Computer Security Symposium 2020
26 - 29 October 2020

ファームウェアの挙動を事前収集するIoTハニーポットの提案および基礎調査

山本 萌花 [1,a)]　掛井 将平[1]　齋藤 彰一[1]

概要：IoT 機器の普及に伴い，IoT 機器を狙った攻撃が増加している．日々進化する攻撃の動向を把握するため，IoT 機器の挙動を模倣して攻撃情報を収集する IoT ハニーポットが利用される．IoT 機器の挙動は多種多様であるため，実機をエミュレートしてハニーポットを構築することが望ましい．そこで，各メーカーが提供しているファームウェアを利用することで様々な IoT 機器の挙動を模倣するハニーポットが提案されている．この手法はファームウェアを仮想マシン上で実行し，インターネットに公開することで高対話型ハニーポットを実現する．しかし，ファームウェアに脆弱性があった場合，攻撃の被害を受ける可能性がある．そこで本稿では，ファームウェアの応答を事前に収集することで安全に運用可能な IoT ハニーポットを提案する．本提案では，ファームウェアに対して脆弱性スキャンや侵入テストを行うことで得られる応答を収集し，ハニーポットへの攻撃に対する挙動を学習する．提案システムの実装にあたって，関連研究および現在取得可能なファームウェアのカーネルやファイルシステムに関して調査を行い，今後の方針について考察する．

キーワード：IoT，ハニーポット，ファームウェア

Proposal and Basic Survey for IoT Honeypot with Pre-collected Firmware Behavior

Moeka Yamamoto[1,a)]　Shohei Kakei[1]　Shoichi Saito[1]

Abstract: As the spread of IoT devices, attacks on IoT devices are increasing. To understand the attack trends, IoT honeypots are used to collect attack information by mimicking the behavior of IoT devices. IoT honeypots should be constructed by emulating the actual devices because IoT devices behave differently. Therefore, firmware-based honeypots are proposed that mimics behavior of various IoT devices by using firmware provided by developers. These methods realize high interactive honeypots by releasing a firmware executed on a virtual machine to the Internet. However, if the firmware is vulnerable, the honeypot may be attacked. In this paper, we propose an IoT honeypot that pre-collecting firmware responses. We collect the responses by vulnerability scanning and penetration testing of IoT firmware and learn behavior it. For the implementation of our proposed system, we survey available firmware and other information. And we discuss future works.

Keywords: IoT, honeypot, firmware

1. はじめに

近年の IT 技術の目覚ましい発展に伴い IoT 機器が急速に普及しており，2022 年には IoT 機器の数は 290 億台を超えると予想されている [1]．IoT 機器はソフトウェアの更新が難しい上に，IoT 機器が攻撃されていることに気づかない可能性が高いため，IoT 機器の脆弱性を狙ったサイバー攻撃の標的となっている．このような対策が不十分な理由としては，IoT 機器は十分なセキュリティ機構を搭載して

[1] 名古屋工業大学
Nagoya Institute of Technology
[a)] 32414105@stn.nitech.ac.jp

いない場合が多いことや，ソフトウェアの自動更新機能が普及していないことが挙げられる．一方，攻撃者は常に脆弱な IoT 機器を探し新しい攻撃を仕掛けている．このような攻撃者に対応するためには，迅速に攻撃の動向や手法を理解し，新たなセキュリティ対策を施さなければならない．

そこで，日々進化する攻撃の動向や IoT 機器の脆弱性を発見するために，IoT ハニーポットが利用される．IoT ハニーポットとは，脆弱性を持つ IoT 機器の挙動を模したシステムであり，攻撃者の侵入および操作を監視することで攻撃情報を収集する．IoT 機器の種類は様々であり，提供するメーカーによって固有の動作が定義されているため，個々の IoT 機器の動作を考慮しない汎用のハニーポットは応答が限定的であるために攻撃者から検出されやすく，効率的に攻撃を収集することができない．したがって，多種多様な IoT 機器の挙動を模倣するには，実機をエミュレートしてハニーポットを構築することが望ましい．しかし，多種にわたる実機の購入は経済的負担となり，ハニーポットの構築および保守運用は困難かつ膨大なコストがかかる．そのような問題点を解決するために，各メーカが提供する IoT 機器専用のファームウェアを仮想マシン上で実行し，インターネットに公開することでハニーポットを実現する Honware[2] が提案されている．Honware はファームウェアイメージから自動でハニーポットを構築するため，実機を必要とせずに低コストで多種多様な IoT 機器を模倣できる．しかし，ファームウェアに脆弱性があった場合，攻撃者によってハニーポットが悪用される可能性がある．

