Computer Security Symposium 2018
22 - 25 October 2018

RISC-Vにおけるプロセッサ情報を用いた動的なアノマリ検知機構

鈴木 庸介[1]　小林 良太郎[1]　加藤 雅彦[2]

概要：近年，IoT 機器の普及によって日常生活の利便性が向上している一方で，IoT 機器を狙ったマルウェアも発生しており，IoT 機器のセキュリティ対策が求められている．しかし，多くの IoT 機器はログインパスワードの設定などがデフォルトのまま使われていることが多いため，マルウェアのパスワードクラックによって簡単に IoT 機器に侵入されている．そこで，パスワードクラックの検知機構によりマルウェアの感染後の活動を防ぐシステムを作ることを目指す．また，ライセンスフリーであり，複数のベンダーが開発に参加していることから，今後 IoT デバイスへの搭載が期待される，RISC-V に注目する．本研究では，プロセッサに RISC-V を使用した仮想マシンを用いて，エミュレーション環境を用意し，プロセッサ情報を用いた，パスワードクラック検知機構を評価した．実験の結果，パスワードクラックの検知が可能であることを確認した．

キーワード：RISC-V，機械学習，パスワードクラック

1. はじめに

近年，あらゆるものがネットワークを通じて相互に繋がる IoT(Internet of Things) が発達してきている．IoT 機器は急速に増え続け，2020 年には 300 億個にも上る見通しである [1]．この IoT 機器の普及によって，様々なものの情報をデータ化することができる．この集めたデータを元に，産業の自動化や，サービスの新たな付加価値に繋がることが期待されている．一方で，IoT 機器のセキュリティについては未だ課題が多く残されており，現存する IoT 機器の 70%には脆弱性が残されているといわれている [2]．

IoT 機器に感染するマルウェアが引き起こしたセキュリティインシデントも既に発生している．2016 年には，IoT 機器に感染するマルウェア「Mirai」によって構築されたボットネットによる大規模な DDoS 攻撃によって，米 DNS プロバイダ Dyn が大きな被害を受けた [3]．この攻撃により Amazon や Twitter に代表される多くのサービスが一時的に利用不可能となった．このような被害を増やさないためにも，IoT 機器のセキュリティ対策は急務となっている．

「Mirai」などの IoT 機器に感染するマルウェアでは，はじめに基本的な辞書攻撃を行い，IoT 機器に侵入を試みようとする．そして，マルウェアは IoT 機器に侵入後，IoT 機器内で活動を行う．そこで我々は，IoT 機器に侵入しようとする時点で，パスワードクラックを検知することができれば，その後のマルウェアの活動を防ぐことができると考えた．

現在，既存のパスワードクラックへの対策として，初期パスワードの変更，ssh などのポート番号の変更，ssh での root ログインの禁止，ログイン試行回数の制限など様々な方法がある．これらにおいて，ssh などのポート番号の変更，ssh での root ログインの禁止，ログイン試行回数の制限は製造者が IoT 機器の製造時に設定することができる．しかし，初期パスワードの変更は各ユーザが行う必要があるが，多くのユーザはわざわざ IoT 機器のパスワードを初期のものから変更しようとしない．そのため，初期パスワードのまま使用されている多くの IoT 機器は，「Mirai」などのマルウェアのパスワードクラックによって，簡単に侵入され，感染してしまう．

このように，ユーザはパスワード変更などの IoT 機器の設定を行おうとしないため，我々は，初めからそのような対策が行われており，ユーザが予め設定を行う必要のない新たな対策方法を考える．そして，プロセッサ情報を特徴量とした機械学習によるパスワードクラックを検知できる判別機構を提案する．プロセッサ情報とは，オペコードや，プログラムカウンタといった情報の他，レジスタ番号

[1] 工学院大学 Kogakuin University 1-24-2, Nishi-shinjuku, Shinjuku-ku, Tokyo, Japan
[2] 長崎県立大学 University of Nagasaki 123, Kawashimo-chou, Sasebo-shi, Nagasaki, Japan

