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が存在するか判別することによって DDoS 攻撃を行うマルウェアの検知を行う．

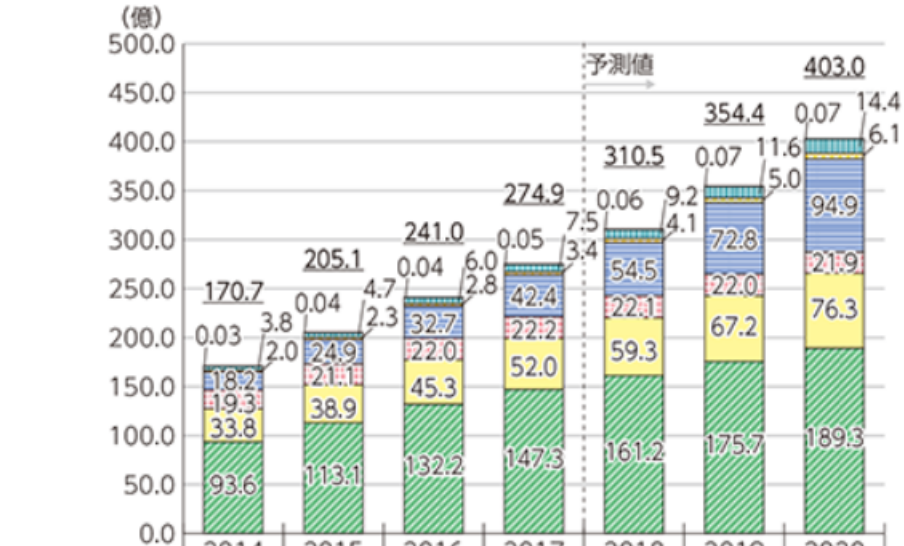
キーワード: DDoS 攻撃，IoT デバイス，マルウェア，Mirai

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　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] では，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．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．

1.2　対象とする領域

実用レベルのサイズのプログラムを作成するためのプログラミング言語について研究する．ここで，行うのは 3 次元グラフィックス向けの言語の設計とそのインタプリタの実装である．

1.3　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．

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

組込みシステム向けマルウェア Mirai の攻撃性能評価 IoT 機器への Telnet を用いたサイバー攻撃の分析 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

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についてハニーポットを用いて，IoT デバイスを対象としたマルウェアの観測を行い，入力されているコマンドは通常の IoT 機器とは大きく異なることが報告された．

2.2 関連技術

2.2.1 Sophos Antivirus for Linux

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，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　アプローチ

事前調査として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　事前調査

事前調査として Web 上で公開されている Mirai のソースコード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 に占めるコマンド列を得た．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．

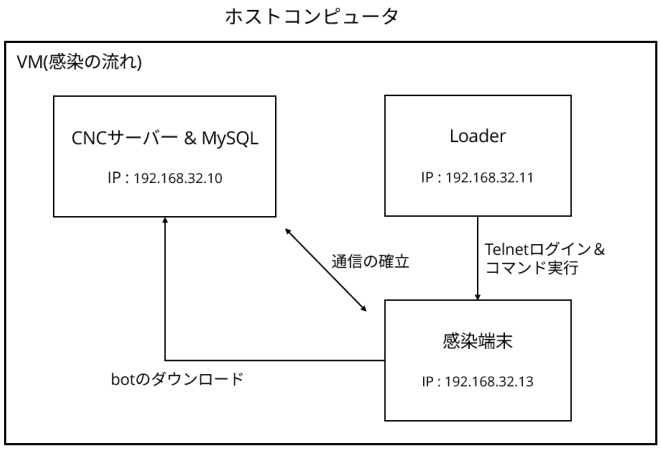


図 2　Mirai の解析環境

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．そこで動作しているプロセスの起動に用いられたバイナリファイルのシンボルテーブルから特徴を抽出し，その特徴を持つ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．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．

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．

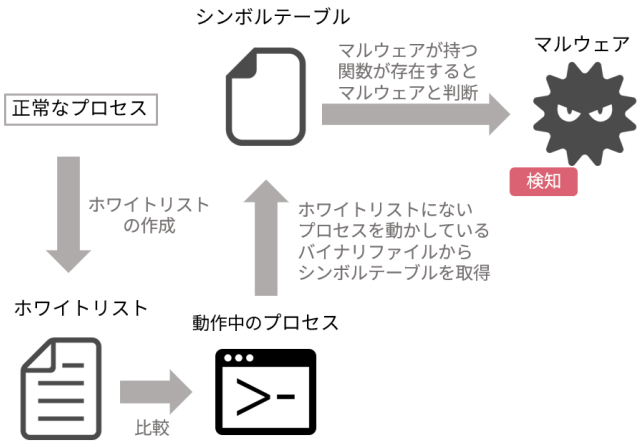


図 3　検知システムの概要

3.4　シンボルテーブルを用いた検知手法の評価

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。の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上智あの 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咲いている際には，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支障をきたしてはならない．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．

