平成 30 年度 公立はこだて未来大学卒業論文

DDoS 攻撃を行うマルウェアの IoT デバイス本体における検知手法の提案

水上 敬介

情報アーキテクチャ学科　　1015237

指導教員　　(主) 稲村 浩　　(副) 中村 義隆

提出日 2019 年 1 月 29 日

Proposal of Detection Method in IoT Device of Executing DDoS Attack

by

Keisuke Mizukami

BA Thesis at Future University Hakodate, 2019

Advisor: Prof. Inamura, Coadvisor: Prof. Nakamura

Department of Information Architecture
Future University Hakodate

January 29, 2019

Abstract– In recent years, IoT devices equipped with communication functions in various things are spreading explosively. As a result It is a social problem that a botnet is constructed by an IoT device infected with malware and a DDoS attack is performed. Malware called Mirai is published on the web site, and many variants of Mirai are made. In this research, we aim to detect malware that performs DDoS attack on IoT device , and to detect unknown malware. we pay attention to the fact that the variants of malware were make from published malware on web site, we detect malware that performs DDoS attack by determining whether there is a specific function of the original malware

Keywords: DDoS attack, IoT Device, malware, Mirai, Linux

概 要: 近年，世の中にある様々なものに通信機能を搭載した IoT 機器が爆発的に普及している．その結果，マルウェアに感染した IoT 機器によってボットネットが構築され大規模な DDoS 攻撃が行われ大きな問題となっている．その中でも，Mirai と呼ばれるマルウェアが Web 上で公開され，Mirai の亜種が多く作られている．本研究では，IoT デバイス本体において DDoS 攻撃を行うマルウェアを検知する手法を検討することによって，未知のマルウェアでも検知を行えることを目的とする．公開されているマルウェアを元に亜種が作成されていることに着目をして，オリジナルのマルウェアが持つ特定の関数の挙動を検出することによって DDoS 攻撃を行うマルウェアの検知を行う．

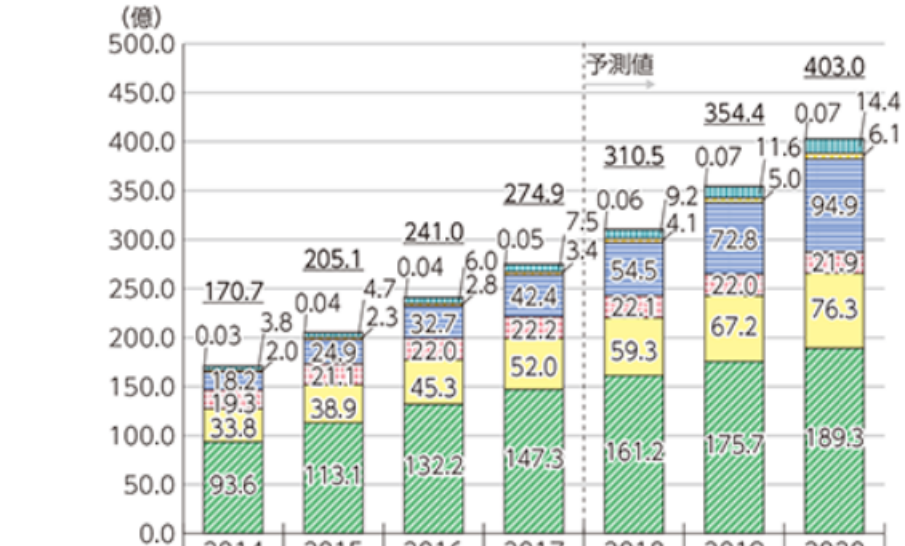
キーワード: DDoS 攻撃，IoT デバイス，マルウェア，Mirai

目 次

第1章　序論

1.1　背景

近年，インターネット技術やセンサー技術の進化を背景に，パソコンやスマートフォンなどのインターネット端末に加え，家電や自動車などの様々なものに通信機能を搭載したIoT デバイスが普及し始めている．総務省によると政界中の IoT デバイスの数は図 1 のように 2017 年時点で IoT デバイスが約 275 億台存在し,2020 年には IoT デバイスが 403 億台に及ぶと予想されている [1]．IoT デバイスの普及に伴い，IoT デバイスを対象とした



	2014	2015	2016	2017	2018	2019	2020
合計	170.7	205.1	241.0	274.9	310.5	354.4	403.0
軍事・宇宙・航空	0.03	0.04	0.04	0.05	0.06	0.07	0.07
自動車	3.8	4.7	6.0	7.5	9.2	11.6	14.4
医療	2.0	2.3	2.8	3.4	4.1	5.0	6.1
産業用途	18.2	24.9	32.7	42.4	54.5	72.8	94.9
コンピューター	19.3	21.1	22.0	22.2	22.1	22.0	21.9
コンシューマー	33.8	38.9	45.3	52.0	59.3	67.2	76.3
通信	93.6	113.1	132.2	147.3	161.2	175.7	189.3

図 1.1: 世界の IoT デバイス数の推移及び予測

マルウェアが急増している．IoT デバイスの重要な問題の 1 つとしてセキュリティ問題が挙げられる．IoT デバイスのユーザ名やパスワードを初期設定の状態で使用する場合が多いことやデバイスの資源が限られていることから，セキュリティが十分に考慮されていない事がある．そのため，IoT デバイスを対象としたマルウェアが脅威となっている．その