やレジスタ値の情報などが含まれる．本稿では，プロセッサに RISC-V を使用した仮想マシンを用いて，プロトタイプを実装し，提案手法の評価を行う．RISC-V とは，RISC アーキテクチャの第 5 世代の完全にライセンスフリーなコンピュータの命令セット (ISA) のことである [4]．ただ，プロセッサに RISC-V を使った IoT 機器はまだ存在していない．しかし，RISC-V はライセンスがフリーであることから，検知機構の付与が容易であるため，研究用として使いやすい．また，RISC-V は Google や NVIDIA などが開発に参加していることから，今後 IoT デバイスに搭載されていくと期待されるため，本研究では先行して研究に取り入れている．

第 2 章では関連研究について，第 3 章では提案機構のメインアイデアについて述べる．第 4 章では提案機構の実装方法について説明し，第 5 章で評価を行う．第 6 章で評価結果を示し，第 7 章で考察を行い，第 8 章で本論文をまとめる．

2. 関連研究

機械学習を用いてマルウェアを，発見する手法が既にいくつか提案されている．Arvind らは，Android に対するマルウェアが実行時に要求する権限に注目し，それを特徴量とすることでマルウェアを検知する手法を提案している [6]．また複数の機械学習手法による分類精度の比較も行っている．

村上らは，プログラムを動的解析して API コールを抽出し，その名前の n-gram を用いて，マルウェアと正常プログラムの分類を行っている [7]．n-gram とは，隣り合う N 個の単語を一塊として文章を分解する手法である．さらに，評価用のデータを学習用のデータとの類似度に基づいて選別し，選別後の評価データの検出精度を向上させる手法についても提案している．

また，マルウェアの静的解析によって抽出した特徴を用いて，機械学習によってマルウェアを分類する方法も広く行われている [8][9]．

これらの機械学習を用いた手法では，API コールや権限を監視しており，ソフトウェアとしてマシン本体に実装する必要がある．これらの手法は一般のマシンには実装可能だが，マシンリソースの問題により IoT 機器に実装することは難しいという問題がある．そこで，提案手法では，ソフトウェア的な情報ではなくハードウェアから取得可能な情報を用いてプログラムの特徴を抽出することで，ビヘイビア法をハードウェアにオフロードし，リソースの競合の問題を解決する．

セキュリティ機構をハードウェアとして実装する手法としては ARM プロセッサに実装されている TrustZone が挙げられる [10]．TrustZone は実行環境を Secure World と Normal World の 2 つに分けている．Normal World から Secure World へのアクセスは制限されるため，Secure World 上の認証情報や支払い情報をマルウェアから守ることが可能となる．また，TrustZone を利用したセキュリティ機構の研究も行われている [11][12]．この手法は，マルウェアから，重要なデータを守るための機構であり，マルウェア実行の検知を目的とした本研究とは異なる．

特定の処理を専用のハードウェアにオフロードすることで，高速な処理を実現する製品も存在する．SmartNIC はネットワークのパケットフィルタリングや暗号化などといった処理を，専用ハードウェアによって処理する技術である [13]．これによって CPU のリソースを他のアプリケーションに充てることが可能となる．また，OpenSSL の暗号化について FPGA ベースによる，専用ハードによってパフォーマンスの向上を図る研究も行われている [14][15]．

大谷と高瀬らは，ソフトウェアシュミレータを用いて，プロセッサが扱うプログラムカウンタやレジスタ値などの情報を特徴量として，機械学習を行い，亜種マルウェアの検知やプログラム分類の予備評価を行っている [16][17]．

3. 提案手法

この章では，今回作成するプロトタイプの概要と使用したプロセッサ情報の詳細について述べる．

3.1 メインアイデア

図 1 に提案機構の概要図を示す．今回作成するプロトタイプでは，本物のプロセッサの代わりに仮想マシンを使用する．

仮想マシンに対してリモートで通信を行い，その時のプロセッサ情報を取得する．その後，取得したプロセッサ情報を特徴量とした機械学習によって，正常アクセスとパスワードクラックの判別を行う．

本稿で使用するプロセッサ情報とは，プロセッサの内部情報を指す．このプロセッサ情報には，静的なプロセッサ情報と，動的なプロセッサ情報が存在する．静的なプロセッサ情報とは，プログラムカウンタやオペコード，レジスタ番号など，バイナリから解析可能な情報を指す．動的なプロセッサ情報とは，レジスタ値など，命令実行時のプロセッサの内部状態を示す情報である．この動的なプロセッサ情報は，実際にプロセッサを動作させないと取得することができない．プログラムの学習，判別にはプロセッサ情報を命令実行順に並べた情報を使用する．この情報をトレースと表記する．トレースを通信時の固有のデータとして学習や判別に利用する．