そこで本稿では，ファームウェアの挙動をあらかじめ収集してエミュレートすることで，実機とファームウェアの両方を直接には動作させずに安全に運用可能な IoT ハニーポットを提案する．本提案では，ファームウェアに対して脆弱性スキャンや侵入テストを行うことで得られる応答を収集し，ハニーポットへの攻撃に対する挙動を機械学習による対話モデルを利用して学習する．この手法では，攻撃者により送信されたリクエストをハニーポット上で実行しないため，安全性を保つことができる．また，挙動の学習に機械学習を利用することで実機同様の挙動を実現し，攻撃者からのハニーポットの検出を防ぐ．本稿では，提案システムの実装にあたって関連研究および現在取得可能なファームウェアのカーネルやファイルシステムに関する調査結果と，その結果を元に今後行うべき実装や利用すべきツールについて述べる．

本稿では，まず第 2 章で IoT ハニーポットに関連する技術および研究について述べ，第 3 章で IoT ハニーポットの課題に対する解決方針を述べる．次に第 4 章で提案システムの概要について述べる．そして第 5 章で提案システムの実装に向けて予備調査を行った結果と今後の方針について考察する．第 6 章で倫理的配慮について述べ，最後に全体のまとめを行う．

2. 関連研究

本章では，IoT ハニーポットおよび IoT ファームウェアに関連する研究について述べる．また，機械学習を適用したハニーポットに関する研究について述べる．

2.1 IoT ハニーポット

多種多様な挙動をする IoT 機器を対象としたハニーポットを展開するために，様々な取り組みが行われている．IoT ハニーポットには大きく分けて，低対話型と高対話型がある．低対話型ハニーポットは，プロトコルやアプリケーションをエミュレートすることで対象システムの挙動を模倣するハニーポットであり，導入が容易である．対して，高対話型ハニーポットは脆弱性を残した OS やアプリケーションを攻撃者に提供することで，高度な攻撃の観測が可能である．

低対話型の IoT ハニーポットとして IoTPOT[3] が提案されている．IoTPOT は telnet に基づく攻撃コマンドに対して適切な応答を返すために，サンドボックス環境でコマンドを実行することで応答を生成する．このような低対話型ハニーポットは，攻撃コマンドをインターネット接続されたハニーポット内で実行することがないため安全であるが，エミュレートした範囲の限られた応答しかできないために，攻撃者にハニーポットであることを検出されやすい．

この課題を解決するために，インターネットで公開されている IoT 機器をスキャンすることで様々なリクエストに応答可能なハニーポットを構築する研究が進められている．IoTCandyJar[4] では，インターネットに公開されている IoT 機器に HTTP リクエストを送信することでレスポンスを収集し，機械学習モデルを利用して攻撃に対して適切なレスポンスを返す IoT ハニーポットを提案している．また，Chameleon[5] も同様の仕組みをしており，スキャンを行うことで得られるリクエストとレスポンスを正規化してデータベースに保存しておくことで，攻撃に対して迅速な対応を行う．これらの手法の共通点は，攻撃者による調査段階でハニーポットであることを検知されないことを目指している点である．また，実機を利用せずに構築できるだけでなく，公開されている様々な IoT 機器の挙動を模倣することが可能である．しかし，攻撃コードを含むリクエストをスキャンすると公開 IoT 機器に実害が及ぶため，実際の攻撃に対する IoT 機器の応答は取得できない．また，膨大な IP アドレスの中から IoT 機器を発見することは困難である上に，IoT 機器の IP アドレスは頻繁に変更されるために定期的に公開 IoT 機器を探す作業が必要となる．

一方，高対話型の IoT ハニーポットとして，様々な IoT 機器の実機を用いたハニーポットが提案されている [6]．これは，IoT 機器が提供する WebUI を介して IoT 機器の遠隔操作や設定変更を狙った攻撃を観測することを目的として

いる．このハニーポットによって，80/tcp および 8080/tcp を対象とした HTTP リクエストの収集および攻撃者によるポートスキャンを検知できることを示している．そして，IoT 機器の WebUI においてもインターフェースの構成や機能は機器に応じて様々であるため，実機を利用することが望ましいと述べている．しかし，多種多様な挙動を行う IoT 機器を対象とした高対話型ハニーポットを構築するには多量の実機が必要となり，保守運用のコストがかかる．

実機を用いる高対話型ハニーポットの課題を解決するために，メーカーが提供する IoT ファームウェアを利用することで，実機を用意することなく様々な IoT 機器を対象とした高対話型ハニーポットを実現する Honware[2] が提案されている．Honware は IoT ファームウェアを動作させてハニーポットとして公開することで，効率的にハニーポットの構築および運用が可能であることを示している．しかし，ファームウェア自体に脆弱性が存在している場合は，ハニーポットは攻撃者によってシステム破壊や制御奪取などの被害を受けるリスクを伴っている．

