

ソフトウェア解析とベンダイインタビューによる IoT機器のセキュリティに関する大規模実態調査

白石 周碁¹ 福本 淳文¹ 吉元 亮太¹ 塩治 榮太朗² 秋山 満昭² 山内 利宏^{1,3}

概要: IoT 機器の利用拡大に伴い, IoT 機器を標的とした攻撃による被害が深刻化している. 今後は機器が持つ脆弱性を利用するなど, さらなる攻撃の高度化が予想される. このため, 我々は IoT 機器のファームウェア解析に基づいた予備調査を実施し, IoT 機器で利用されるセキュリティ機構とソフトウェアのアップデート状況を明らかにした. しかし, 予備調査は調査対象が限定的であり, 結果としては不十分なものであった. 本研究では, IoT 機器の調査をさらに大規模かつ体系的に実施し, 調査によって判明したセキュリティ上の問題点の要因を明らかにするために IoT 機器のベンダにインタビューを実施した. また, 予備調査からの新たな調査項目として, カーネルへの攻撃に対するセキュリティ機能, LSM ベースのセキュリティ機能を追加し, 新たな調査対象としてファームウェアだけでなく, GPL ソースコードを追加した. さらに, 体系的な調査を実施するための新たな調査手法を確立した. 本稿では, 確立した調査手法について説明し, 調査によって得られた結果, およびベンダからのインタビュー結果について述べる.

Large-scale Survey on Secure Development of IoT Devices by Software Analysis and Vendor Interview

SHUGO SHIRAISHI¹ AKIFUMI FUKUMOTO¹ RYOTA YOSHIMOTO¹ EITARO SHIOJI² MITSUAKI AKIYAMA²
TOSHIHIRO YAMAUCHI^{1,3}

Abstract: The damage caused by attacks targeting IoT devices becomes more serious. It is expected that attacks will become more sophisticated in the future. Therefore, we conducted a preliminary investigation based on the firmware analysis of IoT devices and clarified the update status of the security mechanism and software. However, the preliminary survey was limited, and the result was insufficient. In this research, we conducted a large-scale and systematic survey of IoT devices and conducted an interview with the vendors of IoT devices to clarify the factors of the security problems found by the survey. In addition, we added KASLR and LSM-based security functions as new survey items, and added GPL source code as a new survey target. Furthermore, we established a new survey method for conducting a systematic survey. In this paper, we explain the processing flow of the program created when conducting these surveys and the survey method. Finally, we describe the results actually obtained by the survey.

1. はじめに

様々な機器をインターネットに接続することにより, 新たな価値を創出する IoT (Internet of Things) の利用が

拡大している. これに伴い, IoT 機器を狙った攻撃やマルウェアによる被害が深刻化している. IoT 機器が攻撃された場合, IoT 機器が攻撃の踏み台にされる, あるいは IoT によって構成されるサービスや IoT 機器そのものが利用不可能になる危険性がある.

代表的な IoT マルウェアである Mirai [1] は, 脆弱なパスワードを使用する IoT 機器に侵入し, 大規模なボットネットを作成する. また, Mirai はソースコードが公開されていることから, 脆弱なパスワードを狙う Mirai に類似した

¹ 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology,
Okayama University
² NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories
³ 国立研究開発法人科学技術振興機構, さきがけ
JST, PRESTO

IoT マルウェアが数多く確認されている。しかし、現在は IoT マルウェアの攻撃手法の多様化が進み [2]、今後は PC と同様に機器が持つ脆弱性を利用するなど、攻撃の高度化が予想される。このため、IoT マルウェアへの対策として、IoT 機器で使用されるソフトウェアの更新機能や、セキュリティ機能を備える方法がある。

しかし、IoT 機器は汎用的な PC と比較し、ソフトウェアの自動更新機能が普及しておらず [3]、一度 IoT 機器の脆弱性が発見されるとネットワークに接続している同種の IoT 機器が一斉に影響を受ける。それだけでなく、一般的な PC と比較し、ユーザが日常的に IoT 機器を直接操作する機会が少ないことから、ソフトウェアの更新を能動的に実施したり、セキュリティ対策の意識を持ったユーザは少ない。さらに、IoT 機器の利用期間は、PC よりも長いことが想定されるため、ソフトウェアの更新などが適切に行われない場合でも、安全に利用できることが求められる。

我々は、IoT 機器のソフトウェアのセキュリティ上の課題を明らかにすることを目的としてファームウェア解析に基づいた予備調査を実施し、IoT 機器で利用されているメモリ破壊攻撃に対するセキュリティ機構、およびソフトウェアのバージョンを推定することでアップデート状況の実態を明らかにした [4]。予備調査の結果に基づき、多くの IoT 機器は PC と比較してセキュリティ機能が不十分な状態でリリースされており、また脆弱性が報告されている古いバージョンのソフトウェアを使用し続けている、と結論付けた。しかし、予備調査で得られた結論は以降に述べる観点から限定的である。具体的には、セキュリティ機能のうちメモリ破壊攻撃に対するものしか調査していないこと、ソフトウェアバージョンの推定方法が限定的なこと、調査対象が限定的かつ最新ではないこと、セキュリティ機能の低適用率や古いバージョンのソフトウェア利用の理由が不明、などが挙げられる。よって、IoT 機器のさらなる調査が必要であると考えた。

