64 プロセッサが用いられるのに対し、IoT 機器などの多くの組み込み機器では ARM プロセッサが用いられることが一般的である [3]。IoT 機器が普及する以前、マルウェアの多くはパーソナルコンピュータを攻撃対象としていたため x86_64 アーキテクチャを対象とした研究が盛んに行われたが、ARM アーキテクチャを対象とした研究が行われるようになったのは IoT 機器を狙うマルウェアが流行してきた比較的最近のことである [3]。

マルウェアの研究において、アーキテクチャは重要な要素である。一般的にマルウェアの検出手法は特定のアーキテクチャに特化して提案されているため、既存の検出手法を異なるアーキテクチャに転用することは困難とされる。例えば、x86_64 アーキテクチャを対象にしたマルウェアの研究で提案された検出手法は、ARM アーキテクチャを対象にしたマルウェアの検出には転用できない。こうした背景から、ARM アーキテクチャを対象としたマルウェアの研究が急務となっている。

1.2　研究目的と概要

本研究は、ARM プロセッサの IoT 機器を想定した 3 つの実行環境において、マルウェアが与える影響を定量的に評価し、環境ごとの影響を比較することを目的とする。はじめに、影響評価を行う上での評価対象と評価指標を定め、評価実験のための環境構築を行なった。続いて、評価対象のうちラズビアン環境を用いて定量的なマルウェアの影響評価を行った。

本研究の貢献は、以下の通りである。

- ARM プロセッサの IoT 機器を想定した実行環境においてマルウェアが与える影響を定量的に示す手法を提案した
- 提案手法を用いた評価実験を行い、その有効性を示した

1.3　本論文の構成

本論文は、本章を含めて全 5 章で構成されている。2 章では、関連研究として IoT マルウェアの検出・分類手法と V-Sandbox について述べる。3 章では、評価を行う上での評価対象と評価指標を提案し、最後に環境構築について述べる。4 章では、実験概要、実験結果、考察について述べる。5 章では、本研究の結論とまとめ、今後の課題について述べる。

第 2 章

関連研究

2.1　IoT マルウェアの検出・分類手法

IoT 機器を攻撃対象としたマルウェアの検出・分類手法については、既に多くの議論が行われている。以降、IoT 機器を攻撃対象としたマルウェアを IoT マルウェアと呼ぶこととする。IoT マルウェアに限らず、一般にマルウェアの解析には静的解析と動的解析の二つのアプローチが存在する [3]。静的解析は、マルウェアのバイナリファイルから得られる構造的、意味的情報をプログラムを実行することなく分析する手法である。この手法では、アンチアセンブル、コード難読化などの解析回避手法の影響を受けやすいとされている。一方、動的解析は、マルウェアのプログラムを実行し、その挙動を観察したりデバッグを行うことで分析を行う手法である。この手法ではアンチデバッグや遅延実行の影響を受けやすいとされている。マルウェアを実行する際は一般的に、隔離された実行環境であるサンドボックス上で実行し、その挙動を分析する。本研究においても、サンドボックスを構築し動的解析による解析を行った。

IoT マルウェアの検出・分類手法として従来から機械学習が活用されており、学習のための様々な特徴量 [3] が提案されている。その中から、三つの特徴量について取り上げる。

2.1.1　オペコード

マルウェアの検出・分類を行う上で有効な特徴量として、オペコードが挙げられる。オペコードとはプロセッサが実行可能な命令のことであり、レジスタやオペランドを引数に取り、目的の動作を実行する。オペコードはマルウェアの検出・分類に有効なことが Hamed らによって示されている [5]。しかしながら、オペコードは命令セット・アーキテ

クチャに依存しているため、比較的汎用性に欠ける特徴量であると思われる。

2.1.2　文字列データ

プログラムのバイナリデータの中には、文字列として印字可能な文字列が含まれている場合があり、特徴量として用いることが可能である。例えば、IP アドレスや DLL 名、エラーメッセージやコメントなどである。

2.1.3　バイト列

バイト列はバイナリデータをバイト単位で逐次的に表現した数値列である。Nguyen らによって Linux において、いくつかのモデルが検出・分類に有効であることが示されている [6]。一次元のバイト列を表現の形を変えて 2 次元配列にし、グレースケール画像として学習する手法なども提案されている [3]。