安全性を考慮して VM やコンテナなどの仮想環境で実行するハニーポットが提案されている [7]．しかし，仮想環境で動作するハニーポットは処理速度が遅く，実際のサービスと比べると応答時間が長くなるため，攻撃者にハニーポットを検出されやすいという欠点がある．

2.2 IoT ファームウェア

本節では IoT ファームウェアの概要および関連研究について述べる．

2.2.1 概要

IoT ファームウェアはファームウェアヘッダ，ブートローダー，カーネル，ファイルシステムなどで構成されている．IoT 機器に組み込まれるファームウェアは，GPIO ピンや機器の構成情報が書き込まれた不揮発性メモリ（NVRAM）などのハードウェア機能にアクセスすることが特徴である．IoT 機器はリソースが少なくストレージの容量が限られているため，IoT ファームウェアは zip 形式や tar 形式で圧縮されて EEPROM やフラッシュメモリに書き込まれる．メーカーはアップデートとして IoT ファームウェアを提供している．また，自由な改変を目的としてオープンソースの IoT ファームウェアを公開するメーカーもある．

2.2.2 エミュレーション

IoT ファームウェアに存在する脆弱性を調査するために，IoT ファームウェアのエミュレーションが行われる．IoT ファームウェアのエミュレートには，オープンソースの CPU エミュレータである QEMU[8] が利用されている [9]．QEMU でエミュレーションを行うためのアプローチには，ユーザランドを非特権モードでエミュレートするユーザモードエミュレーションと，カーネルを含めたシステム全体をエミュレートするフルシステムモードエミュレーションがある．ユーザモードを利用したアプローチでは，IoT ファームウェアから得られるファイルシステムに含まれるプログラムがホストマシンのプロセスとして実行される．しかし，IoT ファームウェアのプログラムは動作時に特定のハードウェアにアクセスするため，対象のハードウェアが存在しない場合はプロセスがクラッシュする．特に，多くの IoT ファームウェアがブートプロセス中に NVRAM にアクセスすることが報告されており，IoT ファームウェアのエミュレートにおいて NVRAM を用意できないことが課題であった [10]．

そこで，NVRAM に関連する関数呼び出しに対応するライブラリを追加したカーネルを用意することで，IoT ファームウェアを QEMU のフルシステムモードによりエミュレートするシステムとして，FIRMADYNE[10] が開発された．FIRMADYNE では MIPS リトルエンディアン，MIPS ビッグエンディアン，ARM リトルエンディアンの CPU アーキテクチャを持つ IoT ファームウェアのエミュレーションをサポートしており，フルシステムモードを採用しているために詳細な動的分析が可能である．また，Honware はハニーポットとして IoT ファームウェアを実機同様に動作させるために，FIRMADYNE をインターネットに接続できるように修正している．さらに，ハードウェアが存在しないためにプロセスがクラッシュする場合に備えて，カーネルがシグナルハンドラを無視することでプロセスを強制終了しないように変更している．

ファジングによって IoT ファームウェアに存在する脆弱性を発見するために提案された FIRM-AFL[11] や FIRMCORN[9] でもエミュレータを利用している．FIRM-AFL は，QEMU のユーザモードとフルシステムモードを組み合わせたエミュレータを用意することで，エミュレート時に発生するボトルネックの改善に成功している．FIRMCORN は，軽量な CPU エミュレータである Unicorn Engine[12] を利用してエミュレータを実装している．

2.3 ハニーポットにおける機械学習の適用

効率的な攻撃収集のために，機械学習を利用して攻撃者と自律的に通信を行うハニーポットの研究が進んでいる．これらのハニーポットを自己適応型ハニーポットと呼び，確率モデルや強化学習を利用することで実現される．

IoTCandyJar[4] では，スキャンにより収集した膨大な応答の中から攻撃者が期待している応答を見つける応答選択問題にマルコフ決定プロセス（MDP）モデルを利用する．攻撃者が通信を開始して攻撃コードを入力するまでの通信パターンを MDP モデルに学習させることで，最適な応答選択を行う学習モデルを作成する．

また，QRASSH[13] は攻撃者からのコマンドに対して実行するアクションを強化学習を利用して決定するハニーポットである．モデルの報酬として，より多くの悪性ソフ

トウェアをダウンロードすることを設定し，モデルは報酬を最大化する一連の攻撃者の動作を学習する．

これらの手法は，攻撃者とのやり取りを通じて学習が進み，攻撃に対して最適な応答をするモデルが自動的に作成される．すなわち，ハニーポットへの攻撃が増えるほど学習が進むことを意味する．しかし，ハニーポット運用の初期は学習が進んでおらず，学習に効果的な攻撃が収集できなければ，ハニーポットは限られた応答しかできないことが問題として挙げられる．