そこで、本研究では、IoT 機器のソフトウェア解析を大規模かつ多角的に実施し、さらに解析によって判明した IoT 機器のセキュリティ上の問題点の要因を明らかにするために、IoT 機器ベンダに対するインタビュー調査を実施することで、IoT 機器を取り巻くセキュア開発の実態調査を包括的に実施した。ソフトウェア解析の調査項目として、メモリ破壊攻撃に対するセキュリティ機構に加えて、カーネルへの攻撃に対するセキュリティ機能、および LSM ベースのセキュリティ機能を追加した。また、調査対象はファームウェアだけではなく、IoT 機器で使用されるソフトウェアの GPL (GNU General Public License) に従って公開されているソースコード (以下、GPL ソースコードと呼ぶ) も対象とした。我々は、2000 年から 2019 年に公開された国内外 13 ベンダの IoT 機器 1510 種に関する 5,712 ファームウェア、2,379 GPL ソースコードを収集して解析した。

本研究の主な貢献は以下の通りである。

- (1) IoT 機器のセキュリティ機構およびソフトウェアバージョンの体系的な調査方法を確立した。この調査方法を用いた本調査は、我々が知る限り初めての大規模かつ網羅的に実施されている調査である。
- (2) 実態調査によって、IoT 機器は既存のセキュリティ機能を十分に活用していないことを明らかにした。また、約 15 年に渡ってセキュリティ機能の適用率の変化を調査した結果、NX bit は増加傾向、PIE は減少傾向にあることを明らかにした。
- (3) 利用率の高い 32 ビットのプロセッサのアーキテクチャには、セキュリティ機能を適用しても、セキュリティ向上の効果が小さいものがあることを示した。さらに、アーキテクチャを 32 ビットから 64 ビットに変えることで、特にメモリ破壊脆弱性攻撃を緩和できる可能性を示した。
- (4) IoT 機器ベンダへのインタビューによって、ソフトウェアの脆弱性はバージョンアップではなくパッチで対処されることがあることを確認した。よって、ソフトウェア内のバージョン番号文字列に基づいて脆弱性の有無を調査する方法は誤検知を生じる可能性がある。
- (5) IoT 機器ベンダへのインタビューによって、IoT 機器開発のサプライチェーンにおける制約条件を明らかにした。チップベンダが提供する SDK に基づいて IoT 機器ベンダが開発する場合、IoT 機器ベンダがセキュリティ対策としてできることに検証コストや契約上の制約条件がある。

2. 脆弱性とセキュリティ機能

2.1 IoT 機器の脆弱性とマルウェアの脅威

IoT 機器を狙った代表的なマルウェアとして、Mirai [1] がある。Mirai は、推測が容易なパスワードを使用する IoT 機器を対象に、Telnet を用いて侵入し、IoT 機器をボット化する。ボット化した IoT 機器は、感染の拡大化のために、他の IoT 機器に同様の手法で侵入を試みる。攻撃者は、このようにして作成したボットネットを用いて、DDoS 攻撃を行う。Mirai により構築されたボットネットは、2016 年 8 月に初めて観測された。また、同年 9 月にソースコードが公開され、多くの Mirai 亜種が作成された。2019 年に観測された Mirai の亜種 ECHOBOT [5] は、任意のコマンド実行可能の脆弱性 (CVE-2017-17215) や遠隔コード実行可能の脆弱性 (CVE-2013-4863) などを利用し、IoT 機器に感染する。このように、機器の脆弱性を利用する IoT マルウェアが観測され始めている。

このため、IoT 機器を安全に利用するためには、ソフトウェアの脆弱性への対策を適切に実施することが重要である。以降では、調査対象とした攻撃を緩和できる機能 (以降、セキュリティ機能) とソフトウェアについて述べる。

2.2 セキュリティ機能

2.2.1 (機能1) メモリ破壊攻撃に対するセキュリティ機能

コンパイラにより ELF ファイルに適用できるセキュリティ機能について、ファームウェアに含まれる ELF ファイルへの適用率を調査した。ここで、適用率とは、ファイルシステム中で各セキュリティ機能が適用されている ELF ファイル数を、全 ELF ファイル数で割った値である。調査対象の4つの機能について以下に説明する。

RELRO (RELocation Read-Only) : 動的リンクの機能により、仮想アドレス空間内の主にライブラリ関数のアドレスを保持する領域である .got セクションを読み込み専用にするセキュリティ機能である [6]。RELRO を有効にすることで、ELF ファイルのロード時にアドレス解決を行い、.got セクションを読み込み専用にする。これにより、GOT 書き換え攻撃を防止する。

SSP (Stack Smashing Protection) : スタック領域内のローカル変数とリターンアドレスの間に canary と呼ばれる値を挿入し、関数の実行後に canary の書き換えの有無を確認することにより、Buffer Overflow 攻撃を検知する。

NX bit (No eXecute bit) : データを配置したメモリ領域に実行不可属性を付与することで、この領域のデータの実行を禁止する。Buffer Overflow 攻撃等によるコードインジェクション攻撃に対し、有効である。