中でもネットワークサービスを停止させる深刻な問題を引き起こしているマルウェアにはDDoS(Distributed Denial of Service) 攻撃を行っているものが多く存在し，その対策が重要視されている．DDoS 攻撃は，攻撃者が複数の他人のコンピュータを利用し，公開されているサービスに大量のデータを送りつける事によって処理負荷を与えサービスを機能停止に追い込む攻撃である．代表的な DDoS 攻撃を行うマルウェアとして Mirai が挙げられる．Mirai[2] は，無作為な IP アドレスから感染できるデバイスを探し出し，ログイン可能なデバイス上に，悪意のあるソフトウェアをダウンロードし実行させることでそのデバイスを制御下に置く．攻撃者によって制御された端末は他に侵入可能な端末を探し出し，次々と感染させることでボットネットと呼ばれる悪意あるプログラムを使用して乗っ取った多数のコンピュータで構成されるネットワークを構築する．その後，C&C(Command and Control) サーバから送られた指示に対して DDoS 攻撃を行うマルウェアである．2016 年 10 月に発生した，DNS サーバープロバイダである Dyn 社への DDoS 攻撃では IoT デバイスによるボットネットが利用され史上最大規模である 620Gbps の攻撃が観測された [3]．その後，Mirai のソースコードが公開され，Owari，Satori，Okiru といった Mirai の亜種の開発が盛んに行われるようになった．2017 年には，Windows PC を踏み台にして感染可能な IoT デバイスを探索する Mirai が観測された [9]．この Mirai は DoS 攻撃を行う機能を有していないが，探索後にログイン可能な端末が Linux デバイスであれば DoS 攻撃機能を有した Mirai をダウンロードさせ，Windows PC であればスキャン機能に特化した Mirai をダウンロードさせ，さらなる Mirai を様々な IoT デバイス拡散することが可能になっている.Mirai や Mirai 亜種のマルウェアによって，多くの IoT デバイスが DDoS 攻撃に不正利用されるようになったことから，国立研究開発法人情報通信研究機構がパスワード設定などに不備のある IoT 機器の実態把握を目的として日本国内の IPv4 アドレスを対象に SSH，Telnet，HTTP である TCP の 22 番，23 番，80 番ポートを対象にポートスキャンを行った [8]．IoT デバイスの不正利用による DDoS 攻撃が社会的に注目されている．

1.2 IoT デバイスで検知を行う必要性

IDS(Instrusion Detection System) と呼ばれるマルウェアによる不正な通信やホストへの侵入，ファイルの改ざん等の不正な挙動の兆候を検出するシステムの設置場所として 2 種類が考えられる．ネットワーク上に設置するネットワーク型と端末上に設置するホスト型の 2 種類が考えられる．ネットワーク型の IDS では，ネットワークに流れるデータを取得して解析し，異常がないか確認する．不正が疑わ得れるデータを検知したときには管理者に知らせる．ネットワーク上でネットワークトラフィックから DDoS 攻撃を判別するのは難しく誤検知する可能性が考えられる．しかし，マルウェアに基づいて作成されたデータを用いたパターンマッチングによる検知手法では，誤検知率が低く既存のマルウェアを確実に検知できる利点が有る．公開されているソースコードを基に作成されたマルウェアは，オリジナルのマルウェアと共通するシグネチャが存在すると考えられるためパターンマッチングによる検知で亜種のマルウェアにも対応できると想定される．脅威となっているマルウェアは，十分に管理が行われていない IoT デバイスで散見される，放置された初期パスワードのままのアカウントや，保守されていないシステムの脆弱性をついた攻撃を

行うため，侵入されてしまうことは前提とすべきである．そのため，デバイスの性能が限られている IoT デバイス上でもマルウェアの検知を行う必要がある．

1.3　マルウェア Mirai の概要

Mirai は，ネットワーク上で公開されている Linux で動作するデバイスを不正利用し，DDoS 攻撃を行うマルウェアである．ネットワークカメラやルータといった IoT デバイスをターゲットにしている．Mirai は，C&C サーバー，MySQL，Loader，bot の 4 つから構成される．Mirai の概要を図 2 に示す．

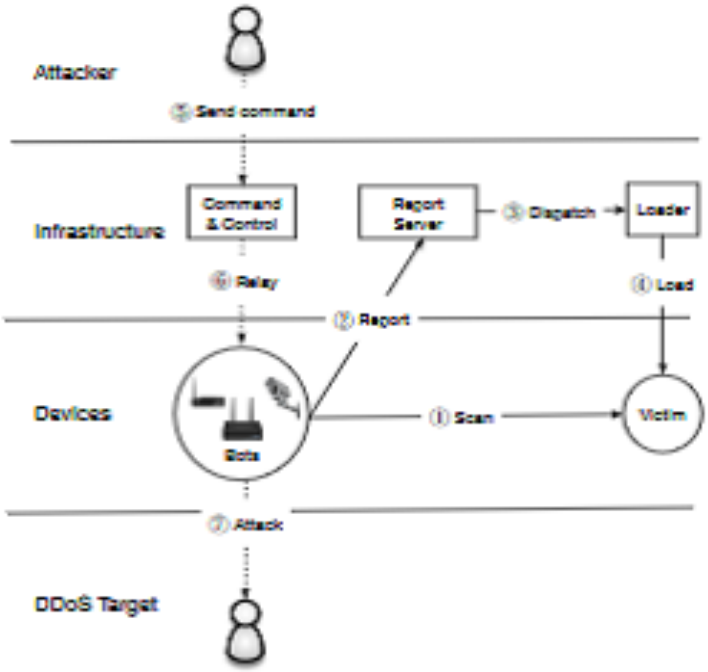


図 1.2: Mirai の概要図

C&C サーバーは，ボットやユーザーからの接続されるのを待機しており，主な機能としてボット管理機能，ユーザー管理機能，攻撃指示機能がある．MySQL にはユーザーのリストと攻撃履歴が記録されるようになっている．bot は，C&C サーバーからのコマンドを待機し，感染先でボットネットに加えられる新しいデバイスを探索するスキャン活動を行う．スキャン活動を行いログインできる端末を見つけた場合には，IP アドレス，ポート，ログイン情報をスキャンサーバーへと送る．感染経路について，Mirai は、Telnet ログインが可能な場合に感染する．Loader は，スキャン活動からレポートサーバーに送られた攻撃対象の情報をもとに Telnet ログインを試みる．Telnet ログインに成功した際には，攻撃者が用意した http サーバーまたは tftp サーバーから，Mirai のバイナリファイルを対象の IoT デバイスにダウンロードし bot を実行させる．bot の動作後には，C&C サーバーとの通信を始め，C&C サーバーから送られてくる攻撃命令を受け取り，IoT デバイ