今回作成するプロトタイプでは，本物のプロセッサの代わりに仮想マシンを使用する．

3.2 プロセッサ情報の概要

表 1 に特徴量として使用したプロセッサ情報を示す．

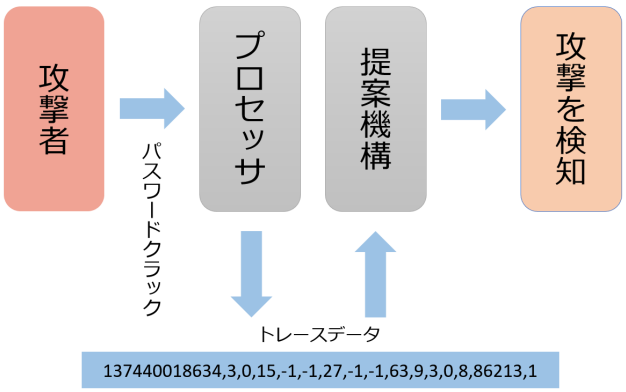

図 1 提案機構の概要

表 1 特徴量として使用したプロセッサ情報

プロセッサ情報	説明
pc	プログラムカウンタ
insn	命令種別
addr	分岐予測先のプログラムカウンタ
rn1 rn2 rn3	レジスタ番号
rv1 rv2 rv3	レジスタ値
dn	NOP の命令距離
dl	LD の命令距離
ds	ST の命令距離
dj	JUMP の命令距離
dca	CALC の命令距離
dcs	CSR の命令距離
do	OTHER の命令距離

表 1 における，insn，addr，rn1，rn2，rn3，dn から do は静的なプロセッサ情報である．また pc，rv1，rv2，rv3 は動的なプロセッサ情報である．静的なプロセッサ情報は QEMU[5] から直接取得できる情報と，そこから計算した情報に分けられる．QEMU とは，フリーの CPU エミュレータであり，本研究では仮想マシンの構築に使用する．insn，addr，rn1，rn2，rn3 は QEMU から直接取得できる情報である．insn は NOP，LOAD，STORE，JUMP，CALC，CSR，OTHER の 7 種類の命令種別を表す．NOP は no operation，LOAD はロード命令，STORE はストア命令，JUMP は条件ジャンプ命令，あるいは，無条件ジャンプ命令，CALC は演算命令，CSR はシステムレジスタを読み書きする命令，OTHER はそれら以外の命令である．dn から do は，QEMU から得られた情報から計算した情報であり命令の距離を表す．これらは，NOP，LOAD，STORE，JUMP，CALC，CSR，OTHER の 7 種類の命令が最後に実行されてから何命令経過したかをあらわしており，それぞれ dn，dl，ds，dj，dca，dcs，do に対応する．

図 1 の下部に示されたトレースデータは，プロセッサ情報の一例であり，pc から順にカンマ区切りですべての情報が示されている．また，機械学習のために，値は全て 10 進数で表している．具体的には，pc や addr の値は 16 進数から 10 進数に変換し，insn では，ラベルを数字に変換しており，NOP，LOAD，STORE，JUMP，CALC，CSR，OTHER はそれぞれ 0 から 6 に対応している．同じように，rn1，rn2，rn3 でも，レジスタ番号のラベルを数字に変換している．また，情報がない部分や初期値には-1 を出力させている．例えば，図 1 の下部に示されたトレースデータの insn は 3 であるので，JUMP 命令である．また，JUMP 命令が実行されているので，JUMP 命令間の距離を表す dj は 0 にリセットされ，その他の命令間の距離を表す dn，dl，ds，dca，dcs，do もそれぞれ値がカウントされている．

4. 実装

この章では，プロトタイプの実装について述べる．今回作成したプロトタイプは，大きく分けて以下の 2 つの要素から構成される．

- 仮想マシン
- 判別器

仮想マシン部で取得できる，静的なプロセッサ情報と動的なプロセッサ情報から構成されるトレースをトレースデータとする．学習・判別に用いるのは，このトレースデータである．