PIE (Position Independent Executable) : ELF ファイルのアドレス参照を相対アドレスにすることにより、実行ファイルが配置される場所に関係なく実行できる技術である。PIE は ASLR (Address Space Layout Randomization) と併用することにより、実行ファイル自身のベースアドレスをランダム化できる [6] ため、コード再利用攻撃に対してより堅牢となる。

2.2.2 (機能2) カーネル攻撃に対するセキュリティ機能

カーネルがロードされるアドレスをランダム化化する KASLR [7] により、攻撃に利用するカーネル上のデータのアドレスの割り出しを困難にする。

2.2.3 (機能3) LSM ベースのセキュリティ機能

Linux のメインラインに含まれている4つの LSM ベースの強制アクセス制御のセキュリティ機能 (SELinux, AppArmor, Smack, TOMOYO Linux) を調査した。

2.2.4 IoT 機器の脆弱性

文献 [8] は、ファジングにより IoT 機器 17 台からメモリ破壊系の脆弱性 15 件を発見した。文献 [9] は、ファジング

により新たな脆弱性 2 件を発見し、これらはいずれもバッファオーバーフローの脆弱性であった。このように、IoT 機器にはメモリ破壊に代表されるプログラムの制御を奪取できる脆弱性が多く存在するため、これらの攻撃を緩和できる前述のセキュリティ機能の適用が有効である。

2.3 ソフトウェア

カーネル : カーネルの種類、およびバージョン名を取得した。IoT 機器で使用されるカーネルは Linux が主流であることが知られている [10], [11]。しかし、Linux には数多くの脆弱性が報告されており、今後は Linux の脆弱性を狙った IoT マルウェアも増加すると予想される。そこで、Linux のバージョン名を確認することにより、ファームウェアリリースとカーネルのバージョンの関係を明らかにし、どの程度新しいカーネルが利用されるのかを調査した。

アプリケーション : アプリケーションの種類、およびバージョン名を取得した。Busybox, Web サーバ, SSH サーバ, および OpenSSL を調査対象とした。IoT 機器では機器の設定を GUI を用いて行う場合などに Web サーバが用いられる。また、IoT 機器では、軽量かつ高速である lighttpd や mini_httpd の利用が多く、バージョンによっては脆弱性が報告されている。また、Busybox, SSH サーバ, および OpenSSL にも同様に脆弱性が存在する場合がある。

3. IoT 機器の調査内容と調査手法

3.1 調査内容と対象

2.2 節で述べたセキュリティ機能の適用率、および 2.3 節で述べたソフトウェアのバージョン情報を調査した。

調査対象であるファームウェア (FW), および GPL ソースコードから解析可能な項目を表 1 に示す。ファームウェアは、アンパックに成功した場合、(1) から (4) について解析できる。一方、GPL ソースコードは、公開されている場合、(2) と (5) の情報を取得し、解析できる。本調査では、GPL ソースコードを、ファームウェア解析で取得できなかった情報を補完するために利用した。

3.2 調査手法

3.2.1 調査手順

調査はプログラム収集を行う Program Collector, ファームウェア解析を行う Firmware Analyzer, および GPL ソースコード解析を行う Source code Analyzer を用いて実施し、セキュリティ機能の活用状況や利用されているソフトウェアのバージョン情報等を抽出する (図 1)。以降では、各調査コンポーネントの説明を行う。

3.2.2 プログラム収集

IoT 機器のファームウェアおよび GPL ソースコードは、Program Collector を用いてベンダサイトから収集する。
ファームウェア収集 : IoT 機器を販売するベンダは、自社

表 1 FW, および GPL ソースコードから解析可能な項目の比較

解析項目	FW	GPL
(1) RELRO, SSP, NX bit, PIE 適用の有無	✓	
(2) ソフトウェアのバージョン名の取得	✓	✓
(3) アーキテクチャ情報の取得	✓	
(4) バージョン間での比較	✓	
(5) カーネル設定ファイルの取得		✓

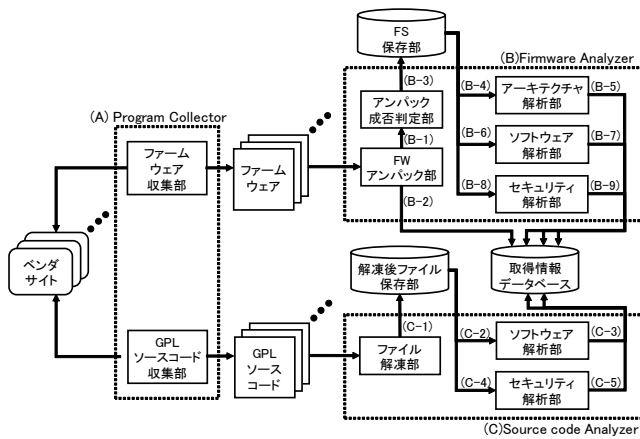


図 1 調査手順概要

のウェブサイトにてファームウェアを公開している。我々は、ベンダのウェブサイトからファームウェアのダウンロードリンクを抽出するスクレイピングツールを、収集対象であるベンダのサイト毎に作成した。本研究で対象にしたベンダは、世界的にシェアの高いベンダおよび国内ベンダの合計 13 社である。ファームウェアを収集するにあたって、既存研究で用いられる IoT 機器の定義（ネットワークに接続する機能を有している機器）に従い、我々はこの中でもコンシューマ向けの製品（例：ホームルータ）を IoT 機器とみなしてベンダサイトからファームウェアを収集した。なお、IoT 機器かどうかの判別が難しい機種も少数であるが存在する。ファームウェアは、一つの製品に対して異なるバージョンのファームウェアが存在するため、我々のツールでは公開されている全てのバージョンのファームウェアを収集した。