3.4.1　提案した検知手法が動作時にシステムに与える負荷による評価

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．

3.4.2　評価結果と検知における制約事項

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．

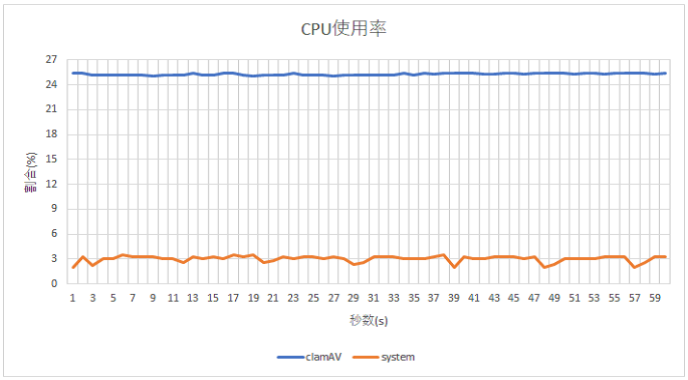


図 4　IoT デバイス上でマルウェアが動作していない状況におけるマルウェア探索のめもり使用率

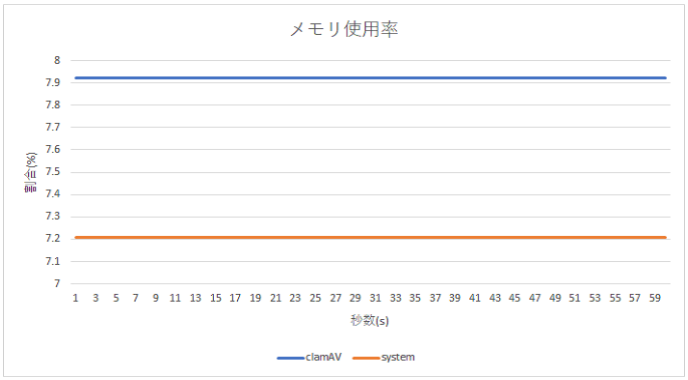


図 5　IoT デバイス上でマルウェアが動作していない状況におけるマルウェア探索の CPU 使用率

第4章　システムコール呼び出し履歴を用いた検知手法

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，攻撃を行うための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おり，同じ動作を繰り返していた．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 システムコール呼出履歴を用いた検知手法のシステム概要

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

とその亜種である Owari を含めて調査したところ，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，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発見する．
3. ホワイトリストにないプロセスに関して，strace を使用しシステムコール呼び出し履歴を取得し，検知条件に一致するプロセスがあるか確認を行う．
4. 検知条件に一致したプロセスを発見した場合に，マルウェアだと判断を行う．

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．

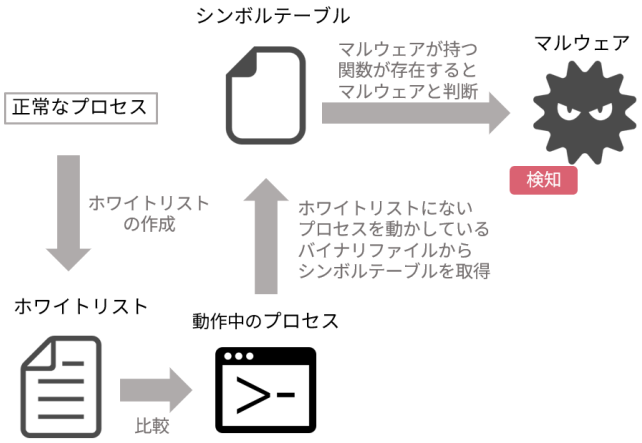


図 6　検知システムの概要

4.5　システムコール呼び出し履歴を用いた検知手法の評価

シンボルテーブルを用いた検知手法と同様の評価を行う．

第5章　考察

5.1　評価結果

Java 言語との比較では，惨敗であり，FUN は 2 倍の記述量を必要とした．しかし，これは，Java のもつパッケージ IKURA が非常に強力であるためで，同一機能をもつライブラリを用意することにより，FUN にも同様の能力を持たせることができることが判明した．

5.2　評価結果

Java 言語との比較では，惨敗であり，FUN は 2 倍の記述量を必要とした．しかし，これは，Java のもつパッケージ IKURA が非常に強力であるためで，同一機能をもつライブラリを用意することにより，FUN にも同様の能力を持たせることができることが判明した．

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)

付録その1

付録その 1(プログラムのソースリストなど) を必要があれば載せる

付録その2

付録その 2(関連資料など) を必要があれば載せる

図 目 次

表 目 次