4.1 仮想マシン

仮想マシンにリモートで通信を行い，その時のトレースを取得する．仮想マシンにはオープンソースのエミュレータである QEMU を使用した．QEMU の “-d in_asm ” オプションを利用することで QEMU 上で実行されている命令のアセンブリコードを取得できる．しかし，既存のデバッグ機能では，CPU のレジスタ値などの情報を取得することができない．そこで QEMU のソースコードを改変し，CPU のレジスタ値などの情報を取得できるようにすると共に，データとして扱いやすいようにログの取得部分を改変した．

また仮想マシンによるトレースの取得は Python のプログラムによって管理される．仮想マシン動作中に，外部の Python のプログラムから動作開始のシグナルを送信することによってトレースの取得が開始される．そしてトレースデータが指定した命令数に達した時または仮想マシンのウィンドウを閉じた時に，プロセッサ情報の取得が終了する．

4.2 判別器

判別器では，機械学習によって，あるトレースデータが正常アクセスのものであるか，パスワードクラックのもの

表 2 評価環境

エミュレータ	qemu-2.12.0
プロセッサ	64 ビット RISC-V
OS	Fedora 28 (Rawhide)

であるかを判別する．機械学習のアルゴリズムにはランダムフォレストを選択した．ランダムフォレストは，複数の決定木を組み合わせて，各決定木の予測結果の多数決によって結果を得る，アンサンブル学習の一種である．高次元のデータ分類においても効率的に動作し，高次元の特徴においても効率的な学習が可能であるという特徴を持つ．プロセッサ情報にはカテゴリ値が多いが，このような場合でもスケーリング不要である点からランダムフォレストが適していると考えた．

判別器の動作は，学習フェーズと判別フェーズに分けられる．学習フェーズでは，あらかじめ取得した正常アクセスの学習用トレースと，パスワードクラックの学習用トレースを使用し，判別器を作成する．判別フェーズでは判別を行いたいプログラムの判別用トレースを判別器に与える．判別用トレースは，1 プロセッサ情報ごとに，正常であるか，攻撃 (パスワードクラック) であるか判別される．判別用トレース内にはプロセッサ情報が多く存在するが，1 命令が 1 プロセッサ情報に対応している．最終的に 1 プログラムのトレースデータ全体のうち，正常アクセスと判別された割合と，パスワードクラックと判別された割合からなる中間値が出力される．正常アクセスと判別された割合を Normal，パスワードクラックと判別された割合を Attack とする．Attack が一定の閾値以上であれば，通信はパスワードクラックであると判別し，そうでなければ正常アクセスであると判別する．

5. 評価

この章では評価方法及び，評価結果について述べる．

5.1 評価環境

評価環境を表 2 に示す．エミュレータには RISC-V に対応している QEMU-2.12.0 を使い，プロセッサには 64 ビットの RISC-V，OS には Fedora 28 を使用した．

5.2 トレースの取得

正常アクセスとして，リモートでログインを行う ssh 通信とリモートでファイルの送信を行う scp 通信の 2 種類を使用している．ssh 通信は，通信開始からログインが完了するまでを動作の区切りとしてトレースを取得する．scp 通信は，通信開始からファイル送信が完了するまでを動作の区切りとしてトレースを取得する．仮想マシンの電源を ON にした後，リモート通信が可能になった時点で，トレースの取得を開始し，被攻撃マシンに対して通信を行う．正常アクセスの動作が終了した時点でトレースの取得を終了する．

表 3 正常アクセスの判別用データ

通常アクセス		命令数	サイズ	学習
通信	失敗回数	[K]	[Mbyte]	
ssh ログイン	0	285	21.7	○
		244	18.2	
ssh ログイン	1	251	18.8	
		319	24.3	
ssh ログイン	2	325	24.8	
		244	18.2	
ssh ログイン	3	48	3.7	
		40	3.1	
scp 通信	0	329	25.1	
		221	16.6	

パスワードクラックを行うにあたって，hydra，medusa，Metasploit，ncrack の 4 種類のパスワードクラックツールの辞書攻撃を使用した．その際，辞書攻撃に使うパスワードリストとして，以下 3 つのファイルを使用した．