2.2　V-Sandbox

サンドボックスとはマルウェアや不正と思われるプログラムを実行し動的解析するための隔離された実行環境のことである。サンドボックスは、マルウェアの影響の評価やプログラムの安全性を評価する際に用いられる。Le らは、IoT マルウェアを解析するためのサンドボックス V-Sandbox を提案している [7]。彼らが提案したサンドボックスでは一般的な IoT マルウェアの特徴として見られる C&C サーバとの通信や脆弱な端末の探索を監視することや動作に必要な共有ライブラリの動的な追加が可能となっている。C&C サーバとは、ボット型のマルウェアに感染した機器に対して攻撃命令を出すためのサーバのことである。また、V-Sandbox では複数の CPU アーキテクチャに対応した実行環境の構築が可能となっている。以降、V-Sandbox で提案されている手法の詳細を述べる。

V-Sandbox ではまず、実行対象のマルウェアのプログラムを readelf や ldd といった解析ツールを用いて解析し、ファイルタイプや動作に必要なライブラリ、マシンの情報などを取得する。得られた情報を用いて仮想環境の設定ファイルを動的に生成し、サンドボックス及び C&C サーバを QEMU を用いて構築する。マルウェアの挙動解析では、システムコール呼び出し、ファイル操作、ネットワークトラフィック、パフォーマンスの観点について情報を記録し、レポートを自動生成することが可能である。

V-Sandbox の評価実験では、従来から提案されている LiSa サンドボックスに比べて実行可能なマルウェアの数が多いことが明らかになり、サンドボックスとしての有効性を示

す結果が得られた。実行可能なマルウェアの数が従来のサンドボックスに比べて増加したのは、V-Sandbox が複数のアーキテクチャに対応していることや共有ライブラリを動的に追加していること、C&C サーバをシミュレートしていることに起因すると考えられる。また、LiSa サンドボックスや Cuckoo サンドボックスに比べ、ネットワークトラフィックやシステムコールに関する情報を多く取得していることが分かった。

Le らは、V-Sandbox の今後の課題として、対応するアーキテクチャを更に追加することや共有オブジェクトを追加することを挙げている。

第 3 章

提案手法

3.1　概要

本章では、ARM プロセッサで動作する IoT 機器を想定した実行環境において、定量的なマルウェアの影響評価を行うための手法と評価実験を行うための環境構築について述べる。評価対象とする実行環境は、ラズビアン環境、コンテナ環境、ユニカーネル環境の 3 つであり、いずれの環境も IoT 機器の機能としてウェブサーバを想定してウェブサーバアプリケーションをインストールする。これらの実行環境上でマルウェアのサンプルを一つずつ実行し、その振る舞いを実行の可否および脅威スコアの観点から評価する。脅威スコアは、マルウェアが与える脅威の程度を定量的に表した値のことである。

提案した手法を用いて評価実験を行うにあたり、実験環境の 1 つであるラズビアン環境について環境構築を行なった。今後、マルウェアの動的解析を行うことを想定してエミュレート型仮想化ソフトウェアである QEMU を用いて環境の構築を行なった。また、マルウェアの実行、及び実行の可否と脅威スコアの算出をプログラムにより自動化し、効率的に評価実験が行える環境を構築した。

3.2　評価対象

本研究では、ARM プロセッサで動作する IoT 機器を想定した 3 つの実行環境を評価対象とする。いずれも IoT 機器の機能としてウェブサーバを想定し、ウェブサーバアプリケーションをインストールする。ウェブサーバとしての機能を兼ね備えてた IoT 機器としては、一般的な市販ルータなどが挙げられる。また、いずれの環境も ARM プロセッサで動作させることを前提とする。

評価対象の 1 つ目が、ラズビアン環境である。ラズビアン環境では、使用する OS として Raspberry Pi OS を使用する。IoT 機器の多くがその OS として Linux を採用しており、その中でも Raspberry Pi OS は多くのシェアを持っている [3]。こうした背景から実行環境として Raspberry Pi OS を採用した。

評価対象の 2 つ目がコンテナ環境である。コンテナ環境では、ウェブサーバがコンテナ環境上で動作していることを想定し、コンテナ型仮想化ソフトウェアとして Docker を使用する。また、使用するイメージは、ウェブサーバの動作に必要な最低限のパッケージを含んだものを使用する。