スが特定のサーバーに攻撃を始める．

1.4 研究目的

本研究では，DDoS 攻撃を行うマルウェア Mirai とその亜種の未知のマルウェア検知である．

1.5 論文の構成

本論文は，1 章に背景，2 章では関連研究・技術，3，4 章では検知手法の提案，5 章では評価実験を行い，6 章でまとめを述べる．

第2章　関連研究・技術

2.1　関連研究

2.1.1　DDoS 攻撃を行うマルウェアの分析

DDoS 攻撃を行うマルウェアの分析を行った研究が行われている．組込みシステム向けマルウェア Mirai の攻撃性能評価 [4] では，Mirai を VM 上で動作させ通信の様子や攻撃の流れの動作を確認し，攻撃性能を計測した．ローカルネットワーク上で実機実験として複数の組み込みボード (odroid-c2) を用いて VM と同様に Mirai の動作環境を構築し，攻撃性能の影響の調査を行った．組み込みシステム向け TCP/IP スタックからなる http サーバーが動作する静的な組込みシステムのプロトタイプを対象に攻撃を行い DDoS 攻撃時には，CPU 使用率が大幅に上がることを明らかにした．しかし，研究結果からマルウェアに対して具体的な検知手法の提案がなされていない．IoT マルウェアによる DDos 攻撃の動的解析による観測と分析 [13] では，ハニーポットを用いて収集した IoT マルウェアの検体を用いて ARM，MIPS，MIPSEL の 3 種類の CPU アーキテクチャを用いてマルウェアを動作させその挙動を観測し，ダミー C&C サーバーを用いて攻撃再現実験を行い DoS 攻撃の観測を行った．マルウェアに対して DoS 攻撃命令が届くタイミングは各感染ホストによって様々であり，マルウェアの動作直後に集中されるわけではないことがわかった．このことから，動的解析により DoS 攻撃の命令を観測する場合には，長期的な観測が必要になる．

2.1.2　動的解析を行ったマルウェア検知の研究

マルウェアの検知手法に関して，API を特徴として用いた研究が広く行われている．API 呼び出しパターンに着目した検知手法 [11] では，API 呼び出しパターン，API 呼び出しによる経過時間とシステム負荷を特徴量としたマルウェア検知手法を提案した．マルウェア 1 検体あたりに 10 間動作をさせ，その間に得られた動的解析ログから API 呼び出しとそれに伴う経過時間とメモリ使用量の情報を抽出し，マルウェアの特徴抽出を行い，機械学習アルゴリズムを用いてマルウェア検知を行う．結果として，API 遷移がほとんど重複していないマルウェアに関しては高い精度で検知を行う事ができた．しかし，呼び出される API がある程度重複しているマルウェア検体を用いた実験を行っていないため，呼び出される API が重複している場合は，検知精度がどの様になるのか明らかにされていない．実行ごとの挙動の差異に基づくマルウェア検知手法 [12] では，マルウェアを複数回実行した際の挙動の差異を判断することによってマルウェアの検知を行う．検査対象である 1 つのマルウェアを 2 回動的解析を行い，それぞれの実行時の API 呼び出しログを取得しログから特定の API の引数を抽出し 2 つの実行ログから取得した引数が異なっている場合

にマルウェアと判断を行った．しかし，毎回決まった動作を行うマルウェアは挙動の変動が見られないため検知ができなかった．しかしそのようなマルウェアに対してはパターンマッチング方による検知が有効だと考えられ，提案手法と組み合わせた効率的な検知手法の提案が課題になっている．この検知手法では，特定のサーバーにマルウェアだと思わしきバイナリファイルを送信し実行してログを取得しているため，Mirai のように実行後に自身のバイナリファイルを消してしまうマルウェアには有効ではない．アノマリ手法を用いた IoT 機器マルウェア感染検出 [5] では，IoT デバイスをもしたハニーポットを用いて多くのマルウェアからダウンロードされたバインリファイル，スクリプトファイルの収集を行った．収集したファイルを用いて動的解析を行い実際に，マルウェアが行う通信を記録した．マルウェアがおこなう通信が IoT デバイスの本来の通信とは異なることを明らかにし，C&C サーバーとの通信を検知することによってマルウェアの検知を行った．しかし，C&C サーバーとの通信が遮断されていたり，通信が暗号化されている場合には検知ができない．通信以外の挙動を併せて検知することでこの問題は解決可能だと考えられ，マルウェアの多様な挙動が観察可能という点で IoT デバイスでのマルウェアの検知は妥当だと考えられる．

2.2 関連技術

2.2.1 Arbor Networks Peakflow

トラフィック管理技術の NetFlow などを使用してネットワーク全体をモニタリングし，DDoS 攻撃の恐れがあるトラフィックを検知する．疑わしいトラフィックにてついては，DDoS 攻撃をさ緩和せる TMS と呼ばれるに中継させ世紀のトラフィックだけを通信させる. このシステムでは DDoS 攻撃だと思われるトラフィックを検知してから 30 秒以内に DDoS 攻撃の緩和動作を始める．

2.2.2 Clam AntiVirus

Clam AntiVirus はオープンソースで提供されているクロスプラットフォームのアンチウィルスソフトウェアである．シグネチャと呼ばれるマルウェアの特徴を記載したファイルによるパターンマッチング方式を　採用しており，約 21755 種類のウィルスに対応をしている．公開されているシグネチャを用いてホスト上にあるファイルをスキャンしシグネチャと一致したファイルが無いか探索を行う．シグネチャと一致するファイルが有った場合には，ユーザーに対して通知を行う．

第3章　シンボルテーブルを用いた検知手法の提案

3.1　検知手法へのアプローチ

検知手法を定めるために，IoT デバイス上で普段の動作とマルウェアがダウンロードされ実行されたあとの動作の違いを明らかにする．その後，Mirai を実際に動作をさせデバイス上で行われている動作の解析を行う．実行コマンド，プロセスの 2 つのログデータの収集を行い，マルウェアが実行される前と後の相違点を明らかにする．