- 72 個のパスワード（正解あり）
- 72 個のパスワード（正解なし）
- 1 個のパスワード（正解あり）

正常アクセスと同様に仮想マシンの電源を ON にした後，リモート通信が可能になった時点で，トレースの取得を開始し，被攻撃マシンに対してパスワードクラックツールの辞書攻撃を実行する．パスワードクラックツールの辞書攻撃が終了した時点でトレースの取得を終了する．

トレースを取得する際に，同じような操作の場合でも，外部割込み等の影響で，トレースの形が異なる．そこで，判別用データは，同じ手順で，複数取得した．

正常アクセスの判別用データを**表 3** に，パスワードクラックの判別用データを**表 4** にそれぞれ示す．表 3 と表 4 の “命令数 ” と “サイズ ” は，それぞれトレースデータに含まれている命令数とファイルサイズを表している．また，表 3 と表 4 の “学習 ” について，該当するトレースデータを機械学習に使用したことを表している．

5.3 評価方法

用意したトレースデータの中から，学習用トレースを選択し，学習を行い，判別器を作成する．

表 3 と表 4 から，正常アクセスとパスワードクラックの学習用データをそれぞれ選択した．正常アクセスの学習用トレースには，パスワード認証を一度も失敗せずに ssh ログインをした時のトレースデータ 1 つを使用する．パスワードクラックの学習用トレースは，hydra，medusa，Metasploit，ncrack の 4 種類のパスワードクラックツールの辞書攻撃を実行した時のトレースデータ 4 つを使用する．また，辞書攻撃には，正解が含まれている 72 個のパ

表 4 パスワードクラックの判別用データ

パスワードクラック			命令数	サイズ	学習
ツール	リスト	正解	[K]	[Mbyte]	
hydra	72	あり	64	4.8	○
	72	なし	40	3.0	
	1	あり	60	4.5	
medusa	72	あり	63	4.7	○
	72	なし	55	2.9	
	1	あり	39	4.1	
Metasploit	72	あり	63	4.8	○
	72	なし	42	3.2	
	1	あり	58	4.4	
ncrack	72	あり	63	4.8	○
	72	なし	41	3.1	
	1	あり	57	4.3	

表 5 正常アクセスの判別結果

通常アクセス		中間値 [%]		判別結果
通信	失敗回数	Normal	Attack	
ssh ログイン	0	99.4	0.6	Normal
		51.4	48.6	Normal
ssh ログイン	1	50.3	49.7	Normal
		82.3	17.7	Normal
ssh ログイン	2	79.2	20.8	Normal
		51.5	48.5	Normal
ssh ログイン	3	0.4	99.6	Attack
		1.6	98.4	Attack
scp 通信	0	88.1	11.9	Normal
		60.4	39.6	Normal

スワードファイルを使う．

その後，用意したトレースデータに対して，判別器を使って，判別を行い，中間値における Attack の値が閾値以上であれば，そのトレースデータはパスワードクラック，閾値未満であれば正常アクセスと判別する．また，本評価では閾値を 50%とする．

6. 結果

正常アクセスの判別結果を**表 5** に，パスワードクラックの判別結果を**表 6** にそれぞれ示す．正常アクセスは各パターンでそれぞれ 2 回ずつトレースデータを取得し，判別を行った．判別結果のスコアは小数点第二位を四捨五入し，小数点第一位までの値を表示させている．

表 5 の “失敗回数 ” については，ログイン試行時の失敗回数を表している．

表 6 のパスワードクラックの “ツール ”については，使っているパスワードクラックツールの名前を表している．また，“リスト ” と “正解 ” については，パスワードファイルのパスワード数とそのファイルにパスワードの正解が含まれているかどうかを表している．

表 5 から，判別用トレースについて，ssh ログインの失敗回数が 3 回であるトレースデータ 2 つを除いて，中間値

表 6 パスワードクラックの判別結果

パスワードクラック			中間値 [%]		判別結果
ツール	リスト	正解	Normal	Attack	
hydra	72	あり	0.1	99.9	Attack
	72	なし	0.1	99.9	Attack
	1	あり	0.6	99.4	Attack
medusa	72	あり	0.0	100.0	Attack
	72	なし	0.0	100.0	Attack
	1	あり	3.8	96.2	Attack
Metasploit	72	あり	0.2	99.8	Attack
	72	なし	0.0	100.0	Attack
	1	あり	2.5	97.5	Attack
ncrack	72	あり	0.0	100.0	Attack
	72	なし	0.2	99.8	Attack
	1	あり	1.4	98.6	Attack

