DPIを用いたIoT機器向け不正侵入検知システムの検討

油田 健太郎[1]　山場 久昭[1]　朴 美娘[2]　岡崎 直宣[1]

概要：従来のコンピュータやスマートフォンだけでなく, 家電や自動車, 医療関連機器やスマートホームなど多様で幅広い機器・設備がインターネットに接続されている．一方で，これらのシステムを悪用する脅威も増加している．本論文では，ファイアウォールによるパケットフィルタリングを用いてそのようなIoT 機器に対する侵入検知・対策の実用性を確認する．その際パケットのペイロード部まで検査する Deep Packet Inspection を用いることで，不正な接続だけでなくプロトコルを偽装した通信の検知を試みる．その結果，従来のパケットフィルタリングより高度な振る舞いが可能であること，パケットが偽装されていてもパケットフィルタリングに有用であることやそれを用いた侵入検知システムの実装について示す．

An Examination of Intrusion Detection System for IoT Devices Using Deep Packet Inspection

KENTARO ABURADA[1]　HISAAKI YAMABA[1]　MIRANG PARK[2]　NAONOBU OKAZAKI[1]

1. はじめに

近年，情報通信技術の発達によりパソコンやスマートフォンなどに限らず家電，自動車，医療機器，産業用ロボットなど多くの機器がインターネットに接続されるようになった．こうしたモノのインターネット (Internet of Things，IoT) の普及などにより，2015 年時点で約 50 億台の IoT 機器がインターネットに接続されており，その台数は年々増え続けている [1]．幅広い機器が相互に接続されることによってこれまでになかった画期的な技術やサービスが日々登場してきている一方で，これらの機器を狙った侵入・攻撃も確認されている．警察庁によると，宛先ポート 23/TCP(Telnet) におけるアクセスについて解析した結果，組込み機器のみを狙った不正な通信が存在することが確認された [2]．また，Telnet 接続以外にも海外製の特定のルータの脆弱性を突く接続試行も増加している．2016 年 9 月には，Mirai と呼ばれるマルウェアに感染したルータや防犯カメラなど数十万台の IoT 機器によって史上最大規模の分散型サービス妨害攻撃が発生した [3]．これらの事例では機器が単にパケットを送信するボットとして利用されたが，もし医療関連機器や産業用ロボットに侵入された場合にはシステムがダウンする恐れや現金搾取の引き換えとして悪用される可能性がある．また近年は自動車製造工場のような産業施設だけでなく，路面電車システム，発電所や送電網といったインフラなど，標的となる設備や施設が多様化してきている [4]．パソコンやスマートフォンだけでなくインターネットに接続可能なあらゆる機器に対するセキュリティ対策が必要となってきている．しかしながら，組込み機器のように限られたリソース環境下で十分なセキュリティ対策を施すのは困難である．また既に数多く出回っている機器すべてに安全な状態を提供することもコスト・技術面から容易ではない．そこで本論文では Deep Packet Inspection というパケット検査技術の動作確認を行った後，DPI を用いた IoT 機器向けの侵入検知システムの実装を試みる．

2. マルウェア

IoT 分野において Mirai に代表される「マルウェア」とは，コンピュータシステムに対して不正な動作を行うソフトウェアの総称である．マルウェアの中には，単独で動作・感染するコンピュータウイルスの他にも，不正行為に利用されるツール型やクッキーを取得するためのデータ型など様々な種類がある．近年は特にマルウェア作成ツールが闇市場で取引されるようになり，個体数や亜種が増加しシグネチャを用いたパターンマッチングによる対策が困難となってきている．また，マルウェア本体も巧妙化し解析環境やアンチウイルスソフトウェアによる調査を回避するものも確認されている．利用者の多いパソコンやスマートフォンを狙ったマルウェアにはソフトウェアの脆弱性を突いて侵入するケースが多い．発見されているもののメーカやベンダから対策パッチがリリースされていないゼロデイ脆弱性を悪用する手口に対しては決め手となる入口対策手法がないため，感染後の振る舞いからの検知やデータ流出を防ぐ防御手法が多く取り入れられている．このような背