評価対象の 3 つ目がユニカーネル環境である。ユニカーネルとは、OS の一種であり、特定のアプリケーションの実行に必要な最低限のライブラリや機能を搭載した軽量 OS のことである。ユニカーネル環境では、ウェブサーバがユニカーネル上で動作していることを想定する。

以上の 3 つの実行環境を評価し、比較することにより、実行環境の差異がマルウェアの与える影響にどのような変化をもたらすのか、何が変化をもたらすのかを検討することができる。

3.3 評価指標

影響を定量的に評価する上で 2 つの指標を設定する。1 つが、”マルウェアの実行の可否”である。マルウェアの実行の可否は、マルウェアが実行されたかどうかを測る指標である。もう 1 つが脅威スコアである。脅威スコアはそのマルウェアが実行環境やサービス、ユーザなどにどの程度脅威を与えたかを測る指標である。以降、各指標について詳しく述べる。

3.3.1 マルウェアの実行の可否

マルウェアの影響を評価する上で重要な視点として、マルウェアのプログラムが正常に動作したかという点が挙げられる。マルウェアがプログラムの実行に失敗した場合、そのマルウェアがもたらす脅威は限定的、あるいは無いものと考えられる。また、プログラムが正常に動作するには適切なライブラリや権限、環境変数、その他条件が揃う必要があることから、マルウェアのプログラムが実行に成功するかどうかは実行環境に大きく依存すると考えられ、環境ごとの影響の差異を比較する上でも重要な項目となる。そこで、本研究では”マルウェアの実行の可否”を評価指標として設定する。

マルウェアの実行の可否では、各マルウェアを実行可能、もしくは実行不可の 2 種類に分類する。分類を行う際のフローチャートを図 3.1 に示す。まず、マルウェアを実行し、ファイルが破損しているなどの理由で実行に失敗するプログラムについては実行不可として分類する。実行できたもののうち、異常終了したものについても実行不可として分類する。正常終了したものについては実行可能とし、一定時間処理が継続しているものについてもプログラムが正常に動作しているとみなして実行可能と分類する。

3.3.2 脅威スコア

本研究では、実行したマルウェアがもたらす脅威の程度を”脅威スコア”として定量化する。尚、本来であればマルウェアの脅威の程度を計測する場合、システムコール呼び出しやパケットの監視などを行い、実際にどのような脅威が発生したかを観測した上で脅威の程度を定量化するのが自然であが、本研究では著者の技術不足のためマルウェアの実際の振る舞いについては観測せず、マルウェアの分類に従った希望的観測による脅威の定量化を行った。

脅威スコアは次のように導出する。まず、”マルウェアの実行の可否”において実行可能であったマルウェアについては、そのマルウェアの分類に応じた相応の脅威が発生したと仮定し、分類に応じた脅威スコアを設定する。マルウェアの分類はインターネット上で提供されているウイルス解析サービスを利用して判定する。分類に応じた脅威スコアを表 3.1 に示す。表 3.1 では、一般に脅威が大きいとされるランサムウェアやバックドアなどは脅威スコアが大きく設定されており、アドウェアやコインマイナーといった比較的脅威が小さいと考えられるものについては脅威スコアが小さく設定されている。表 3.1 を作成するにあたり、セキュリティサービスを提供している企業 netscope が公開しているマルウェアの脅威の分類表 [8] を参考にした。続いて、”マルウェアの実行の可否”において実行不可であったマルウェアについては、そのマルウェアの脅威は発生しなかったと仮定し、脅威スコアは 0 と判断する。