3.2　Mirai ソースコードを用いた稼働調査

Mirai の挙動の稼働調査として Web 上で公開されている Mirai のソースコード [7] を用いてマルウェアが感染する際の感染動作と C&C サーバーとの通信が行われ，攻撃命令を待機するまでのマルウェアの動作を確認した．Mirai が IoT デバイスに感染する様子を確認するために，VM を用いた解析環境を図 2 に示す．Mirai をダウンロードさせ実行させるための感染端末，Loader,C&C サーバー,MySQL の 4 つを用意した．MySQL に C&C サーバーの管理ユーザを登録し C&C サーバーと bot の通信状態を確認できるようにした．Loader が感染端末に Telnet ログインを行い，感染端末の通信が確立される．ログイン後に実行されるコマンドの収集を行い、表 1 に示すコマンド列を得た．表 1 のように Mirai はバイナリファイルをダウンロードした際に，バイナリファイルの名称を dvrHelper に変更している・しかし，バイナリファイルの実行後に，ps コマンドでプロセス名を確認すると，無作為なプロセス名で動作し，他の端末から Telnet ログインができなくなっていることが確認された．Mirai には，DDoS 攻撃を行う機能だけではなく，特定のポートを閉じる機能やプロセス名を無作為にする機能が存在することが確認された．

表 3.1: 表 1 マルウェアによる実行コマンド

/bin/busybox wget;
/bin/busybox wget
http://192.168.32.10:80/bins/mirai.x86
-O ->dvrHelper;
/bin/busybox chmod 777 dvrHelper;
./dvrHelper telnet．x86;

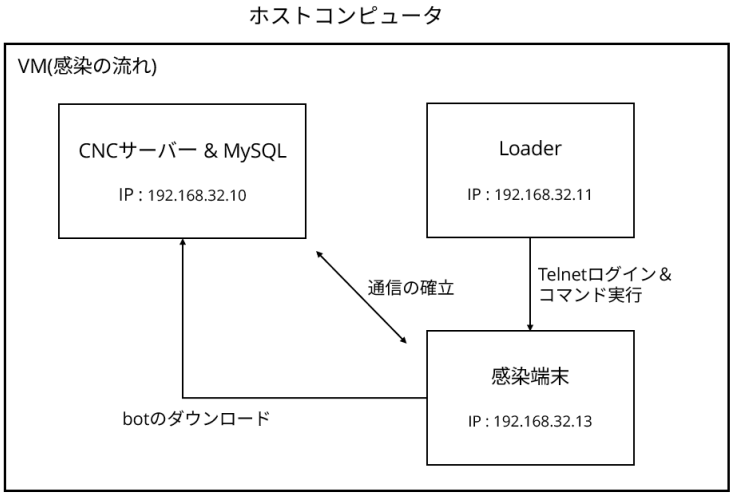


図 3.1: Mirai の解析環境

3.3 シンボルテーブルを用いた検知手法の提案

計算資源が潤沢でない IoT デバイス上でも実現可能な，Mirai 亜種の動作を検知する軽量な動的解析に基づく検知システムを提案する．検知システムの概要を図 3 に示す．Mirai とその亜種である Owari を含めて調査したところ，DDoS 攻撃を行うマルウェアについて亜種を含めて同様の機能を持つ，同一のコードが再利用されていることが確認された．そこで，マルウェアが持つ特定の関数の具備を検知条件として定め，この条件を満たすプロセスの稼働を検出することによってマルウェア感染の有無を判定する手法を以下に述べる．

1. IoT デバイス上で動作を行うプロセスのホワイトリストを作成する．ホワイトリストとは，端末上で可動が許可されたプロセスリストのことである．

2. プロセスを監視し、作成されたホワイトリストをもとに記載がないプロセスを発見する．

3. ホワイトリストにないプロセスに関して，プロセスを動かしているバイナリファイルのシンボルテーブルを確認し，プロセス名を無作為に変更するなどのマルウェアの特定の関数が存在しているか確認を行う．

4. マルウェアが持つ特定の関数の存在が確認できた場合には，マルウェアだと判断を

行う．

5. ホワイトリストにないプロセスに関して，シンボルテーブルの探索が終わった場合には 2 に戻る

検知項目でマルウェアが持つ特定の関数として，DDoS 攻撃を行う関数や，事前調査で把握した，プロセス名を無作為にする関数や，特定のポートを閉じる関数などが候補に挙げられる．