[1] 宮崎大学
[2] 神奈川工科大学

Copyright (c) 2017 by the Information Processing Society of Japan

景から，L7 ファイアウォールなどの次世代ファイアウォールが注目されている．

3. ファイアウォール

クライアントサイドのプログラムでは利用する OS や組込みシステムの違う多様な機器に対して対策が困難である．またリソースが十分に確保できない場合も多く考えられる．そこで，より効率的に運用できるようにネットワーク境界ごとでのファイアウォールによるアクセス管理を試みる．ファイアウォールには大きく分けてアプリケーションゲートウェイ型，パケットフィルタリング型の 2 つがある．

3.1 アプリケーションゲートウェイ型

プロキシサーバとして動作する方式である．特定のプロトコルに特化し，そのプロトコルを解釈することで詳細なアクセス制御を可能としている．またファイアウォール内部の機器が直接外部と接続しないため安全性は高いが，プロトコルごとにそれを解釈できる個別のゲートウェイプログラムを用意する必要がある（図 **1**）．

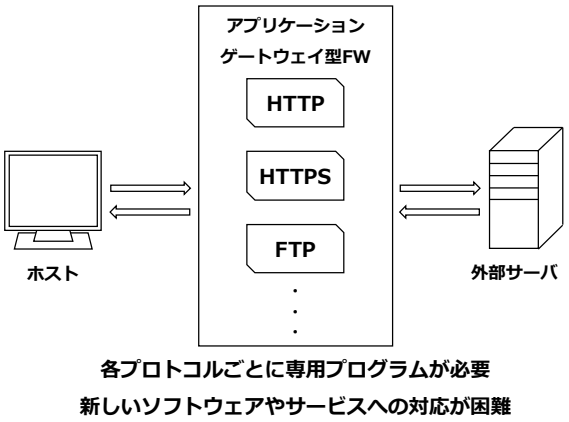

図 **1** アプリケーションゲートウェイ型ファイアウォールの概要

3.2 パケットフィルタリング型

3.2.1 従来のパケットフィルタリング

パケットフィルタリングは受信したパケットの IP アドレスやポート番号を検査してパケットを破棄するかどうかを決定する手法である．高速な処理が可能だがアドレス・ポート情報を偽装することで通過されてしまうため攻撃耐性が低い．また，プライベートアドレス環境下から外部へ DNS や FTP などのリクエストを送信した場合，送信元ポート番号でフィルタリングすると同一ポートから送信された悪意ある通信も通してしまう．これを宛先ポート番号でフィルタリングすると 1024 以上のすべてのポートを利用する通信を通してしまうことになり，静的なフィルタリングでは限界がある．

3.2.2 ステートフルパケットインスペクション

この欠点を克服するためにステートフルパケットインスペクション（Stateful Packet Inspection，以下 SPI）が考案された（図 **2**）．実際に通過したパケットから戻りパケットの通過条件を逐次作成するダイナミックパケットフィルタリングの 1 種で，LAN 側から送信したデータを保存しておくことで，受信したパケットのうちフラグやシーケンス番号において整合性の取れているパケットのみを通過させられるように動的にルールを構成する．