3. ハニーポットの構築・運用における課題と解決方針

IoT ハニーポットにおける問題点は，実機を利用することによる構築コストの高さと，運用において安全性を考慮することでハニーポットが攻撃者に検出される確率が高まることである．そこで本提案では，実機の代わりにメーカーが提供する IoT ファームウェアを利用することで低コストで様々な IoT 機器を模したハニーポットを構築する．また，仮想環境で動作させたファームウェアの挙動を事前に収集する．これにより，リクエストを実機で実行する必要がないため安全であり，短時間でレスポンスを送信できるため，攻撃への応答時間による検出が防がれる．

また，収集した IoT ファームウェアの挙動を学習する方法として，確率モデルや強化学習を利用するアプローチが提案されているが，これらは攻撃者との対話を通じて学習が進むモデルである．そのため，学習に効果的な攻撃を得られなければハニーポットは限られた応答しかできない．そこで本研究では，自然言語処理における対話モデルを IoT ハニーポットの応答生成に利用することを検討する．対話モデルは，入力された文章に対する自動応答を目的として質問応答やチャットボットに利用される．自動応答に利用されているモデルとして，長期記憶を得意とするリカレントニューラルネットワーク（RNN）モデルや，GAN を代表とする生成モデルが挙げられ，研究が盛んな分野である．また，対話モデルでは Attention 機構を利用することにより，入力文と出力文に相関のある特徴を抽出する研究が行われている [14]．近年の IoT 機器を狙った攻撃はボットやスクリプトによって自動化されていることが多く，攻撃者がレスポンスから特定のキーワード（IoT 機器の機種など）を取得した場合に攻撃を実行すること [4] や，攻撃リクエストの中に特定の機器に関連する単語が含まれていることが報告されている [6]．そこで，対話モデルにおける特徴抽出手法を適用することで，リクエストとレスポンスの対応関係を学習し，攻撃者が参照している特徴を得られると考えられる．強化学習では学習を進めるために攻撃者との対話を続ける必要があるが，対話モデルではあらかじめデータセットで十分に学習を行うことで，運用開始時においても攻撃者と適当な対話を行うことが可能である．

4. 提案

本章では，広くダウンロード可能である IoT ファームウェアをエミュレートし，挙動を事前に把握することで低コストかつ安全に運用可能な IoT ハニーポットの概要と実装方針を述べる．

4.1 概要

IoT 機器に対する攻撃は，攻撃者によるポートスキャンにより判明した IoT 機器上で動作しているサービスに対して行われる [15]．例えば，管理用に公開している ssh や telnet，Web インターフェースを通じた不正アクセスが挙げられる．システム内部への侵入後には，機密情報の窃取や設定変更，遠隔操作が行われることもある．

本研究では，IoT ハニーポットで起動するサービスに対する攻撃コマンド，およびシステム侵入後に注入される攻撃コードやマルウェアを収集することを目的とする．そのため，IoT ハニーポットへの偵察行為，不正アクセス，そして侵入後の内部探索に対応する挙動をエミュレートする．

提案システムの構成を**図 1** に示す．提案する IoT ハニーポットは，運用に向けてハニーポットの構築を行う準備段階と，構築したハニーポットをインターネットに公開して攻撃を収集する運用段階に分かれている．本稿において，準備段階で挙動収集のために行う攻撃を模擬攻撃，運用段階で攻撃者によって行われる攻撃を実攻撃と呼ぶ．準備段階では IoT ファームウェアからファイルシステムを抽出し，CPU アーキテクチャと一致する適切なカーネルを用いて仮想環境でエミュレートする．準備段階の実行時には IoT ファームウェアにローカルの IP アドレスを割り当て，ターミナルや Web ブラウザから IP アドレスを通じてアクセス可能とする．そして，IoT ファームウェアに対して模擬攻撃を行い，そのリクエストとレスポンスをセットとしてデータベースに保存する．収集したリクエストとレスポンスを対話モデルに学習させ，学習済み対話モデルをハニーポットへの実攻撃に対する応答決定に利用する．

4.2 準備段階

ハニーポットの準備段階は以下の順序で行われる．

(1) IoT ファームウェアの取得（Crawler）
指定したメーカーの Web サイトや FTP サイトをクローリングすることで IoT ファームウェアを収集する．クローラーは Scrapy[16] を利用して実装し, ファームウェアに間連があると考えられる拡張子が.zip や.bin のファイルをダウンロードする．

(2) IoT ファームウェアのエミュレーション（Emulator）
QEMU を利用して，様々な CPU アーキテクチャを持つ IoT ファームウェアをエミュレートする．そのための事前準備として，対象ファームウェアを展開し，