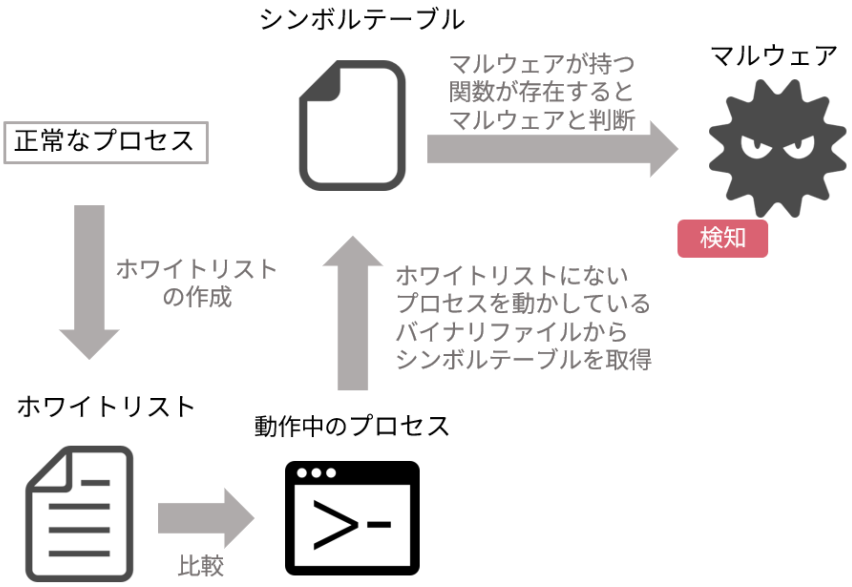


図 3.2: 検知システムの概要

3.4　マルウェア探索動作による負荷と検知における制約事項

IoT デバイスは放置されることが多く，常に操作を行っているわけではない．そのため継続的にアンチウィルスソフトウェアなどの検知システムを利用して IoT デバイスにマルウェアがダウンロードされ実行されていないか確認をし，IoT デバイスが安全な状態であることを把握する必要がある。検知システムのマルウェア探索動作によって，IoT デバイスの動作が妨げられる可能性がある．検知システムの動作を行っている際の IoT デバイスの状態はマルウェアが動作している状態とマルウェアが動作していない 2 種類に分類される．IoT デバイス上で Mirai など DDoS 攻撃をおこなうマルウェアが動作している際には，特定のサーバーに対して DDoS 攻撃を行ってしまうため IoT デバイスの正常な動作を妨げてまでマルウェアを検知する必要がある．しかし，IoT デバイス上でマルウェアが動作していない状況下において映像，音声，ログなどの様々なデータを伝達するため動作等がマルウェアの探索動作によって阻害されてはならない．そのため，IoT デバイスにマルウェアが動作していない状況下において，提案した検知システムによるマルウェア探索動作が IoT デバイス本来の動作を阻害していないか評価を行う．また，IoT デバイスに Mirai が動作した場合に，提案した検知システムによって検知が可能であることを評価する．提案手法の可動に必要なマルウェア探索動作によって IoT デバイス本来の動作が阻害されてないことを評価するために，LinuxOS を対象とする既存のアンチウィルスソフトである Clam AntiVirus を動作させた状態の CPU とメモリ使用率をそれぞれ基準値とし，提案手法によるマルウェア探索動作の動作負荷について比較を行い，併せて提案手法によって Mirai マルウェアの検知が可能であることを確認した．提案した検知システムによって Mirai が動作していない状況での、CPU, メモリの使用率について sar と呼ばれるシステムの負荷状況を確認するコマンドを用いて 1 分間計測を行なった．表のスペックのラズベリーパイを利用し負荷状況を測定した．

表 3.2: 評価環境の IoT デバイスのスペック

評価環境の IoT デバイスのスペック	
OS	Openwrt 4.9.120
CPU	Quad Core 1.2GHz Broadcom BCM2837
MEM	1GB

sar コマンドを用いて得た CPU, メモリの使用率について Clam AntiVirus と提案した検知システムの比較を行った結果が図 4，5 になる．Clam AntiVirus を利用した場合には，平均 CPU 使用率が 25.28%，メモリ使用率は 7.93%となった．提案した検知手法では，平均 CPU 使用率が 3.03%，メモリ使用率が 7.21%となった．メモリ使用率は比較対象の Clam AntiVirus と提案した検知手法では 12.5%減，CPU 使用率は，Clam AntiVirus に対して提案した検知手法は 88%減となったことからマルウェアの可動を検知する目的で一般的によく利用される Clam AntiVirus に比較して提案手法の実装では資源消費が少なく他のプロセスの動作を妨げる可能性は低いと言える．しかし，提案した検知した検知手法は実行形式ファイルに含まれるシンボルテーブルの内容に基づいている為，マルウェアの実行形式ファイルに対して strip コマンドを用いるなどしてシンボルテーブルが削除された場合には検知が行えないという課題がある．

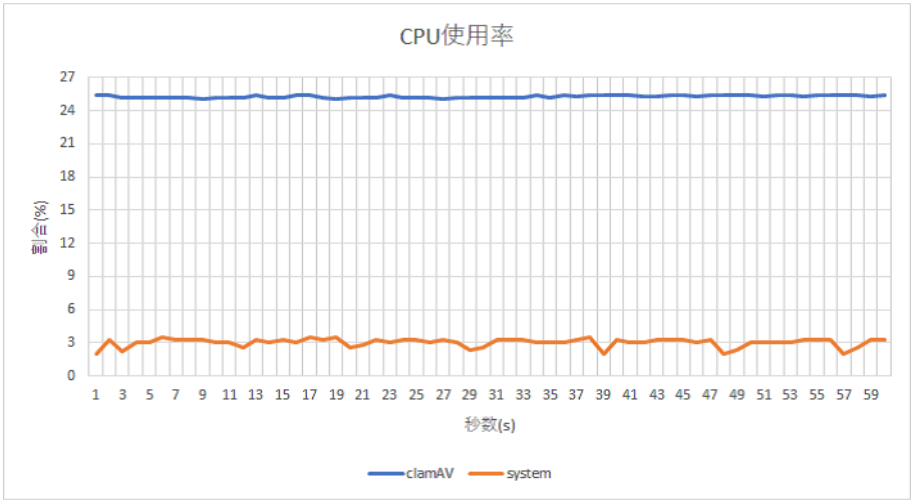


図 3.3: IoT デバイス上でマルウェアが動作していない状況下におけるマルウェア探索のメモリ使用率

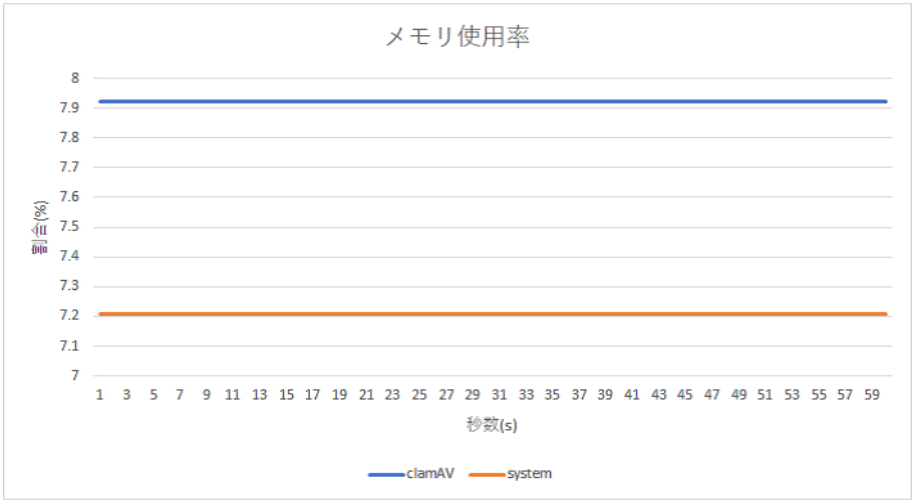


図 3.4: IoT デバイス上でマルウェアが動作していない状況におけるマルウェア探索の CPU 使用率

第4章　システムコール呼び出し履歴を用いた検知手法の提案

4.1　新たな検知手法の必要性

前章で述べたシンボルテーブルを用いてマルウェアの検知を行う検知手法では，strip コマンドを用いてシンボルテーブルを削除したり，検知条件となっている関数名を変更するといった攻撃者側による検知回避の対処が取られた際には有効な検知が行えないという課題がある．しかし，関数が呼び出すシステムコールの呼び出し順番は関数名の名称を変更しただけでは変化しない．strace と呼ばれる動作しているプロセスから呼び出されているシステムコールを追跡するコマンドを用いて，Mirai マルウェアのプログラムにおいて特徴的な動作を実装した内部関数に着目しこの関数から呼び出されるシステムコールの系列を用いた検知を行うことによって検知回避の対処がなされた場合でも検知が可能になる．