GPL ソースコード収集：ベンダのウェブサイトで公開されている場合が多いため、ファームウェア収集と同様の方法で取得した。一部のベンダは、入手にあたってサポートセンター窓口にお問い合わせの必要があったため、手動にて窓口のリクエストを送信して入手した。

3.2.3 ファームウェア解析

ファームウェアの解析を行う Firmware Analyzer は、4 つのモジュールで構成されており以下の手順で解析する。FW アンパック部でファームウェアをアンパックし、抽出されたデータをアンパック成否判定部に送信する (B-1) とともに、カーネル情報を取得情報データベースに送信する (B-2)。アンパック成否判定部でアンパックの成功が確認された場合、抽出されたファイルシステムを FS 保存部に保存する (B-3)。その後、アーキテクチャ解析部、ソフトウェア解析部、セキュリティ解析部に解析対象のデータを送信し (B-4, B-6, B-8)、各解析部で得られた情報を取得情報データベースに送信する (B-5, B-7, B-9)。以降では各モジュールの説明をする。

ファームウェア (FW) アンパック部：ファームウェアか

らファイルシステムの抽出、およびカーネル情報の取得を行う。この際、ファームウェアに含まれるファイルの抽出を行うツールである Binwalk [12] を利用する。また、Binwalk は、ファームウェアの解析時に、ファームウェアに含まれるファイルの形式やバージョン名などの情報を出力する。この出力結果から、カーネル情報を取得する。

アンパック成否判定部：ファームウェアのアンパックが適切に実施されたか否かを判定する。この際、ファイルシステムが抽出された場合をアンパック成功とみなす。

アーキテクチャ解析部：ファイルシステム中の ELF ファイルに対して file コマンドを実行し、利用されているアーキテクチャの種類とビット数 (32/64) を取得する。

ソフトウェア解析部：ファイルシステム中の ELF ファイルのバージョン情報を下記の 2 通りで取得する。

- 実際にアプリケーションを実行することによりバージョン名を取得する。この際、QEMU user mode emulation [13] を利用した。
- アプリケーションの実行ファイルに strings コマンドを使用することにより、バージョン名が文字列として出力される場合は、出力された情報を取得した。

これらの取得方法では、正規表現を用いてバージョン名と思われる情報を抽出する。このため、予備調査 [4] で作成した解析プログラムの正規表現を、精度向上とカバレッジ拡大のために改良と追加をして、解析可能な実行ファイルを増加させた。

セキュリティ解析部：RELRO, SSP, NX bit, PIE の適用の有無を確認する。この際、checksec [14] を利用した。

3.2.4 GPL ソースコード解析

GPL ソースコードの解析を行う Source code Analyzer は、3 つのモジュールで構成されており、以下の手順で解析する。ファイル解凍部で解凍された GPL ソースコードを解凍後ファイル保存部に送信する (C-1)。その後、ソフトウェア解析部、セキュリティ解析部に解析対象のデータを送信し (C-2, C-4)、各解析部で得られた情報を取得情報データベースに送信する (C-3, C-5)。以降では各モジュールの説明をする。

ファイル解凍部：収集した GPL ソースコードは圧縮されているため、事前に解凍した。

ソフトウェア解析部：GPL ソースコード中のソフトウェアのバージョン情報を以下の 2 通りで取得した。

- GPL ソースコードのファイル名に、ソフトウェア名に続いてバージョン名が記述されている場合に、そのバージョン名を取得する。
- 解凍後のディレクトリ内に、対象のソフトウェアのバージョン名が記載されている Makafile が含まれている場合に、記載されたバージョン名を取得する。

セキュリティ解析部：GPL ソースコード中に Linux のソースコードが格納されたディレクトリが存在する場合、同

表 2 収集したファームウェアと GPL ソースコードの内訳

ベンダ	収集した FW	(1) アンパック 成功 FW	(2)GPL ソースコード	(1) にも存在する (2)
A	1,314	846	99	136
B	1,791	394	29	6
C	553	264	14	1
D	223	176	2	0
E	247	141	9	0
F	222	128	2	0
G	134	97	106	54
H	113	57	2,067	1
I	239	44	0	0
J	374	40	25	0
K	71	28	0	0
L	163	3	5	0
M	268	1	21	0
Total	5,712	2,219	2,379	198