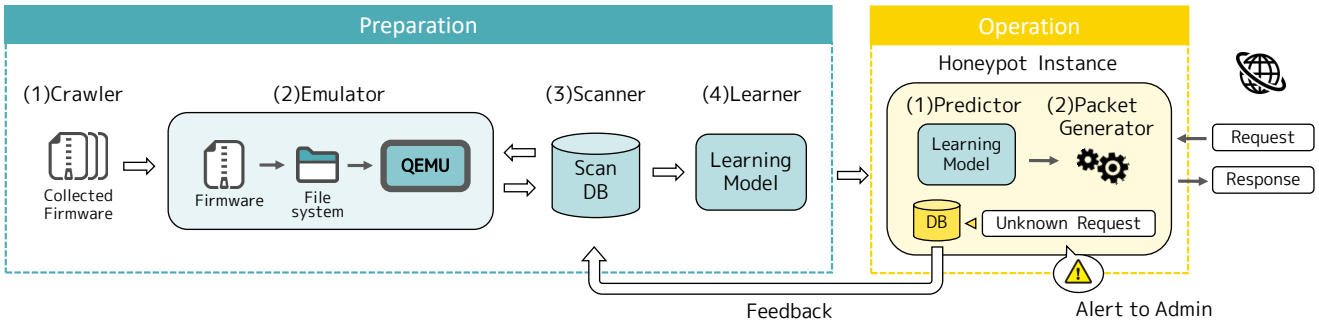

図 1　提案システムの構成図.

Fig. 1　Structure of the proposed system.

Binwalk[17] を利用してファイルシステムを抽出する．ファイルシステムに含まれる Busybox などのプログラムの ELF ヘッダより CPU アーキテクチャを特定する．また，エミュレーションにおいては，IoT のファームウェアに含まれるカーネルは利用せず，NVRAM などの特定のハードウェアをサポートしたカスタムカーネルを用いる．これらにより，QEMU イメージを作成および実行する．起動後のファームウェアはネットワークが遮断されている場合やサービスが停止している場合があるため，ifconfig や service コマンドを利用し，ネットワーク設定とサービスの起動を行う．

(3) IoT ファームウェアの応答収集（Scanner）
起動しているサービスに対して模擬攻撃を行うことで IoT ファームウェアの応答を収集する．様々な応答を収集するためには攻撃者が送信する入力を想定する必要がある．そこで，模擬攻撃として脆弱性スキャンツールや侵入テストツールを利用することで応答を収集する．また，ファジングや動的スキャンによってリクエストを変化させ，可能な限り多くのレスポンスを獲得する．これらのツールによるリクエストおよびレスポンスパケットを Wireshark[18] により収集する．収集したパケットは学習に利用するために，Scapy[19] によって解析を行い，ヘッダ部の主要なフィールドとボディー部を数値型または文字列としてデータベースに保存する．

(4) 挙動の学習（Learner）
ハニーポットが受信したリクエストが模擬攻撃のリクエストと一致していた場合，対応する応答を送り返すことが可能である．一方，ハニーポットが模擬攻撃に含まれていないリクエストを受けた場合，そのレスポンスは獲得していないため，収集データの中から最も適切な応答を返さなければならない．そこで，学習していない未知のリクエストに対して，正解に近いレスポンスを生成することができる機械学習を利用してハニーポットを運用する．模擬攻撃によるリクエストを入力，レスポンスを出力とした機械学習による対話モデルを作成し，その対応を学習させる．リクエストとレスポンスの対応を学習する対話モデルとして Sequence-to-Sequence（Seq2Seq）モデル [20] を検討している．Seq2Seq モデルは機械翻訳で利用されるモデルであり，時系列データや可変長データを対象とする RNN モデルを用いて実装される．さらに，Seq2Seq モデルに対して Attention 機構 [14] を適用することで，リクエストとレスポンスに関連する特徴を抽出する．Attention 機構を利用することで，攻撃者がレスポンスのどの要素を重要視しているかを可視化することができ，攻撃分析において重要な情報が得られると考えられる．

4.3　運用段階

運用段階ではハニーポットを公開して，受信したリクエストに対する応答を以下の流れで作成する．

(1) レスポンスの生成（Predictor）
リクエストに対して前処理を行い，学習済みモデルに入力する．そして，学習済みモデルから得られた出力をレスポンスとして利用する．

(2) レスポンスパケットの生成（Packet Generator）
学習済みモデルより得られた出力に対して，Scapy を利用してハニーポットの IP アドレスや位置情報などの適切なヘッダを付加し，通信相手へレスポンスとしてパケットを送信する．

攻撃者によるリクエストと学習済みモデルが生成したレスポンスは分析用データベースに保存する．新たに観測されたリクエストに学習していないリクエストが含まれていた場合は，未知の攻撃が行われていることを検知したとして管理者に警告を行う．そして，そのリクエストは Scanner にフィードバックを行い，模擬攻撃としてファームウェアの応答を獲得して，対話モデルにその対応を学習させる．この繰り返しにより，提案ハニーポットの精度を向上させる．