4.2　Mirai の特徴的な動作に基づく検知条件知手法

Mirai マルウェアは特徴的な動作として，サーバーに DDoS 攻撃を行う動作の他にインターネットに公開されているホストに対して新たな侵入先を見つけるために telnet ログインが可能な端末をスキャンする活動を行っている．また，Mirai はサーバに DDoS 攻撃を行うプロセスと telnet ログインが可能な端末をスキャンする活動のプロセスは独立して動作しているため，プロセスは別々に存在している．DDoS 攻撃を行うプロセスは DoS 攻撃を行っている場合や，攻撃命令を待機している挙動など動的解析を行うタイミングによって，異なった解析結果が得られることが考えられる．しかし，スキャン活動を行うプロセスはログインできる端末を探索している挙動が主なため動的解析を任意のタイミングで行っても，共通している解析結果が得られると考えられる．DDoS 攻撃を行うためのプロセスとスキャン活動を行っているプロセスについて strace を用いてシステムコールを追跡したところ，DoS 攻撃を行うプロセスは攻撃命令を待機している状態になるまでに呼び出されるシステムコールは様々なものがあった．しかし，スキャン活動を行っているプロセスは sendto と呼ばれるソケットへメッセージを送るシステムコールを連続して呼び出していた．そのため，任意のタイミングで strace をおこないシステムコールを追跡しても同様の結果を得ることができる．なので，スキャン活動を行うプロセスに着目をしスキャン活動が呼び出すシステムコールの系列を用いた検知を行う．スキャン活動を行うプロセスのシステムコールの実行状況を追跡したところ sendto を連続して呼び出しており，sendto によって送信されるメッセージの宛先アドレスが呼び出しごとに異なったアドレスであること，送信先のポートが 23 であったことからこのシステムコールを検知に用いる特徴とする．検知条件として 3 つの条件を定める．

1. sendto のシステムコールが 2 回以上連続して呼び出されていること
2. sendto によって送信先のポートが 23 であること
3. sendto によって送信されるメッセージの宛先アドレスが呼び出しごとに異なったアドレスであること

4.3 誤検知の可能性

前章で述べた検知条件をもとにマルウェア探索を行った際に，誤検知する場合として，以下の 3 つが考えられる

1. IoT デバイス上で sendto の呼び出しが多いプログラムの実行
2. IoT デバイスから複数の端末に向けてメッセージを送信
3. IoT デバイスから複数の端末を遠隔操作しサーバー等の設定やログファイルを特定のサーバーへ転送

strace を用いてシステムコールを確認し上記の動作が検知条件に一致するのか確認を行った．IoT デバイス上で sendto の呼び出しが多いプログラムが実行されるプログラムとして，一定時間 sendto のみを呼び出すプログラムについて考える．sendto が呼び出されるだけのプログラムでは，検知条件に一致しやすく誤検知する可能性がある．しかし，送信先のポートが 23 であり，送信されるメッセージの宛先がすべて別の宛先アドレスである sendto が呼び続ける正規プログラムが存在するとは考えにくい．IoT デバイスから複数の端末に向けてメッセージを送る動作として考えられるものが，wall や write など IoT デバイスに telnet，ssh ログインしている端末にメッセージを送るコマンドがある．wall，write コマンドを実行してシステムコールを確認した結果が表 2 のようになる．表 2 のように sendto を呼び出すことが確認されなかったため，IoT デバイスから複数の端末に向けてメッセージを送信する場合には誤検知することがない．IoT デバイスから複数の端末を遠隔操作する方法について，ssh や telnet，parallel-ssh といったリモートシェルを用いて手動でコマンドを入力して端末を操作する場合とスクリプトファイルなどで端末を自動的に操作させる 2 種類がある．IoT デバイスから複数端末を手動でコマンドを入力してファイルの転送などを行いシステムコールを確認した結果，sendto を連続では呼び出していなかった．スクリプトファイルを利用してファイルの転送を行う場合も，同様に sendto を連続で呼び出していることを確認できなかった．sendto だけを呼び出すプログラムを telnet，ssh を使用して端末上で実行した場合，本来はシステムコールである sendto が連続で呼び出されていたものが sendto の次に wirte のシステムコールが呼び出され sendto が 2 回以上連続で呼び出されていることが確認できなかった．ssh や telnet を利用して遠隔操作を行う場合や他の端末にメッセージを送信する場合には sendto が 2 回以上連続で呼び出されていることがないため誤検知することはないと考えられるため，これらの検知条件は妥当だと考える．

4.4 straceコマンドによる監視対象のプロセスの実行速度の調査

strace コマンドによってシステムコール呼び出し履歴を監視されるプロセスの実行速度が低下することが考えられる．そのため，strace コマンドによってプログラムの実行速度の変化を調べ strace によるシステムコールの監視動作がプロセスに与える実行速度の影響を調査した．使用するプログラムとしては，農林水産研究情報総合センターが定める，よく使用される Linux コマンド [14] を使用した．cp,tar,df,ps,cat の 5 つのコマンドを対象に strace によるシステムコールの監視動作によるプロセスの実行速度の影響を調査した．実行したコマンドはは表 4 になる．

表 4.1: 実行したコマンド

cat gpl-2.0.txt
cp gpl-2.0.txt cp.txt
ps -w
df
tar -zcvf gpl-2.0.tar.gz gpl-2.0.txt