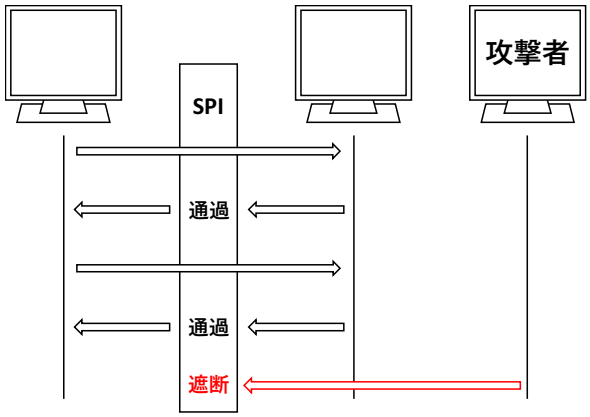

図 **2** SPI の概要

3.2.3 ディープパケットインスペクション

ヘッダ部分の検査だけではデータ部に混入しているウイルスを検出することができない．今回はパケットのヘッダ部だけでなくデータ部まで検査範囲を拡大したディープパケットインスペクション（Deep Packet Inspection，以下 DPI）に着目した（図 **3**）．実際にデータの含まれているペイロード部まで検査するためウイルスや不正コードを検出することができる．従来の方式では検出できなかった，整合性が取れていても不正なデータが混入されているパケットの検出・破棄が可能であり，昨今各社のファイアウォール機器に広く取り入れられている方式である．7 層（Layer 7）の情報を取り扱うため L7 Firewall とも呼ばれる．

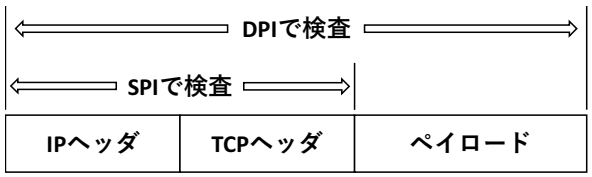

図 **3** DPI の検査範囲

3.3 侵入検知・防御システム

通常のファイアウォールとは異なるが，同じくネットワークやサーバを保護するアクセス制御方式に侵入検知システム（Intrusion Detection System，以下 IDS），侵入防御システム（Intrusion Prevention System，以下 IPS）が

Copyright (c) 2017 by the Information Processing Society of Japan

ある．IDS は行われている通信のパケットのコピーのうち，ネットワーク層からアプリケーション層に対する侵入，攻撃を検知するシステムである．IPS は通信路上に設置することで，IDS では検知にとどまっていた攻撃の防御が可能である．これらは**図 4** の通りネットワーク層，トランスポート層の制御を行っていたファイアウォールに対し守備範囲が広いため，ポートスキャンなどのネットワーク層を用いた攻撃だけでなく，3way ハンドシェイクのコネクション要求を悪用する SYN Flood 攻撃からの防御にも有効である．SYN Flood 攻撃などの DoS (Denial of Service) 攻撃やワームを含むパケット持つ特徴パターン（シグネチャ）がないかをパターンマッチングを行って確認するシグネチャ型，正常なシステム利用では考えられないプロトコルやトラフィックを検出するアノマリ型があり，ファイアウォールより高度なアクセス制御が可能となっている．