4.4 評価手法の検討

提案手法の評価として，ハニーポットの運用段階における性能に関して 3 つの項目を検討する．

1. 提案ハニーポットへの攻撃観測
   提案ハニーポットをインターネットに公開して実攻撃の観測を行い，IoT ハニーポットとして正常に機能するかを確認する．また，観測したアクセスや収集した攻撃に関して分析を行う．
2. 提案ハニーポットの処理速度の算出
   ハニーポットが攻撃リクエストを受信して，対応するレスポンスを送信するまでの処理速度を算出する．攻撃者は攻撃対象の応答時間によってハニーポット検知を行うため，応答が極端に早いまたは遅い場合は検知される可能性がある．
3. HoneyScore による提案ハニーポットの評価
   HoneyScore[21] は Shodan より提供されているハニーポット評価ツールである．評価対象ハニーポットの IP アドレスを入力すると，既知のハニーポットの特徴を利用して対象がハニーポットである確率を 0.0 から 1.0 の範囲で算出するため，ハニーポットが実システムをどの程度模倣しているかの評価基準となる．

5. IoT ファームウェアに関する調査

本章では，現在取得可能な IoT ファームウェアの調査内容およびその結果について述べる．IoT ファームウェアから得られる情報より，安定したエミュレータの構築方法や，応答収集において行うべき模擬攻撃の考察を行う．

5.1 調査項目と方法

実装において必要な IoT ファームウェアの調査項目として以下を挙げる．

- CPU アーキテクチャ
- カーネルの種類
- ファイルシステムの種類
- ファイルシステムに含まれるサービス

ファームウェアの CPU アーキテクチャと，カーネルおよびファイルシステムに関連する情報は，QEMU で動作させるために必要な情報である．Binwalk を利用してファイル構造やプログラムヘッダを調べることによってバージョンを特定する．

また，ファイルシステムに含まれるサービスの調査は，攻撃者によって被害を受ける可能性のあるサービスを特定し，ハニーポットの応答収集のために行うべき模擬攻撃を決定するために行う．収集したファームウェアに IoT 機器で使用される一般的なサービスが含まれているか調査するため，ファームウェアのファイルシステムから情報収集を行うツールである Firmwalker[22] を利用する．

表 1　収集した IoT ファームウェアの CPU アーキテクチャの内訳．
Table 1　Found CPU architecture of IoT firmware.

CPU アーキテクチャ	エンディアン	個数
MIPS	Little	829
MIPS	Big	495
ARM	Little	259
PowerPC	Big	18
ARM	Big	12
MIPS64	Big	5
Intel 80386	Little	4
不明	Little	9
	合計	1,631

5.2 調査結果

5.2.1 収集した IoT ファームウェア

本調査では，FIRMADYNE に付属しているクローラーである Scraper を利用して各メーカーが公開している IoT ファームウェアを収集した．Scraper では 42 メーカーの Web サイトからファームウェアをダウンロードするための Scrapy スクリプトが提供されているが，これらのうち 13 メーカーのスクリプトが現在も利用可能であり，合計 2,087 個の IoT ファームウェアがダウンロードされた．収集したデータセットは，ルータやスイッチ，IP カメラなどのネットワーク機器のファームウェアで構成されていた．

5.2.2 CPU アーキテクチャ

2,087 個の IoT ファームウェアのうちファイルシステムの抽出に成功したものは 1,631 個であった．抽出した各ファイルシステムに含まれる Busybox などの ELF ヘッダを調べることで，対応する CPU アーキテクチャを特定した．その内訳を表 1 に示す．この結果より，MIPS リトルエンディアンおよびビッグエンディアン方式，そして ARM リトルエンディアン方式が合計 1,583 個であり，IoT ファームウェアの多くで採用されているアーキテクチャであることが分かった．よって，エミュレートにおいて上記 3 つの CPU アーキテクチャのカーネルを利用することで，多くの IoT ファームウェアに対応できると考えられる．

5.2.3 カーネル

組み込み機器のカーネルは Linux カーネルが広く利用されていることが知られている [2]．本調査で収集した 2,087 個のファームウェアのうち，Binwalk によってカーネルの種類を特定できたファームウェアは 1,503 個であり，すべて Linux カーネルが採用されていた．また，文献 [2] では，ファームウェアのエミュレートには意図的に古いバージョンの Linux カーネルを利用している．これは，古いバージョンのカーネルが広く普及しており，その多くが新しいカーネルのサポートを必要としないためである．よって，本研究においても FIRMADYNE や Honware で利用されている MIPS の Linux カーネル 2.6.32 および ARM の Linux カーネル 4.1.52 を利用する予定である．

表 2　収集した IoT ファームウェアに利用されているファイルシステムの種類と内訳
Table 2　Found file systems of collected IoT firmware.

ファイルシステム	個数
Squashfs	1,410
JFFS2	22
Linux Ext	19
CramFS	1
NTFS	1
不明	178
合計	1,631

表 3　収集した IoT ファームウェアに含まれているサービスの内訳.
Table 3　Found Services of collected IoT firmware.