コマンドの引数にファイルが必要になる場合にはフリーソフトライセンスである GPL2 の内容が書かれているテキストファイルを使用した. 表 3 の 5 つコマンドに対しての strace コマンドを実行した場合の速度比較を行った. 実行した結果が表 5 になる．

表 4.2: 計測結果

	無為 (秒)	strace(秒)
cat	3.5730	3.6894
cp	0.0048	0.2517
ps	0.5134	10.9312
df	0.6264	0.1212
tar	0.57014	0.0294

システムコールの監視動作による速度は cat コマンドは 3%低下,cp コマンドは 5079%低下,ps コマンドは 2029%低下，df コマンドは 417%低下,tar コマンドは 1839%低下．この結果から，呼び出されるシステムコールによって速度の低下量は変化するがシステムコール監視動作によってプロセスの実行速度は低下することがわかった．システムコール監視対象となるプロセスが正常なプロセスだった場合，実行速度の低下がするため，マルウェア探索動作による 1 プロセスあたりの strace の実行回数，アタッチする時間を短くする必要がある．

4.5 システムコール呼び出し履歴を用いた検知手法の提案

計算資源が潤沢でない IoT デバイス上でも実現可能な，Mirai 亜種の動作を検知する軽量な動的解析に基づく検知システムを提案する．検知システムの概要を図 6 に示す．Mirai は telnet ログインが可能な端末を探索するスキャン活動を行う機能を持ち，システムコー

ルの一種である sendto を複数回連続で呼び出している．そこで動作しているプロセスからシステムコール呼び出し履歴を取得し，スキャン活動を行っているプロセスの動作を確認することでマルウェア感染の有無を判定する手法を以下に述べる．

1. IoT デバイス上で動作を行うプロセスのホワイトリストを作成する．
2. プロセスを監視し、作成されたホワイトリストをもとに記載がないプロセスを特定する．
3. 特定した複数のプロセスに関して，strace を 1 秒間実行し検知条件に一致したシステムコール呼び出し呼び出し履歴があるか監視する．もし検知条件に一致したシステムコール呼び出し履歴があった場合にはマルウェアだと判断を行い通知を行う．
4. 1 プロセスあたりの strace によるシステムコール監視動作の回数を減らすために，strace を実行したあとに監視動を中断する時間を設ける．マルウェアを検出するのに要する時間の最悪値を 30 秒数とし，監視動作を中断する時間は以下の式で求める．

$$\text{中断時間} = \frac{30}{\text{監視プロセス数}} - 1 \tag{4.1}$$

(4.2)

5. 複数の監視対象となるプロセスに対して strace を実行する．すべての監視対象となるプロセスに対して strace を実行した場合には 2 に戻り繰り返す．

検知システムとして前章で述べたシンボルテーブルを用いた検知システムに変更したものを利用した．

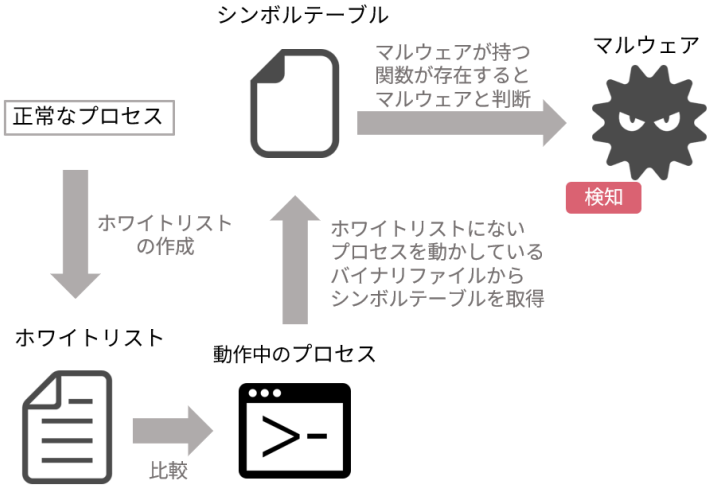


図 6　検知システムの概要

第5章　評価実験

本研究で実装した検知システムにおける定常的な動作負荷の評価を行い，DDoS 攻撃を行うマルウェアを用いて検知精度の評価を行った．

5.1　システムコール呼び出し履歴を用いた検知手法による定常的な動作負荷の評価

IoT デバイスにマルウェアが動作していない状況下において，提案した検知システムによるマルウェア探索動作によって IoT デバイス本来の動作が阻害されていないことを評価するために，3.4 節で述べたように，LinuxOS を対象とする既存のアンチウィルスソフトである Clam AV を動作させた状態の CPU，メモリの使用率を基準値とし，提案した検知システムによって Mirai が動作していない状況での，CPU，メモリの使用率について sar コマンドを用いて 1 分間計測を行った．sar コマンドを用いて得た CPU，メモリの使用率について，Clam AV と比較を行った結果が図○，○になる．

図○　IoT デバイス上でマルウェアが動作していない状況におけるマルウェア探索のメモリ使用率

図○　IoT デバイス上でマルウェアが動作していない状況におけるマルウェア探索の CPU 使用率

Clam AntiVirus を利用した場合には，平均 CPU 使用率が○%，メモリ使用率は○%となった．提案した検知手法では，平均 CPU 使用率が○%，メモリ使用率が○%となった．メモリ使用率は比較対象の Clam AntiVirus と提案した検知手法では○%減，CPU 使用率は，Clam AntiVirus に対して提案した検知手法は○%減となった

5.2　Mirai とその亜種マルウェアを対象とする判別性能評価

ハニーポットを用いて DDoS 攻撃を行うマルウェアを収集し，収集したマルウェアを検体として用いて，システムコール呼び出し履歴を用いた検知手法の検知精度の評価を行った．

5.2.1　ハニーポットによるマルウェアの収集

ハニーポットと呼ばれる攻撃者に端末を意図的に侵入させ，マルウェアをダウンロードさせ実行するまでの挙動を取得するシステムを用いて DDoS 攻撃を行うマルウェアを収集した．シェルの対話の中でダウンロードされるバイナリファイルを実行させることなく保存することが可能な Michel Oosterhof によって開発された Cowrie[15] と呼ばれるハニーポットを用いた．Cowrie によって収集されたバイナリファイルについて Virus Total と呼ばれるマルウェア検知オンラインサービスを用いて解析を行い，DDoS 攻撃を行うマルウェアの分類分けを行った．Virus Total[16] はユーザーから投稿された検体を 54 のウィルス