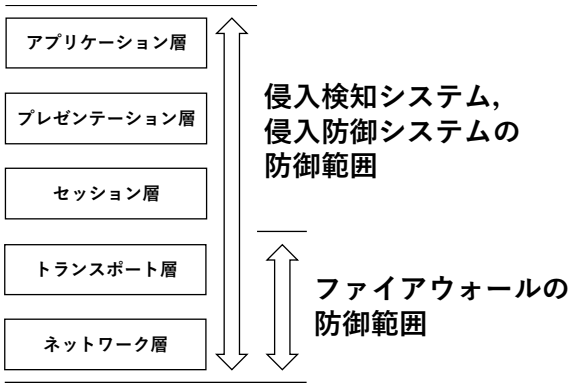

図 4 FW・IDS・IPS の防御範囲

4. 対策と関連研究

Telnet は本来，ネットワークに接続された機器を遠隔操作するためのプロトコルである．TCP 最初期から標準化されており広く普及しているため多くの OS で利用可能であるが，暗号化されずにデータ伝送が行われることから公開鍵暗号方式を用いた SSH が代替として利用されている．しかし Telnet ポートが開放された状態で数多くのシステムや機器が市場に出回ってしまった影響で，パソコンと違いセキュリティ対策の十分でない．そのため機器へ Telnet を用いてリモート接続で侵入し乗っ取りを行う攻撃手法が広まっていった．前述した Mirai は機器に初期設定として利用されることの多いユーザ名及びパスワードの組み合わせをリモート接続で試行し侵入および感染を試みるマルウェアであった．IoT やインターネット接続機器の急増に伴い”root”や”password”など汎用的なパスワードが初期設定されたまま利用されるケースも少なくない．この問題に関しては各ベンダやセキュリティ企業が注意喚起を促している [5]．リモートログインを行う telnet を狙った攻撃だけでなく，既知の脆弱性を持つ特定機器を狙った攻撃も IoT 機器の脅威となる [6]．[7] では制御システムを狙った近年の脅威とその対策の考察について述べられている．産業機器で用いられる機器同士のデータ交換のための標準規格である Object Linking and Embedding for Process Control (OPC) やシステムの属するネットワーク全体の保護など制御システムに絞った脅威の特性がまとめられている．また仮想化，ネットワーク系対策の 2 つの観点から制御システムのセキュリティについて考察されている．[8] では，組込みシステム向け仮想化環境である SafeG を用いて脅威検知のための機能提供を実現している．仮想化環境を用いて，汎用 OS とは独立している監視機能を持たせた OS をリアルタイムで動作させ，汎用 OS の実行シーケンス，割込みの情報をリアルタイムに取得している．この概要を表したものが**図 5** である．これらの情報はネットワークからの攻撃やウイルスに感染した際の振る舞いの変化を検出するために有用である．汎用 OS とリアルタイム OS を組み合わせて利用することでシステム全体の信頼性とリアルタイム性を維持している．

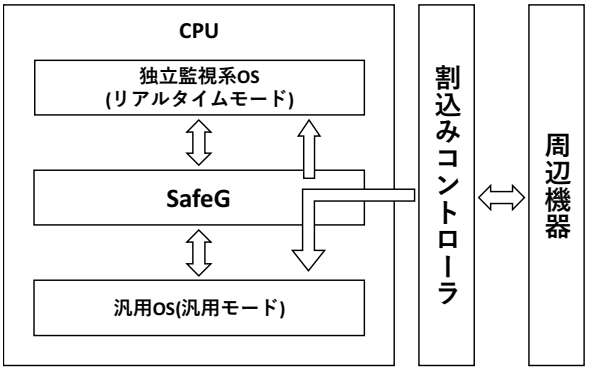

図 5 組込みシステム向け仮想化環境を用いた汎用 OS の振る舞いの監視

4.1 暗号化通信の検知

DPI に関する研究には以下に示すものがある．[9] では，ファイル共有ソフトである Winny の暗号化を強化した Winnyp の通信の検知を試みている．Winnyp は，従来の RC4 暗号のみを用いてデータの暗号を行っていた Winny に加えて，新たに DES や MD5 との排他的論理和などを用いたりダミーデータを付与してペイロード長を不定にしたりすることによって匿名性を高めている．このため，まずハッシュ変換処理や暗号処理を行い鍵を作成して Winnyp クライアントと同一の方法で暗号データを復号する．その後復号したデータを識別し Winnyp による通信であると判断された場合には異常を通知するシステムである．この概要を表したものが**図 6** である．Winny 及び Winnyp では設定によって通信に任意のポートを利用することができるため，このように復号化したデータ部を用いて Winnyp の通信であるかを推定し制御している．またこの研究では，