3.4 環境構築

ラズビアン環境を用いた評価実験を行うにあたり、実験環境の構築と実行の可否及び脅威スコア導出の自動化を行った。

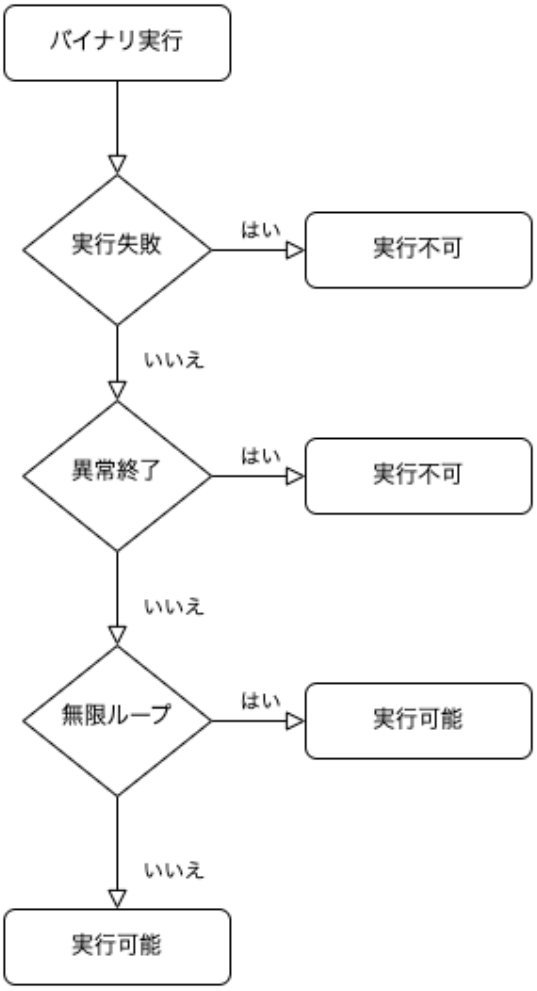

図 3.1: 実行の可否を判定するフローチャート

表 **3.1:** マルウェアの分類と脅威スコアの対応表

脅威スコア	分類
3	Ransomware Trojans Viruses Downloaders Backdoors Rootkits Exploits Password Stealers
2	Spyware
1	Bundlers Coinminers Adware Dialers Hoaxes Hacktools Keygens Jokes

3.4.1 ラズビアン環境の構築

ラズビアン環境の構築と評価の自動化は図 3.2 のような構成で行った。まず、Ubuntu 上で QEMU を動作させ Raspberry Pi OS をエミュレートした。QEMU とは、オープンソースのエミュレータ型仮想化ソフトウェアであり、様々なアーキテクチャをエミューレートすることができる。QEMU はマルウェアの動的解析を行う環境として提案されており、動的解析のためのプラグイン [9] も多く提供されている。本研究では今後、マルウェアの動的解析を行うことを想定して QEMU を採用した。以降、エミュレートした Raspberry Pi OS 環境をゲスト環境と言う。続いて、ゲスト環境にウェブサーバアプリケーションとして、Nginx をインストールした。Nginx は、オープンソースのウェブサーバアプリケーションであり、世界で広く利用されているウェブサーバである。次に、マルウェアのサンプルをゲスト環境に転送することや評価を自動化することを考慮して、ホスト環境からゲスト環境に SSH 接続できるように QEMU 及び Raspberry Pi OS の設定を行った。