(Attack) が閾値（既に述べた通り本稿では 50%に設定した）を下回っているため，正常アクセスと判別できていることが確認できる．ただ，ssh ログインの失敗回数が 3 回であるトレースデータ 2 つは中間値 (Attack) が閾値を上回っているため，パスワードクラックと判別される．

表 6 から，判別用トレースについて，中間値 (Attack) が閾値を上回っているため，全てのトレースデータがパスワードクラックと判別できていることが確認できる．

7. 考察

まず，表 5 の正常アクセスの ssh ログインの失敗回数が 3 回であるトレースデータ 2 つの判別結果について述べる．これらのトレースデータの中間値 (Attack) はそれぞれ 99.6%，98.4%となり，パスワードクラックと判別されている．これらのトレースデータでは，ssh ログインでパスワード認証を 3 回失敗し，ssh 通信が強制的に切断されるという動作になっている．よって，一度も ssh ログインが成功せずに通信が終了していることになる．一方で，判別器の作成時に使用した学習用データの動作について，通常用データは，ssh ログインでパスワード認証が成功したもの，攻撃用データは，パスワードクラックによって，パスワード認証での失敗を繰り返したものとなっている．したがって，ssh ログインの失敗回数が 3 回であるトレースデータはパスワード認証の失敗を繰り返すパスワードクラックの動作に近いものであることから，パスワードクラックと判別されるようになると考えられる．

次に，機械学習における，特徴量の重要度と判別結果の関係について述べる．表 1 の特徴量から，機械学習時に得られる特徴量の重要度を表示させたところ，**表 7** のようになった．重要度の値は小数点第 4 位を四捨五入し，小数点第 3 位まで表示している．

表 7 から，dn の重要度の値が最も高いことが分かる．dn は NOP 命令が実行されてから何命令経過したかを表しているため，通常アクセスとパスワードクラックの学習

表 7 プロセッサ情報の重要度

プロセッサ情報	重要度
dn	0.473
dcs	0.292
pc	0.126
rv1	0.038
ds	0.011
dj	0.010
rv2	0.010
rn1	0.009
dca	0.007
do	0.007
dl	0.007
rn2	0.004
addr	0.002
insn	0.002
rn3	0.001
rv3	0.001

用データで，NOP 命令が実行される頻度などに差があると考えた．そこで，通常アクセスとパスワードクラックの学習用データのファイルの中身の解析を行った．すると，通常アクセスの学習用データでは，NOP 命令が多く実行されているのに対し，パスワードクラックの学習用データ 4 つでは，それぞれ NOP 命令がほとんど実行されていなかった．NOP 命令の実行回数は，パスワードクラックツールの medusa を使った学習用データでは 1 回，その他 3 つのパスワードクラックツールを使った学習用データでは 0 回だった．そのため，パスワードクラックの学習用データで，dn の値は初期値の-1 を取ることがほとんどであった．よって，特徴量の重要度から，通常アクセスとパスワードクラックの判別に NOP 命令の有無が大きく影響していることがわかった．

続いて，判別結果と NOP 命令の関係について述べる．パスワードクラックと判別されている，表 5 の ssh ログインの失敗回数が 3 回のトレースデータ 2 つについて，ファイルの中身の解析を行ったところ，1 度も NOP 命令は実行されていなかった．よって，NOP 命令の実行回数がパスワードクラックの学習用データの特徴と似ているため，パスワードクラックと判別されていると考えられる．また，通常アクセスの中間値 (Nomarl) が 50%から 60%であるトレースデータについて，ファイルの中身の解析を行ったところ，初めて NOP 命令が実行されたのがいずれも 10 万命令以降だったので，それまでの約 10 万命令は，dn の値が初期値の-1 であった．一方で，通常アクセスの学習用データや通常アクセスの Nomarl の値が高いトレースデータでは，初めて NOP 命令が実行されたのが 2 万命令以降だったので，それまでの約 2 万命令は，dn の値が初期値の-1 であった．よって，これらの 2 パターンのトレースデータで，dn の値が初期値の-1 である回数が違うことになる．したがって，これらの違いが判別結果に影響を及ぼしていると思われる．