Copyright (c) 2017 by the Information Processing Society of Japan

GPGPU を用いることによって複数の暗号の復号処理を並列化することで，ギガビットクラスのネットワークにも対応できる高速処理を実現している．

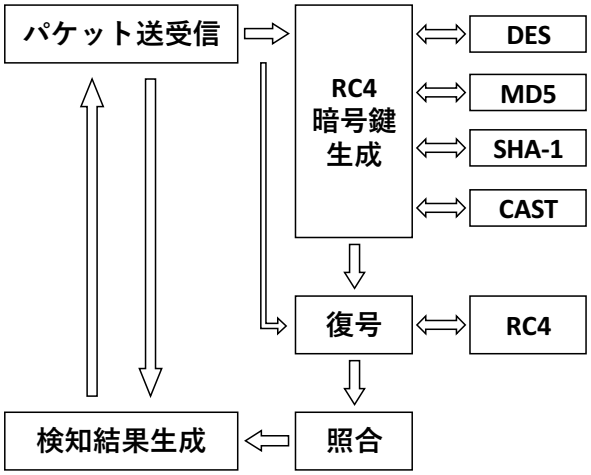

図 6　Winnyp 通信検知機能の概要

4.2　プロトコル偽装マルウェアの作成

[10] ではプロトコルを偽装した通信を行うマルウェアの潜在的な危険性を確認するために，そのような通信を行うマルウェアを試作している．入力として平文，フォーマット，そして鍵を与えることで出力としてフォーマットに沿った暗号文を得ることができるフォーマット変形符号を利用して，任意のプロトコルに擬態した暗号文を生成する．総合的な擬態通信を行うフレームワークである Marionette を用いて暗号文を送信する．この概要を表したものが**図 7** である．図の Command&Control サーバはマルウェアに感染したコンピュータ群への指令送信や制御を担う司令塔である．これを用いて HTTP の GET メソッドを HTTPS 通信に擬態する実験を行い，各層における検知の有効性を確認している．

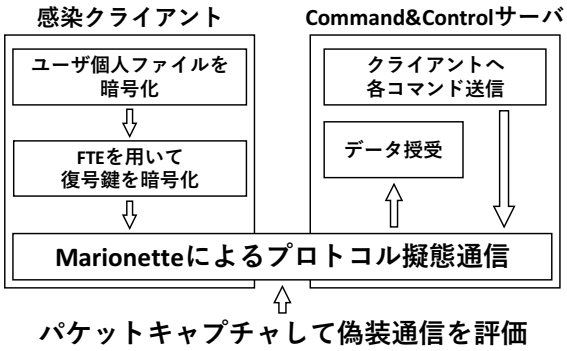

図 7　プロトコル擬態通信マルウェアのプロトタイプ概要

5.　DPI 機能の実装

通信プロトコルを偽装したパケットが検出できる点，IoT 機器の利用プロトコルの多様性から汎用性・拡張性に優れたシステムが好ましい点などから，DPI を用いた IoT 機器に特化したセキュリティシステムについて検討する．まず DPI 機能を実装するにあたり現在利用可能なエンジン，ライブラリについて調査した．

5.1　Hi-Performance Protocol Identification Engine(HiPPIE)

HiPPIE は Linux のカーネルモジュールとして提供するためオープンソースで開発が進められているプロジェクトである．DPI に加えパケットフィルタリング機能も同様に備えている．

5.2　Linux layer 7 packet classifier

Linux layer 7 packet classifier は Netfilter フレームワーク用の DPI パッケージである．Netfilter は Linux カーネルに標準実装されているため親和性が高い．iptables から設定・制御することが可能である．

5.3　nDPI

nDPI はネットワーク管理システムなどを研究・販売している ntop 社が整備しているオープンソースの DPI 用ライブラリである．ホストアドレスや使用ポートを定義することで新しいプロトコルを追加することができるため，利用する機器や環境に合わせてプロトコルを追加することができる．今回は新規のソフトウェアに組み込んで実装するためこのライブラリを用いた．

6.　動作確認

本項以降の実験に使用した環境を**表 1** に示す．本研究では VirtualBox を用いて仮想マシンにインストールした CentOS 上に環境を構築した．また実行環境のネットワークはプロキシサーバを介して外部と通信を行う．