表 3 年ごとに集計したアンパック成功ファームウェアの内訳

年	全ての FW	各年のユニークな FW	年	全ての FW	各年のユニークな FW
2000	1	1	2011	35	28
2001	0	0	2012	46	39
2002	0	0	2013	68	51
2003	5	3	2014	122	75
2004	32	19	2015	247	136
2005	50	37	2016	340	178
2006	17	12	2017	375	220
2007	25	21	2018	387	214
2008	17	14	2019	264	146
2009	16	15	不明	109	
2010	63	37	Total	2,219	1,246

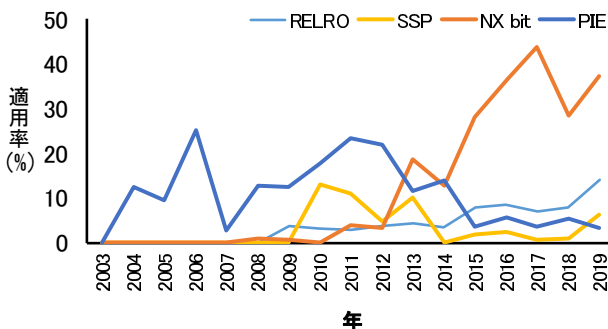


図 2 リリースされた年ごとの各セキュリティ機能の適用率

ディレクトリ内に Linux カーネルの設定ファイル（例：.config ファイル）が含まれている場合がある。 .config ファイルには、Linux カーネルのセキュリティ機能に関する設定が記載されているため、この情報をもとに、Linux カーネルのセキュリティ機能が利用可能か否かを調査した。

3.3 データセット

収集したファームウェアと、これらのうちアンパックに成功したもの、および収集した GPL ソースコードの総数を表 2 に示す。また、収集したファームウェアの年ごとの総数を表 3 に示す。表 2 の項目「(1) にも存在する (2)」

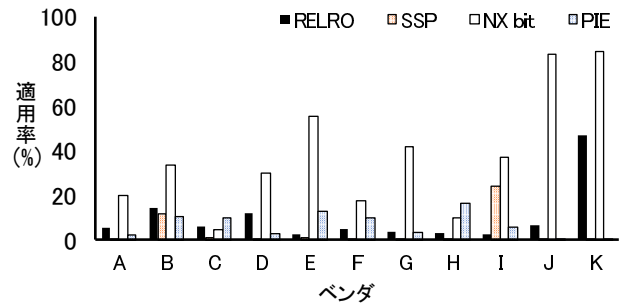


図 3 ベンダごとの各セキュリティ機能の適用率

は、アンパックに成功したファームウェアのうち、対応する GPL ソースコードも存在した件数である。

以降では、ファームウェア対象の調査では「(1) アンパック成功 FW」を、GPL ソースコード対象の調査では「(1) にも存在する (2)」を調査対象とする。ファームウェアのアップデート回数が多い IoT 機器は、調査結果に大きく影響するため、4 章の図 2 と図 3 の調査結果では、同じ年に複数回アップデートした製品に関しては、その年の最新のファームウェアのみを評価対象とした。この評価対象の数は、表 3 の項目「各年のユニークな FW」に該当する。

4. 調査結果

4.1 セキュリティ機能の適用率

RELRO, SSP, NX bit, および PIE の年ごと、およびベンダごとの適用率を図 2, 図 3 に示す。図 2 では、ファームウェアから ELF ファイルが抽出できた 2003 年以降の結果を載せている。また、図 3 では、調査対象のファームウェアが 5 件以上のベンダを載せている。図 2 より、以下のことが分かる。PIE は、2011 年以降、適用率が減少に転じている。一方、NX bit は、他のセキュリティ機能と比較し、2011 年以降大幅な増加傾向にある。また、図 3 より、ベンダによって適用率には差があることが分かる。

KASLR, および LSM ベースのセキュリティ機能の利用有無を、アンパックに成功したファームウェアに対応する GPL ソースコード 198 件を対象に調査した。カーネルの設定ファイルを 48 件確認できたものの、これらのセキュリティ機能を有効にしているものは確認できなかった。

4.2 ソフトウェアのバージョン

推定したバージョンに脆弱性が報告されていた件数を表 4 に示す。また、ファームウェアアップデートによりソフトウェアのバージョンが変化した製品の総数を表 5 に示す。調査では、最も古いバージョンと最も新しいバージョンのファームウェア同士を比較し、バージョン変化を確認した。表 4, では、2019 年リリースの 264 件のファームウェアを対象とし、ソフトウェアのバージョン名を推定できなかった場合は調査の対象外としている。また、表 5 で

表 4 2019 年リリースのファームウェアで推定したバージョンに脆弱性が報告されていたソフトウェアの件数

脆弱性	カーネル	Busy box	Web サーバ	SSH サーバ	Open SSL
有	229	234	25	157	160
無	0	0	2	10	11
Total	229	234	27	167	149

表 5 ファームウェアアップデートによりソフトウェアのバージョンが変化した製品

変化	カーネル	Busy box	Web サーバ	SSH サーバ	Open SSL
有	39	42	16	16	15
無	330	306	49	85	90
Total	369	348	65	101	105