3.4.2 評価の自動化

評価実験では、数百個のサンプルを評価することを想定しており、手動で評価を行うには時間が掛かる。そのため、ゲスト環境の立ち上げ、マルウェアの転送、実行、評価を自

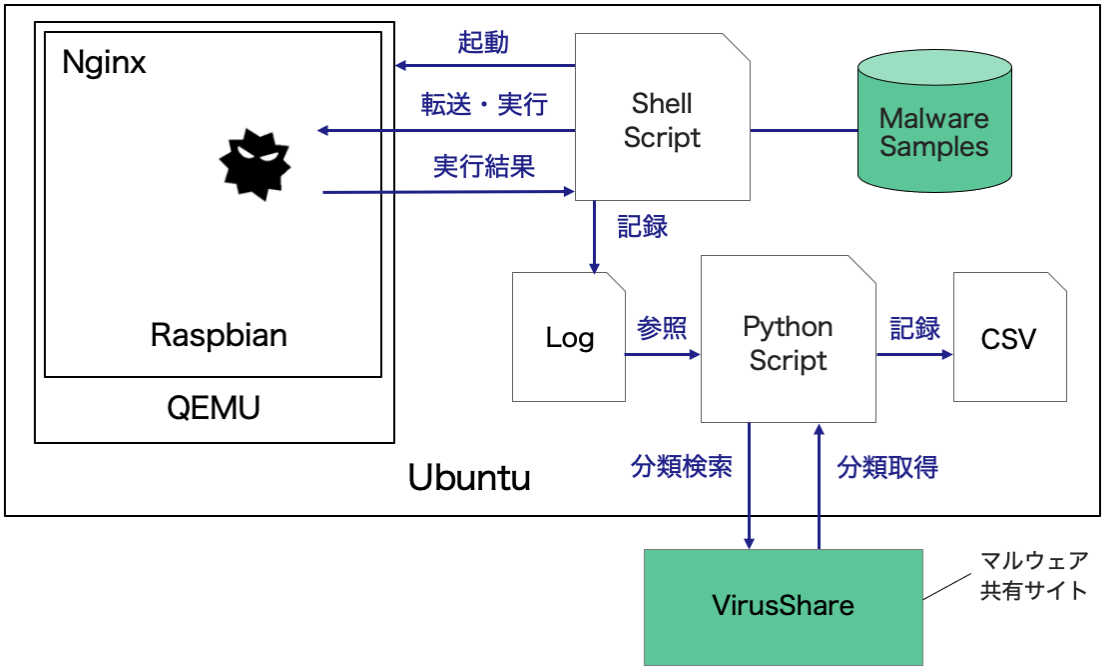

図 3.2: ラズビアン環境と自動化の構成図

動化した。自動化の構成図は図 3.2 の通りである。まず、サンプルごとの実行環境を同じにするため、サンプルごとに新しいゲスト環境を立ち上げる必要がある。ゲスト環境を立ち上げた後、マルウェアを SCP コマンドで転送し、SSH コマンドを用いてマルウェアを実行する。SSH コマンドの終了ステータスからプログラムが正常終了したかどうかなどを判定することができ、終了ステータスが 0 または 124 であれば実行可能、終了ステータスがそれ以外であれば実行不可として”マルウェアの実行の可否”を求める。終了ステータス 0 はプログラムが正常に終了したことを示しており、124 はプログラムがタイムアウトしたことを示す。次に、マルウェアの脅威スコアを求める。マルウェアの実行の可否において実行不可と判定されたものについては脅威スコアは 0 とし、実行可能と判定されたものについては、インターネット上で公開されているマルウェアの検体データベースである VirusShare でそのマルウェアがどの分類に分類されるかを求め、表 3.1 の分類と脅威スコアの対応表に基づいて脅威スコアを設定する。VirusShare では、複数のマルウェア解析サービスの解析結果を取得することができる。その解析結果から文字列のパターンマッチングにより最も合致数の多い分類をそのマルウェアの分類とした。最後に、求めた脅威スコアは CSV 形式で出力した。マルウェアの実行の可否を元に脅威スコアを求める工程

は Python プログラムを用い、その他については BashShell スクリプトを用いて自動化を行った。

第 4 章

評価

4.1　実験概要

本節では、評価実験の概要について述べる。本研究では評価実験として、3.4 章で構築したラズビアン環境を用いてマルウェアの実行の可否および脅威スコアを求める実験を行った。本実験の実験環境を表 4.1 に示す。評価に使用するサンプルとして、VirusShare で管理されているマルウェアのうち ARM アーキテクチャ上で動作するマルウェアのみをまとめた IoT_ARM リポジトリ [10] のデータセットを用いた。評価を行う上で、実行可能な形式でないサンプルについては除外した。実験に使用したサンプルの分類内訳を表 4.2 に示す。

表 4.1: 実験環境

	OS	CPU	Memory	Architecture
Host machine	Ubuntu 22.04.1 LTS（64bit）	Intel Core i7-10750H 2.60GHz*12	16GB	x86_64
Sandbox machine	Raspbian Buster 2020.02.14（32bit）	QEMU	256MB	ARM

表 4.2: サンプルの分類内訳

分類	サンプル数
Trojans	442
Backdoor	338
Exploits	18
Hacktools	1
Unknown	2
その他	0
計	801

また、マルウェアの実行の可否を求める際に無限ループと判定する時間の長さは 10 秒とした。

4.2 実験結果

801 個のサンプルについての脅威スコアの評価結果を表 4.3 に示す。脅威スコアは合計が 1212、平均が 1.513、分散が $4.99 \times 10^{-}3$ という結果が得られた。

表 4.3: 脅威スコアの結果

脅威スコア	1212
平均	1.513
分散	$4.99 \times 10^{-}3$

次に、マルウェアの分類別の実行の可否を表 4.4 に示す。表 4.4 に示す通り、サンプルのおよそ半数である 404 個について実行可能となった。分類別に見ると、Trojans は約 60%、Backdoor は約 30% が実行不可という結果となった。また、Trojans と Backdoor 以外の分類については全てのサンプルが実行不可となった。