表 1　実験に用いた環境

	ホストマシン	仮想マシン
OS	Windows 10 Enterprise	CentOS 7
IP アドレス	172.22.2.138	192.168.56.101

まずホストマシン側で行った通信を Wireshark でパケットキャプチャし，得られた結果を仮想マシンで処理する．DPI には nDPI ライブラリに同梱されている ndpiReader を利用する．ndpiReader は入力方法や追加のプロトコル定義ファイルを指定してパケットを判別するプログラムである．入力は NIC とキャプチャ済みのパケットを記録し

Copyright (c) 2017 by the Information Processing Society of Japan

た pcap ファイルから選ぶことができる．まずコンパイルしたプログラムのあるディレクトリへ移動し図 8 に示すコマンドを実行することで capture.pcap に保存されたパケットに対してスキャンを行う．また実行結果を pcap.txt というファイルに保存する．

```
sudo ./ndpiReader -i capture.pcap -v 2 -w pcap.txt
```

図 8　キャプチャしたパケットを保存したファイルからパケットを検査し，結果をテキストファイルへ保存するコマンド

nDPI の公式ホームページ[*1]にて対応プロトコルを確認して，リストの中からアクセスするサイト・開始するサービスを選択する．今回は例として，Web ブラウザで Amazon，Youtube にアクセスする．アクセスを終えたらキャプチャを停止し，パケットを保存する．このとき，後で処理しなければならないパケット数が多いのであらかじめ TCP のパケットのみをフィルタで抽出して pcap ファイルへ保存する．その後保存したファイルを ndpiReader へ入力し判別結果を得る．

```
30  TCP 192.168.99.12:80 <-> 172.22.2.138:57455
    [proto: 7.178/HTTP.Amazon]
    [4 pkts/467 bytes]
    [Host: www.amazon.co.jp:443]

31  TCP 192.168.99.12:80 <-> 172.22.2.138:57456
    [proto: 7.178/HTTP.Amazon]
    [4 pkts/473 bytes]
    [Host: fls-fe.amazon.co.jp:443]

32  TCP 192.168.99.12:80 <-> 172.22.2.138:57458
    [proto: 7.124/HTTP.YouTube]
    [4 pkts/457 bytes]
    [Host: i.ytimg.com:443]

33  TCP 192.168.99.12:80 <-> 172.22.2.138:57457
    [proto: 7.124/HTTP.YouTube]
    [4 pkts/457 bytes]
    [Host: i.ytimg.com:443]
```

図 9　Web ブラウザからのアクセスでキャプチャしたパケットの検査結果

結果のうち該当行を抽出したものが図 9 である．HTTP.Amazon や HTTP.Youtube とあるように，ポート 80 を介する通信でも実際に使用したサービスまで判別できていることがわかる．次に，既定のポートを変更してもプロトコルを判別できるか確認する．今回は例として本来ポート 22 を使う SSH を，仮想マシンで違うポートを介するように設定し SSH 接続を行ってパケットを確認する．概要を図 10 に示す．