サービス	個数	割合 [%]
httpd	1,497	91.8
openssl	858	52.6
lighthttpd	108	6.6
telnet	47	2.9
telnetd	39	2.4
tftp	36	2.2
dropbear	26	1.6
ssh	24	1.5
sshd	18	1.1
scp	12	0.7

5.2.4　ファイルシステム

収集した IoT ファームウェアに適用されているファイルシステムの種類について調査した．その結果を表 2 に示す．Squashfs が抽出されたファイルシステム全体の 86.4% を占めていた．Squashfs は Linux 向けのファイルシステムであり，NTFS を除くその他のファイルシステムも Linux 向けのファイルシステムであった．ファイルシステムの方式が不明のファームウェアは，提供するメーカー独自の方式を利用しているため解析およびエミュレートが困難である [2]．そのため，本研究では Linux 向けファイルシステムを採用しているファームウェアをエミュレート対象とする．

5.2.5　サービス

収集した IoT ファームウェアに含まれているサービスを調査した．表 3 はファームウェアから抽出したファイルシステム内に含まれるサービスの有無を調べることにより判明した全サービスの一覧である．調査対象 1,631 個のファームウェアのうち，9 割以上のファームウェアに Web サービスが含まれていた．これは IoT 機器の設定や管理を行うための WebUI を提供するサービスである．また，遠隔操作するための ssh や telnet，ファイル転送サービスの tftp や scp が含まれており，外部からの操作が可能であるこれらのサービスは攻撃者に狙われる可能性が高い．

5.3　FIRMADYNE の性能調査

収集した IoT ファームウェアが FIRMADYNE によりエミュレート可能であるか調査した．2,087 個のファームウェアのうち FIRMADYNE による QEMU イメージの起動に成功したのは 230 個であり，成功率は 11.0%であった．起動に失敗したイメージの QEMU 起動時のログを確認したところ，大半が init バイナリを正しく実行できないことによる IoT ファームウェアのブートプロセスで失敗していた．この問題は文献 [2], [9] で改良方法が提案されている．それらの手法を参考にして FIRMADYNE を拡張し，IoT のファームウェアの起動の成功率を上げることが本研究での課題である．

5.4　提案手法の実装に向けた方針

本節では，IoT ファームウェアに関する調査結果から提案ハニーポットの実装に向けた方針を述べる．

5.4.1　クローラー

今後収集するファームウェアを増やすために，Scraper のプログラムを変更し，クローリング対象 Web サイトを増やす必要がある．メーカーや IoT 機器の種類によって IoT ファームウェアの挙動は異なるため，可能な限り多種にわたる IoT ファームウェアの取得を目指す．

5.4.2　エミュレータ

IoT ファームウェアに対応する CPU アーキテクチャの多くは MIPS のリトルエンディアン，ビッグエンディアン方式または ARM のリトルエンディアン方式であった．FIRMADYNE はこの 3 つの CPU アーキテクチャを持つ IoT ファームウェアに対応するが，調査において FIRMADYNE によるエミュレートに成功した IoT ファームウェアはデータセット全体の 11.0%であった．そのため，FIRMADYNE で提供されるカーネルを改良し，IoT ファームウェアが NVRAM や IoT 機器特有のハードウェアにアクセスする場合に広く対応させる必要がある．IoT 機器特有のハードウェアには NVRAM の他に，GPIO ピンや追加ストレージなどの周辺機器が含まれる [10]．IoT ファームウェアのプログラム実行時の挙動や依存するハードウェアはメーカーや機種によって異なるため，安定したエミュレータを構築するにはファームウェアの緻密な分析が必要である．

5.4.3　模擬攻撃

攻撃者が興味を持つのは IoT 機器上で動作するサービスであるため，安定して動作するエミュレータを作成し，様々なリクエストに対応する応答を収集する必要がある．

調査により，IoT ファームウェアには Web サービスが多く含まれていることが分かった．そこで，実装段階では 80 番ポートで起動する Web サービスに対する偵察行為を対象として，ファームウェアに模擬攻撃を行うことで挙動を学習する．Web サービスに対するオープンソースの脆弱性スキャンおよび侵入テストに利用されるツールとして

OWASP ZAP[23] の利用を検討している．OWASP ZAP はリクエストのファジングや動的スキャン機能を有するため，リクエストの変化に対するサービスの挙動を詳しく収集できると考えられる．また，運用時に対応できなかった実攻撃を観測した場合は，模擬攻撃の対象範囲を拡大させ，より多くの実攻撃に対して応答できるように挙動を学習する．