表 4.4: マルウェアの分類別の実行の可否

	サンプル数	実行可能	実行不可
Trojans	442	176(39.8%)	266(60.1%)
Backdoor	338	228(67.4%)	110(24.8%)
Exploits	18	0	18
Hacktools	1	0	1
Unknown	2	0	2
その他	0	0	0
計	801	404(50.4%)	397(49.6%)

次に、異常終了時の終了ステータスとその内訳を表 4.5 に示す。表 4.5 に示す通り、Trojans と Backdoor、Exploits のいずれにおいても、9 割以上が終了ステータス 139 または 127 で終了している。終了ステータス 139 は、プログラムが不正なメモリアドレスを参照した際に出力されるステータスである。終了ステータス 127 は、不明なコマンドを実行した際に出力されるステータスである。Exploits に関しては、特に終了ステータス 127 で異常終了する割合が高く、異常終了した Exploits のサンプルのうち約 89% が終了ステータス 127 で終了している。

マルウェアの評価時間について、手動で行なった場合はマルウェア 1 つあたり 180 秒程度掛かったのに対し、評価の自動化を行なった場合は 100 秒程度掛かる結果となった。

表 **4.5:** 異常終了時の終了ステータスとその内訳

Trojans	Backdoor	Exploits	Hacktools
139(157)	139(64)	127(16)	127(1)
127(84)	127(42)	139(1)	
135(21)	132(2)	1(1)	
2(3)	135(1)		
132(1)	2(1)		

139: 不正なメモリ参照　127: 不明なコマンドの実行

4.3　考察

本節では、実験結果に対する考察を述べる。

本実験を通して、マルウェア 1 つあたりに掛かる評価時間が手動の場合に 180 秒程度掛かっていたところを、評価の自動化により 100 秒程度まで短縮することが可能であることが示された。このことから、より効率的に評価を行えるという点で、節 3.4 で提案した自動化手法の有用性が示されたと言えるだろう。

表 4.3、および表 4.4 に示す通り、提案手法により脅威スコアおよび実行の可否を求められることが明らかになった。これら 2 つの指標によりマルウェアの影響を定量的に示すことができるため、マルウェアの影響を定量的に評価する手法として提案手法の有効性が示されたと言えるだろう。

異常終了時の終了ステータスについて表 4.4 に示す通り、Trojans と Backdoor、Exploits のいずれにおいても、9 割以上が終了ステータス 139 または 127 で終了している。終了ステータス 139 は不正なメモリアクセスが発生した場合に出力される終了ステータスであり、終了ステータス 127 は不明なコマンドを実行しようとした際に出力される終了ステータスである。終了ステータス 139 が発生した原因として、実行環境がマルウェアの想定している環境と異なっていたことが考えられる。具体的には、OS やカーネルがマルウェアの想定するものとは異なっており、想定していたメモリ管理が行われずメモリアクセス違反が発生したと考えられる。終了ステータス 127 が発生した原因としては、マルウェアがプログラム中で使用しているコマンドが実行環境に存在しなかったためであると考えられる。マルウェアの中にはプログラム中で特定のツールのコマンドを呼び出しているものがあるが、そのツールが実行環境に存在しない場合には不明なコマンドの実行となるため終了ステータス 127 を出力して異常終了することになる。この時、マルウェアが呼び出すコマンドを使用する権限を持っていなかった場合は終了ステータス 126 を出力して異常終了することになる。以上のことから、実行環境で利用可能なコマンドを限定する、または

コマンドの実行権限を制限することでマルウェアの脅威を抑制することができると推測される。

第 5 章

おわりに

5.1　結論とまとめ

本研究では、ARM プロセッサの IoT 機器を想定した 3 つの実行環境において、マルウェアが与える影響を定量的に評価する手法を提案し、ラズビアン環境について実際に評価実験を行い、その結果を考察した。本研究の調査から、本研究で提案した評価手法がマルウェアの影響を定量的に評価する手法として有効であることが明らかとなった。また、マルウェアの脅威を抑制するための手段として、利用可能なコマンドを限定する、もしくは権限を制限することが有効であることが示唆された。