結果として，機械学習を行ったことにより，通常アクセスとパスワードクラックの判別には NOP 命令が大きく関係しているという特徴がわかった．

8. まとめ

IoT デバイス上のプロセッサからの情報を入力とし，正常アクセスとパスワードクラックを判別する手法を提案した．また判別にはプロセッサの情報を特徴量とした機械学習を用いた．提案機構の評価のため，仮想マシンを用いてエミュレートし，検証した結果，正常アクセスを正しく判別することが出来た．パスワードクラックについても同様に正しく判別することができた．今後は，FPGA を使った，より現実のハードウェアに近い状況でのシミュレーションや，実際のハードウェアへの実装を検討する．

謝辞 本研究の一部は，JSPS 科研費 17K00076, 16K00071 の支援により行った．

参考文献

[1] 総務省, 平成 29 年版情報通信白書第 1 部第 3 章 IoT 化する情報処理産業, http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/pdf/n3300000.pdf (Accessed 2018-08-16).
[2] P. Maki, S. Rauti, S. Hosseinzadeh, L. Koivunen, and V. Leppanen. Interface diversification in IoT operating systems, UCC '16: Proceedings of the 9th International Conference on Utility and Cloud Computing, pp.304-309, 2016.
[3] J. A. Jerkins. Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp.1-5, 2017.
[4] RISC-V, https://riscv.org/ (Accessed 2018-08-09).
[5] QEMU. https://www.qemu.org/ (Accessed 2018-08-09).
[6] A. Mahindru and P. Singh. Dynamic Permissions based Android Malware Detection using Machine Learning Techniques, Proceedings of the 10th Innovations in Software Engineering Conference, pp.202-210, 2017.
[7] 村上純一, 鵜飼裕司. 類似度に基づいた評価データの選別によるマルウェア検知精度の向上, コンピュータセキュリティシンポジウム 2013 論文集 2013(4), pp.870-876, 2013.
[8] F. Adkins, L. Jones, M. Carlisle, and J. Upchurch. Heuristic malware detection via basic block comparison, 2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE), pp.11-18, 2013.
[9] P. Khodamoradi, M. Fazlali, F. Mardukhi, and M. Nosrati. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms, 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), pp.1-6, 2015.
[10] TrustZone - Arm Developer, https://developer.arm.com/technologies/trustzone (Accessed 2018-01-29).
[11] A. M. Azab, P. Ning, J.Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen. Hypervision Across

Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp.90-102, 2014.
[12] L. Guan, P. Liu, X. Xing, X. Ge, S. Zhang, M. Yu, and T. Jaeger. TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone, Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, pp.488-501, 2017.
[13] G. Sabin and M. Rashti. Security offload using the SmartNIC, A programmable 10 Gbps ethernet NIC, 2015 National Aerospace and Electronics Conference (NAECON), pp.273-276, 2015.
[14] A. Thiruneelakandan and T. Thirumurugan. An approach towards improved cyber security by hardware acceleration of OpenSSL cryptographic functions, 2011 International Conference on Electronics, Communication and Computing Technologies, pp.13-16, 2011.
[15] J. K. T. Chang, S. Liu, J. L. Gaudiot, and C. Liu. Hardware-assisted security mechanism: The acceleration of cryptographic operations with low hardware cost, International Performance Computing and Communications Conference, pp.327-328, 2010.
[16] 大谷元輝, 高瀬誉, 小林良太郎, 加藤雅彦, プロセッサレベルの特徴量に着目した亜種マルウェアの検知情報処理学会研究報告 Vol.2018-CSEC-80, No.31, pp.1-8, 2018.
[17] 小林良太郎, 高瀬誉, 大谷元輝, 大村廉, 加藤雅彦, 機械学習を用いたプロセッサレベルでのプログラム分類に関する予備評価電子情報通信学会信学技報 ICSS, Vol.117, No.316, pp.5-10, 2017.