対策エンジンによって解析するオンラインサービスであり，投稿された検体についてマルウェアの分類を知ることができる．2019/01/09 から 2019/01/28 の期間でハニーポットを断続的に運用してバイナリファイルの収集を行った．収集したバイナリファイルを Virus Total に投稿し，Virus Total の解析結果から，Mirai または Mirai の亜種のマルウェアを DDoS 攻撃を行うマルウェアとして分類分けした．分析した結果，収集したバイナリファイルは，空ファイルのものや，マルウェアをダウンロードさせ実行させるファイル，DDoS 攻撃を行うマルウェアのバイナリファイル等が散見され，DDoS 攻撃を行うマルウェアは○○検体が存在した．

5.2.2 提案システムによるマルウェアの検知精度評価

前項で収集した OO 検体の DDoS 攻撃を行うマルウェアを用いてシステムコール呼び出し履歴を用いた検知手法の検知精度の評価を行った．入手したマルウェアをネットワークから隔離した状態で検知システムを動作させマルウェアの検知できるかどうか確認を行った．検知率と誤検知率は以下の式で求めた．

$$/centering \text{検知率} = \frac{\text{提案システムによるマルウェアの検知数}}{\text{マルウェアの検体数}} \tag{5.1}$$

$$\text{誤検知率} = \frac{\text{提案システムによってマルウェアと判断されたプログラム数}}{\text{正常なプログラム数}} \tag{5.2}$$

第6章　結論

6.1　まとめ

シンボルテーブルを用いた検知手法では，マルウェアの実行形式ファイルに含まれるシンボルテーブルを削除された場合には検知をすることができなかった．しかし，システムコール呼び出し履歴を用いた検知手法では，シンボルテーブルを削除された場合でも Mirai を検知することが可能だった．

6.2　今後の展望

スキャン活動を行っていない bashlite と呼ばれる DDoS 攻撃を行うマルウェアは検知を行う事ができないため．スキャン活動の他にも検知条件を定めてスキャン活動を行っていないマルウェアも検知できるようにする必要がある．ハニーポットを用いて集めたマルウェアは Mirai が主だったため，Mirai 亜種であるマルウェアの代表例である，Wicked，Owari，Satori，Hajime のマルウェア検体を入手することができなかったため，ハニーポットでその検体を入手して Mirai 亜種のマルウェアの検知精度を求める必要がある．攻撃者側が IoT デバイスに Telnet ログインが成功にし，マルウェアをデバイス上にダウンロードさせる際に，kill コマンドを用いて検知システムを停止させてからマルウェアを実行させた場合には，マルウェア検知をすることができない．改善策として kill コマンドによって，検知システムが停止した際にも不正侵入だとみなし，利用者に検知システムの再起動を促し，不正侵入されたこと通知を必要がある．他にも，ホワイトリストの改ざんによって，マルウェアの挙動を提案システムによって監視することができず，マルウェアの探知ができない可能性がある．

謝辞

本研究において、長期にわたる評価実験に協力いただきました、株式会社○○の△△△△様に感謝いたします.

参考文献

[1] 総務省：IoT デバイスの急速な普及 , 情報通信白書（オンライン）, 入手先 <http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nd111200.html>(参照 2018-06-17).

[2] 宮田健：IoT デバイスを狙うマルウェア 「Mirai」とは何か――その正体と対策, Tech Factry（オンライン）, 入手先<http://techfactory.itmedia.co.jp/tf/articles/1704/13/news010.html> (参照 2018-06-20).

[3] Scott Hilton:Dyn Analysis Summary Of Friday October 21 Attack,Oracle Dyn（オンライン）, 入手先<https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack>(参照 2018-06-20).

[4] 長柄啓吾, 松原豊, 青木克憲 ほか：組込みシステム向けマルウェア Mirai の攻撃性能評価，研究報告システム・アーキテクチャ，vol.2017-ARC-225,No.41,p1-6(2017)

[5] 坂野加奈, 上原哲太郎：アノマリ検知手法を用いた IoT 機器のマルウェア感染検出, 研究報告セキュリティ心理学とトラスト,vol.2018-SRT-27 No.3,p1-6 (2018)

[6] 青木一樹, 後藤滋樹：マルウェア検知のための API コールパターンの分析, 電子情報通信学会総合大会講演論文集 2014 年 情報・システム,vol.2,No.179,2014-03-04

[7] Jerry Gamblin:jgamblin/Mirai-Source-Code,GitHub（オンライン）, 入手先<https://github.com/jgamblin/Mirai-Source-Code>(参照 2018-09-20)

[8] https://www.nict.go.jp/info/topics/2018/11/07-2.html

[9] https://internet.watch.impress.co.jp/docs/news/1046239.html

[10] 総務省：IoT デバイスの急速な普及 , 情報通信白書（オンライン）, 入手先 <http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nd111200.html>(参照 2018-06-17).

[11] 坂野加奈, 上原哲太郎：アノマリ検知手法を用いた IoT 機器のマルウェア感染検出, 研究報告セキュリティ心理学とトラスト,vol.2018-SRT-27 No.3,p1-6 (2018)

[12] 坂野加奈, 上原哲太郎：アノマリ検知手法を用いた IoT 機器のマルウェア感染検出, 研究報告セキュリティ心理学とトラスト,vol.2018-SRT-27 No.3,p1-6 (2018)

[13] 坂野加奈, 上原哲太郎：アノマリ検知手法を用いた IoT 機器のマルウェア感染検出, 研究報告セキュリティ心理学とトラスト,vol.2018-SRT-27 No.3,p1-6 (2018)

[14] https://itcweb.cc.affrc.go.jp/affrit/doku.php?id=faq/tips/unix-command

[15] https://itcweb.cc.affrc.go.jp/affrit/doku.php?id=faq/tips/unix-command

[16] https://itcweb.cc.affrc.go.jp/affrit/doku.php?id=faq/tips/unix-command

図 目 次

表 目 次