表 6 2019 年リリースのファームウェアの Linux カーネルのバージョン

バージョン	総数	バージョン	総数	バージョン	総数
2.6.16	1	3.0.36	6	3.10.4	2
2.6.18	3	3.3.8	29	3.10.9	3
2.6.19	1	3.4.10	2	3.14.4	4
2.6.21	28	3.4.11	2	3.14.7	9
2.6.22	8	3.6.5	1	3.18.2	7
2.6.31	27	3.10.1	25	4.1.27	2
2.6.32	1	3.10.2	1	不明	35
2.6.36	67				

は、アップデートが一回以上行われた製品は 418 件存在した。このうち、比較するソフトウェアのバージョンが一方でも推定できなかった場合は調査の対象外としている。

表 4, 表 5 より、多くの IoT 機器は脆弱性が確認されているソフトウェアを利用しているものの、ソフトウェアのアップデートは実施されていないことが多いことが分かる。

また、2019 年リリースのファームウェア (264 件) で利用されるカーネルのバージョンを表 6 に示す。表 6 では、バージョン情報が推定できない場合は不明とした。表 6 より、古いもので Linux 2.6.16 (2006 年 3 月リリース)、最も新しいもので Linux 4.1.27 (2016 年 6 月リリース) と、多くのファームウェアは数年前から十数年前にリリースされた Linux カーネルを利用していることが分かる。

4.3 アーキテクチャの種類

アーキテクチャごとの利用率を図 4 に示す。図 4 では、ファームウェアから調査対象の ELF ファイルが抽出できた 2003 年以降の結果を載せている。図 4 より、多くの IoT 機器は 32 ビットの MIPS アーキテクチャを利用しており、これに続いて 32 ビットの ARM アーキテクチャの利用率が高いことが分かった。また、64 ビットのアーキテクチャとして MIPS、および x86-64 アーキテクチャ確認された。しかし、図 4 より、これらが利用された例は 2003 年から 2019 年までほとんど存在せず、現在においても 32 ビットのアーキテクチャが主流であることが分かった。

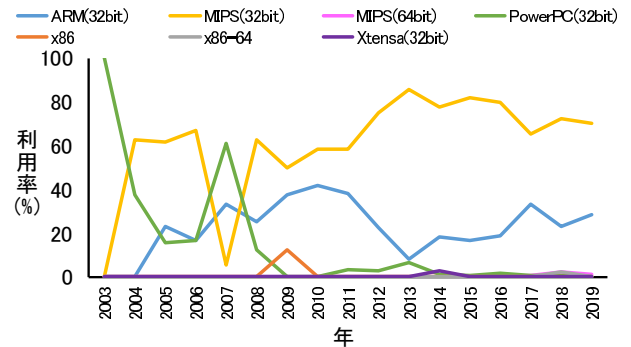


図 4 アーキテクチャの年ごとの利用率

5. ベンダへのインタビュー調査

実態調査では、セキュリティ機能が十分に活用されていない状況、および推測したソフトウェアのバージョン情報が必ずしも最新ではないことを明らかにした。これらの状況が発生している理由を明らかにするため、IoT 機器を販売するベンダ複数社に対してインタビュー調査を実施した。

5.1 インタビュー調査方法

インタビュー調査は、コロナ禍の状況を鑑みて、メールを利用したオンラインインタビューの形式で実施した。インタビュー調査は、実態調査の対象になっている 13 社の中から 4 社 (以降では、V1, V2, V3, V4 と表記する) に対して実施した。我々は、セキュリティ機能に関する質問、およびソフトウェアのバージョンに関する質問を行なった。我々は、各社に対して任意でのインタビューへの回答を依頼し、また回答内容については社名を匿名化した上で論文に掲載することの承認を得た。

5.2 インタビュー調査結果

5.2.1 セキュリティ機能

セキュリティ機能については、“同じベンダであっても、製品毎にセキュリティ機能の適用率に差がある場合についての理由”について質問した。

V1 および V2 は、セキュリティ機能を意識的に適用しているわけではない、と回答した。また、V3 および V4 からは、チップベンダから提供されるソフトウェア開発キット (SDK) が影響していることが指摘された。この場合の SDK とは、リファレンスとなるファームウェアであり、これをベースに各製品のファームウェアを開発している。V3 は、ビルド条件を変更した場合の影響範囲が不明であることから変更が安易にできないことを指摘した。V4 は、セキュリティ機能の適用に関する検証コストに加えて、チップベンダによる SDK の動作保証やサポートの可否にも注意する必要があることを指摘した。

これらの回答から、セキュリティ機能の適用率の違いについては、ベンダ各社の開発方針というよりは、チップベ

ンダが提供する SDK の特性に影響されている場合が多いことが判明した。

5.2.2 ソフトウェアのバージョン

ソフトウェアのバージョンについては、“ソフトウェアに脆弱性が発見された場合の対応”，および“ソフトウェアのバージョンアップ・最新化する基準”について質問した。

全てのベンダから、パッチの適用に関する言及があり、特に V2, V3, V4 からは、脆弱性の対処ではバージョンアップを行わずパッチを適用する場合がある、と言及された。V2 から、モジュールそのものは基本的なバージョンとしては正しい、という回答があったものの、全てのベンダより、バージョンアップせずにパッチが適用された場合はバージョン番号から脆弱性を識別する方法は正確ではない、との指摘があった。