6. 研究倫理

今回予備調査として利用した IoT ファームウェアは各メーカーの Web サイトで提供されているものを利用した．エミュレート時において，IoT ファームウェアそのものをインターネットに公開することはなく，外部との通信は行わない．模擬攻撃によって得られる応答の中には機密情報が含まれる可能性がある．ハニーポットで利用する際には，そのような情報を除いた応答に変換することで安全性を保証する予定である．また，脆弱性を発見した場合は対象となるメーカーにすみやかに報告し，対処を依頼する予定である．

7. おわりに

未知の脆弱性や攻撃方法を検知することを目的として，実機を使用せずに安全かつ低コストで運用可能な IoT ハニーポットの実現に向けた提案について述べた．本提案では，広くダウンロード可能な IoT ファームウェアに対して事前に模擬攻撃を行い挙動をエミュレートすることで，安価かつ安全に考慮したハニーポットの構築を行う．さらに，IoT ファームウェアの挙動の学習として機械学習を利用した対話モデルを検討し，すべてのリクエストに対して適切な応答が可能なモデルを作成することで，攻撃者から検出されにくいハニーポットの構築を目標とする．

実装に向けた予備調査として IoT ファームウェアを収集し，カーネルやファイルシステムの種類やファームウェアに含まれているサービスに関して調査を行った．これらの情報から，今後利用すべきツールやエミュレート環境に関する考察を行った．

今後の方針として，安定した IoT ファームウェアのエミュレート環境を用意し，脆弱性スキャンや侵入テストを行うことで応答を収集する．さらに，対話モデルによって挙動を学習し，攻撃に対して適切な応答を行うハニーポットの運用に向けて実装を進める．

謝辞 本研究の一部は，JSPS 科研費基盤研究 (C)19K11962 による助成を受けたものです．

参考文献

[1] Ericsson : Ericsson Mobility Report 2017, https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf.
[2] Vetterl, A., and Clayton, R. : Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days, 2019 APWG Symposium on Electronic Crime Research (eCrime), pp. 1–13 (2019).
[3] Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. : IoTPOT: analysing the rise of IoT compromises, 9th USENIX Workshop on Offensive Technologies (2015).
[4] Luo, T., Xu, Z., Jin, X., Jia, Y., and Ouyang, X. : IoTCandyJar: Towards an Intelligent-Interaction Honeypot for IoT Devices, Black Hat (2017).
[5] Zhou, Y. : Chameleon: towards adaptive honeypot for internet of things., Proceedings of the ACM Turing Celebration Conference-China, pp. 1–5 (2019).
[6] 江澤優太, 田宮和樹, 中山颯, 鉄穎, 吉岡克成, 松本勉 : 実機を用いたハニーポットによる組込み機器の WebUI に対するサイバー攻撃の分析, コンピュータセキュリティシンポジウム 2017 論文集, 2017,Vol. 2 (2017).
[7] Garfinkel, T., Adams, K., Warfield, A., and Franklin, J. : Compatibility Is Not Transparency: VMM Detection Myths and Realities., HotOS (2017).
[8] QEMU, https://www.qemu.org/.
[9] Gui, Z., Shu, H., Kang, F., and Xiong, X. : FIRMCORN: Vulnerability-Oriented Fuzzing of IoT Firmware via Optimized Virtual Execution, IEEE Acces, vol. 8, pp. 29826–29841 (2020).
[10] Chen, D. D., Woo, M., Brumley, D., and Egele, M. : Towards Automated Dynamic Analysis for Linux-based Embedded Firmware, NDSS, Vol. 16, pp. 1–16 (2016).
[11] Zheng, Y., Davanian, A., Yin, H., Song, C., Zhu, H., and Sun, L. : FIRM-AFL: high-throughput greybox fuzzing of iot firmware via augmented process emulation, 28th USENIX Security Symposium, pp. 2099–1114, (2019).
[12] Unicorn, https://www.unicorn-engine.org/.
[13] Pauna, A., Iacob, A. C., and Bica, I. : Qrassh-a self-adaptive ssh honeypot driven by q-learning., 2018 international conference on communications, pp. 441–446 (2018).
[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I. : Attention is all you need., In Advances in neural information processing systems, pp. 5998–6008 (2017).
[15] 井上博之 : IoT（つながる組込み機器）における脅威の現状, 精密工学会誌, vol 83, No 1 (2017).
[16] Scrapy, https://scrapy.org/.
[17] Binwalk, https://github.com/ReFirmLabs/binwalk.
[18] Wireshark, https://www.wireshark.org/.
[19] Scapy, https://scapy.net/.
[20] Sutskever, I., Vinyals, O., and Le, Q. V. : Sequence to sequence learning with neural networks, Advances in neural information processing systems , pp. 3104–3112 (2014).
[21] Honeyscore, https://honeyscore.shodan.io/.
[22] Firmwalker, https://github.com/craigz28/firmwalker.
[23] OWASP ZAP, https://www.zaproxy.org/.