5.2　今後の課題

今後の課題として、以下の 4 点が挙げられる。

- 脅威スコアを希望的観測ではなくマルウェアの振る舞いに基づいて算出すること
- コンテナ環境及びユニカーネル環境についても評価実験を行うこと
- 各環境における影響評価を比較し環境の特性を考察すること
- 評価にかかる時間をさらに短縮すること

1 つ目の課題について述べる。本研究で提案した手法では、脅威スコアをマルウェアの実際の振る舞いではなく分類に基づいて算出した。しかし、この算出方法は希望的観測によるものであり実際の振る舞いを反映していない。より厳密な評価を行うためには、システムコール呼出しや外部との通信、ファイル操作といった実際の振る舞いを解析し、脅威の程度を算出することが必要である。

2 つ目の課題について述べる。本研究では評価対象である 3 つの環境のうちラズビアン環境のみを対象にマルウェアの影響評価を行った。提案手法の有効性を示すためには残り 2 つの環境についても同様の手法により評価を行い、提案手法がマルウェアの影響を評価する上で有効な手法であることを示す必要がある。

3 つ目の課題について述べる。本研究ではラズビアン環境についてのみ影響評価を行いその結果を示したが、各環境の特性を議論するためには残る 2 つの環境についても評価実験を行いそれらの結果を比較することが必要である。

4 つ目の課題について述べる。本研究では、マルウェアの影響評価の手順を自動化することで、手動ではマルウェア 1 つあたりに 180 秒程度の評価時間が掛かっていたところを 100 秒程度まで短縮することを可能にした。しかしながら、数千個以上のマルウェアを評価する場合、数十時間という時間を要するため、提案手法の有用性を向上させるためには更なる時間短縮が望まれる。時間短縮の余地として、QEMU のスナップショット機能を用いることで 60 秒程度掛かっている実行環境の起動を数秒程度に短縮できると予想している。

参考文献

[1] 警視庁. 「令和４年上半期におけるサイバー空間をめぐる脅威の情勢等について」, 2023.2.11. https://www.npa.go.jp/publications/statistics/cybersecurity/data/R04_kami_cyber_jousei.pdf.

[2] 独立行政法人 IPA 情報処理推進機構. 「平成 25 年通信利用動向調査の結果」, 2023.2.11. https://www.ipa.go.jp/security/vuln/10threats2022.html.

[3] Anandharaju Durai Raju, Ibrahim Y. Abualhaol, Ronnie Salvador Giagone, Yang Zhou, and Shengqiang Huang. A survey on cross-architectural iot malware threat hunting. *IEEE Access*, Vol. 9, pp. 91686–91709, 2021.

[4] National Operation Towards IoT Clean Environment. 「notice」, 2023.2.11. https://notice.go.jp.

[5] Hamed HaddadPajouh, Ali Dehghantanha, Raouf Khayami, and Kim-Kwang Raymond Choo. A deep recurrent neural network based approach for internet of things malware threat hunting. *Future Generation Computer Systems*, Vol. 85, pp. 88–96, 2018.

[6] Khanh Duy Tung Nguyen, Tran Minh Tuan, Son Hai Le, Anh Phan Viet, Mizuhito Ogawa, and Nguyen Le Minh. Comparison of three deep learning-based approaches for iot malware detection. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 382–388, 2018.

[7] Hai-Viet Le and Quoc-Dung Ngo. V-sandbox for dynamic analysis iot botnet. *IEEE Access*, Vol. 8, pp. 145768–145786, 2020.

[8] Netscope. 「malware severity levels and detection types」, 2023.2.11. https://docs.netskope.com/en/malware-severity-levels-and-detection-types.html.

[9] PANDA. 「panda.re」, 2023.2.12. https://panda.re.

[10] refade. 「iot_arm」, 2023.2.12. https://github.com/refade/IoT_ARM.

謝辞

本研究を行うにあたり、1 年間に渡り多くのご指導をいただいた名古屋大学大学院情報学研究科情報システム学専攻高田広章教授、および名古屋大学大学院情報学研究科情報システム学専攻松原豊准教授に深く感謝いたします。また、日常の議論を通じて多くの知識や示唆をいただいた高田・松原研究室の皆様に深く感謝いたします。特に、元田匡哉氏、坪井正太郎氏には、研究に関して様々な助言をいただきました。深く感謝いたします。

IoT 機器を想定した実行環境におけるマルウェアの影響評価

２０２３年２月

1019100020 大羽俊輔