ファームウェアアップデート時のソフトウェアの最新化については、V1, V2, V3 からは必ずしも実施しないとの回答を得た。V2 は、脆弱性が確認された場合のみバージョンアップを実施すると回答し、V3 は、最新化にすることによる製品に対する影響度や開発リソースの状況によって最新化するかを検討すると回答した。

6. 考察

6.1 PIE の適用率が減少傾向にある理由

PIE を適用している場合は未適用に比べ、アーキテクチャによっては処理時間が約 5～10% 増加することが知られている [15]。このため、このオーバーヘッドを回避していることが考えられる。また、32 ビットのアドレス空間では、ASLR のエントロピーがブルートフォース攻撃に十分耐えられるほど大きくないことも理由として考えられる [16]。図 4 が示す通り、IoT 機器は 32 ビットアーキテクチャを用いる場合が多く、PIE による利点が得られにくいため、利用率が減少している可能性がある。

また、文献 [16] では、ブルートフォース攻撃に耐えるエントロピーを持つ ASLR の設計には、64 ビットアーキテクチャを利用すべきと主張している。このため、セキュアな開発を実施する場合、コストを考慮する必要があるものの、64 ビットアーキテクチャは十分に選択肢になり得る。

6.2 カーネルバージョンやプロセッサの制約

Linux カーネルのバージョンや、プロセッサのアーキテクチャによる制約条件により、そもそも KASLR や LSM ベースのセキュリティ機能を利用できない場合がある。また、SELinux は組み込み機器のファイルシステムで多く利用される SquashFS では利用できない。このことから、セキュリティ機能が未使用である場合が多いと推察できる。

6.3 脆弱性調査のためのバージョン特定手法の課題

ソフトウェアに含まれるバージョン番号文字列に基づい

表 7 同一のバージョン間でハッシュ値が変化したソフトウェア

変化	Busybox	Web サーバ	SSH サーバ	OpenSSL
有	242	33	36	34
無	64	12	48	54
Total	306	45	84	88

て脆弱性の有無を調査する方法には、問題点があることをベンダへのインタビュー調査で明らかにした。表 4, 表 5 より、脆弱性が確認されているものの、ソフトウェアのバージョンアップを行わない IoT 機器が多いことがわかる。この理由として、ベンダの回答の通り、ソフトウェアのバージョンアップではなく、パッチを適用することにより脆弱性に対処していることがある。また、バージョン名の後ろに「devel-XXX」など、通常のバージョン名ではないバージョン名をもつ lighttpd が存在した。このことから、パッチを適用していることが推察される。また、ベンダからの回答として、パッチを適用した場合はバージョン番号に変化がない場合があると指摘があった。

実態を調べるため、同じ製品のうち、最も古いバージョンと最も新しいバージョンのファームウェアの両方が同じバージョンの同じソフトウェアを利用している場合、両者 2 つのソフトウェアのハッシュ値を比較した。調査対象の 4 つのアプリケーションで上記の比較を行った結果を表 7 に示す。表 7 から、同じバージョンの同じソフトウェアで、ハッシュ値が異なる場合が多く存在することが分かる。この原因として、パッチの適用により、ハッシュ値が異なる場合が存在すると推察できる。これに加えて、ビルドされた日付やファームウェアのバージョンが ELF ファイルに埋め込まれており、ハッシュ値が異なる事例も確認した。

以上のことから、脆弱性の有無を調査する際はバージョン番号やハッシュ値による単純な比較ではなく、パッチ適用の有無を含めたより詳細なプログラム解析が求められる。

6.4 IoT 機器開発におけるサプライチェーンの課題

ベンダへのインタビュー調査によって、IoT 機器をセキュアに開発する際に、そのベースとなる SDK を提供するチップベンダとのサプライチェーン上の制約事項があることを明らかにした。Thompson と Zatko は、IoT 機器のセキュリティ機能の低い適用率に対して、“セキュリティ機能を適用することはなんら難しいことはないので、IoT 機器で利用するソフトウェアのビルド時に有効にすべき”と提言している [17]。しかしながら、上記のサプライチェーンを鑑みると IoT 機器ベンダの努力だけでは解決できない問題であることがわかる。よって、IoT 機器ベンダだけではなく、チップベンダに対してもセキュリティを考慮した開発を進めるための技術を検討する必要がある。

7. 研究倫理

データセットとして利用したファームウェア、および

GPL ソースコードは各ベンダのサイトで公開されており、誰でも入手可能なものである。これらを収集するにあたり、サイトへの負荷を低減するために、アクセスは一般のエンドユーザが行う自然な動作に従うように実施した。具体的には、各ベンダのサイトにアクセスし、対象とする IoT 機器の製品ページからダウンロードリンクを探して収集した。その際に、アクセスの間隔を十分に空けることで、サイトへの負荷が集中することを避けるよう努力した。

8. 関連研究

文献 [17] は 2018 年に人気であった 28 台のホームルータを対象に、ASLR, NX bit, RELRO, および SSP の適用率を調査している。この調査により、PC と比較し、ホームルータは ELF ファイルに十分なセキュリティ機能を適用していないことを示している。また、全体的に ARM を使用しているホームルータの方が、セキュリティ機能の適用率が高かったと述べられていた。我々の研究は、ホームルータに限らず幅広い IoT 機器を対象として大規模にファームウェア、GPL ソースコード、およびベンダへの調査により解析しているため、より IoT 機器の全体の傾向を時系列変化を含めて示す結果が得られていると推察できる。