[*1] http://www.ntop.org/products/deep-packet-inspection/ndpi/

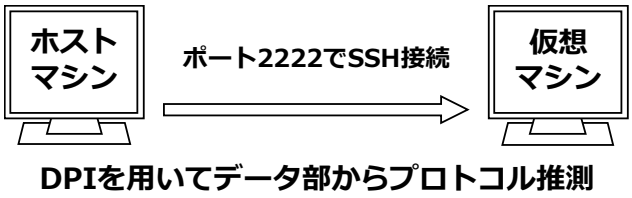

図 10　プロトコル推定の実験概要

実装に備え ndpiReader でパケットを直接キャプチャするライブキャプチャモードを利用する．仮想マシンからホストマシンへ接続するためにネットワークの設定を行う．仮想マシンから外部ネットワークへの接続用に NAT，ホストマシンから仮想マシンへの接続用にホストオンリーアダプタを利用する．このため VirtualBox の仮想マシン設定のネットワーク項にて，アダプタ 1 を NAT，アダプタ 2 をホストオンリーアダプタとして設定しておく．仮想マシンを起動して/etc/sysconfig/network-scripts に ifcfg-eth1 というファイルを作成し，図 11 のように記述し再起動する．HWADDR にはホストオンリーアダプタに割り当てられている MAC アドレスを設定する．NETWORK には VirtualBox のホストオンリーネットワークのネットワークアドレス，IPADDR にそのネットワークに属する IP アドレスを設定する．

```
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=static
HWADDR=**:**:**:**:**:**
NAME="eth1"
IPADDR=192.168.56.101
NETMASK=255.255.255.0
NETWORK=192.168.56.1
```

図 11　/etc/sysconfig/network-scripts/ifcfg-eth1 の記述内容

仮想マシンの/etc/ssh/sshd_config を編集することで SSH 接続ポートを任意のポートに変更することができる．このほか SELinux や firewalld にもこのポートで外部から接続できるように設定を行う．SELinux は図 12 に示すコマンドを実行することで設定できる．firewalld は設定ファイルである/usr/lib/firewalld/services/ssh.xml を/etc/firewalld/services/ssh.xml へコピーし port="22" の記述を書き換えることで開放するポートを変更できる．

```
semanage port -a -t ssh -p tcp （新しいポート番号）
```

図 12　SELinux へのルール追加コマンド

設定が完了したら SSH 接続を行ってパケット検査の結果を確認する．DPI には前項と同様に ndpiReader を利用し，対象のネットワークインターフェースを eth1 とする．ポート変更前後のパケットの比較・保存のために wireshark

Copyright (c) 2017 by the Information Processing Society of Japan

によるパケットキャプチャも行う．ユーザ認証を行い SSH 接続が確立されたらキャプチャしたパケットを確認する．

Source	Destination	Protocol	Info
192.168.56.101	192.168.56.1	SSH	Server: Protocol (SSH-2.0-
192.168.56.101	192.168.56.1	SSH	Server: Encrypted packet (
192.168.56.101	192.168.56.1	SSH	Server: Encrypted packet (
192.168.56.101	192.168.56.1	SSH	Server: Encrypted packet (
192.168.56.101	192.168.56.1	SSH	Server: Encrypted packet (

図 13　ポート変更前のパケットキャプチャ結果

Source	Destination	Protocol	Info
192.168.56.101	192.168.56.1	TCP	2222→59598 [SYN, ACK] Seq
192.168.56.101	192.168.56.1	TCP	2222→59598 [PSH, ACK] Seq
192.168.56.101	192.168.56.1	TCP	2222→59598 [ACK] Seq=24 A
192.168.56.101	192.168.56.1	TCP	2222→59598 [ACK] Seq=24 A
192.168.56.101	192.168.56.1	TCP	2222→59598 [PSH, ACK] Seq

図 14　ポート変更後のパケットキャプチャ結果

server ip	server port	host ip	host port	detected protocol	monitor
192.168.56.1	59598	192.168.56.101	2222	92/SSH	1

図 15　ポート変更後のパケット検査結果

ポート変更前のパケットキャプチャの結果を**図 13**，変更後の結果を**図 14** に示す．ポートを変更してから TCP となってしまい実際のプロトコルが判別できていないことがわかる．しかし ndpiReader でキャプチャした結果（**図 15**）では SSH と判別できていることが確認できた．

7. 検知システムの実装

前項より実際に DPI が行えていることが確認できたので，侵入検知システムを実装する．これまで利用してきた ndpiReader では画面上で見やすいように整形されていたが，管理性向上のために csv 形式で出力する．csv へ出力する項目は以下に示す通りである．