文献 [6] は 32/64bit Linux ディストリビューションを対象に RELRO, SSP, PIE, および Automatic Fortification の適用率を調査し、ディストリビューション毎の普及度の違いや一部の対策技術しか機能していない事例を明らかにした。本研究は、IoT 機器を多角的に大規模調査し、解析したことに新規性があり、IoT 機器における Linux の利用実態やセキュリティ機能の適用率と要因を明らかにした。

文献 [11] は Linux を利用する組み込み機器のファームウェアのエミュレーションと動的解析を実行し、脆弱性を調査する FIRMADYNE を提案した。これに対し、我々の研究は、ファームウェアを静的、および動的に解析し、セキュリティ機能が適用されていない要因を調査した。

9. おわりに

IoT 機器のソフトウェアのセキュリティ上の課題を明らかにすることを目的に、ファームウェアと GPL ソースコードを解析し、利用されるセキュリティ機構やソフトウェアのバージョンを調査した。また、これらの調査結果をもとにベンダへインタビューを行った。

結果として、IoT 機器はセキュリティ対策が十分に行われておらず、また脆弱性が存在するバージョンのソフトウェアを利用し続けていることが分かった。この理由をベンダに質問したところ、ベンダ各社の開発方針ではなく、チップベンダが提供する SDK の特性に影響されている場合が多いなど、サプライチェーン上の制約条件があることを明らかにした。また、脆弱性に対してはパッチを適用することにより対処されることがあることを確認した。

今後の課題として、脆弱性の有無を判別するためにプログラムのパッチを解析する技術の実現がある。

謝辞 本研究の一部は、JST, さきがけ, JPMJPR1938, および JSPS 科研費 JP19H04111 の助成を受けたものです。

参考文献

- [1] Antonakakis, M. et al.: Understanding the mirai bot-net: *Proc. 26th USENIX Conference on Security Symposium*, pp.1093–1110 (2017).
- [2] Vignau, B., Khoury, R., Hallé, S.: 10 Years of IoT Malware: A Feature-Based Taxonomy, *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp.458–465 (2019).
- [3] 桑名栄二: IoT 機器の脆弱性と対策, *ビジネスコミュニケーション*, Vol.56, No.56, pp.12–13, 2019.
- [4] 白石周基, 福本淳文, 塩治榮太朗, 秋山満昭, 山内利宏: ファームウェアに着目した IoT 機器のセキュリティ機能の調査, 第 50 回情報通信システムセキュリティ研究会 (ICSS), 電子情報通信学会技術研究報告, vol.119, no.437, pp.37–42 (2020).
- [5] トレンドマイクロセキュリティブログ: 複数の脆弱性を利用してさまざまなルータを狙う「Mirai」の新しい亜種を確認, 入手先 (<https://blog.trendmicro.co.jp/archives/20908>) (参照 2020-07-28).
- [6] 齋藤孝道ほか: Linux ディストリビューション間での ELF バイナリにおけるセキュリティ対策機構の適用状況の比較, *コンピュータセキュリティシンポジウム 2018 論文集*, vol.2018, no.2, pp.1079–1086 (2018).
- [7] Hund, R. et al.: Practical Timing Side Channel Attacks against Kernel Space ASLR, *2013 IEEE Symposium on Security and Privacy*, (2013).
- [8] Chen, J. et al.: IoTfuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing, *Proc. ISOC NDSS*, (2018).
- [9] Zheng, Y. et al.: FIRM-AFL: High-Throughput Grey-box Fuzzing of IoT Firmware via Augmented Process Emulation, *Proc. 28th USENIX Conference on Security Symposium*, pp.1099–1114 (2019).
- [10] Costin, A. et al.: A Large-Scale Analysis of the Security of Embedded Firmwares, *Proc. 23rd USENIX Security Symposium*, pp.95–110 (2014).
- [11] Chen, D. D., Egele, M., Woo, D., Brumley, D.: Towards automated dynamic analysis for Linux-based embedded firmware, *Proc. ISOC NDSS*, (2016).
- [12] ReFirmLabs: binwalk, available from (<https://github.com/ReFirmLabs/binwalk>) (accessed 2020-07-31).
- [13] QEMU: QEMU version 4.1.0 User Documentation, available from (<https://qemu.weilnetz.de/doc/qemu-doc.html>) (accessed 2020-07-31).
- [14] slimm609: checksec.sh, available from (<https://github.com/slimm609/checksec.sh>) (accessed 2019-12-20).
- [15] ubuntu wiki: Security/Features, available from (<https://wiki.ubuntu.com/Security/Features#pie>) (accessed 2020-8-1).
- [16] Shacham, H. et al.: On the Effectiveness of Address-Space Randomization, *Proc. 11th ACM conference on Computer and communications security (CCS' 04)*, pp.298–307 (2004).
- [17] Parker, T., Sarah, Z.: Build Safety of Software in 28 Popular Home Routers, *Cyber-ITL*, (2018).