- 日付・時刻
- IP プロトコル
- 通信元の IP アドレス・ポート番号
- 通信先の IP アドレス・ポート番号
- 判別されたプロトコル
- 検知フラグ（検知対象であれば 1，そうでない場合は 0）

検知フラグは，検知対象のリストに記述されているプロトコルの通信が検知された場合に，区別するために立てるフラグである．この他，ndpiReader も参考に以下に示すオプションをプログラムに実装した．

-i < ∗.pcap もしくはネットワークインターフェース >
　パケットのソースを選択
-p < ファイル形式指定なし >
　フォーマットに則り追加プロトコル定義を記述したファイルを指定
-m < ∗.csv>
　検知対象のプロトコルをカンマ区切りで記述したファイルを指定
-w < ∗.csv>
　-i で指定された入力に対して DPI を行った結果の保存ファイルを指定（csv 形式）
-s < ∗.sh>
　検知対象のプロトコルの通信を検知した際に指定されたシェルスクリプトを実行
-a
　検知対象のプロトコルの通信を検知した際にデスクトップ通知を表示

出力ファイルへのフラグだけでなく，より動的な対処を行うためにシェルスクリプトの実行機能を付加した．また，デスクトップ通知には GNOME デスクトップ環境に標準で含まれているパッケージである zenity を利用した．

7.1 プロトコル検出の通知

SSH の通信ポートを標準である 22 から変更し，SSH 接続を検知対象として評価を行う．

```
sudo ./ndpiChecker -i eth1 -v 2 -w capture.csv -m allow.csv -a
```

図 16　リアルタイムにキャプチャしたパケットを検査し，結果を csv ファイルへ保存するコマンド

今回実行するコマンド（**図 16**）は，eth1 をキャプチャして実行結果を capture.csv へ保存するものである．また allow.csv に記述されたプロトコルによる通信を観測したらデスクトップ通知を表示し出力ファイルでフラグを立てる．

図 17　検知を知らせるデスクトップ通知

実際に実行し出力された csv ファイルを Excel で整理したものが**表 2** である．また表示されたデスクトップ通知が**図 17** である．本来ポート 22 を利用する SSH，23 を利用する Telnet のポートをそれぞれ変更してもプロトコルが判別されており，monitor 列でも SSH を検知できていることが確認できた．なお，実行結果はデータの取得順に昇順でソートしている．

7.2 プロトコル定義の追加

次に nDPI のプロトコル定義を追加できる機能を利用して，登録した任意のサービスの検知機能を評価する．ローカル環境で Web サーバを別に立ち上げ，そのサーバとの通信の検知を試みる．CentOS 7 をインストールした仮想

Copyright (c) 2017 by the Information Processing Society of Japan

表 2　リアルタイムにキャプチャしたパケットを検査した結果

ip protocol	server ip	server port	host ip	host port	detected protocol	monitor
TCP	192.168.56.1	55561	192.168.56.101	2222	92/SSH	1
TCP	192.168.56.1	55562	192.168.56.101	2323	77/Telnet	0
TCP	192.168.56.1	55564	192.168.56.101	2222	92/SSH	1
TCP	192.168.56.1	55565	192.168.56.101	2323	77/Telnet	0

表 3　実装したプログラムを実行した結果

ip protocol	server ip	server port	host ip	host port	detected protocol	monitor
TCP	192.168.56.1	58258	192.168.56.101	2222	92/SSH	0
UDP	192.168.56.1	137	192.168.56.255	137	10/NetBIOS	0
TCP	192.168.56.1	58259	192.168.56.101	2323	77/Telnet	0
TCP	192.168.56.101	60332	192.168.56.102	8080	223/MainService	1
TCP	192.168.56.1	58260	192.168.56.101	2222	92/SSH	0
TCP	192.168.56.101	60334	192.168.56.102	8080	223/MainService	1
TCP	192.168.56.1	58261	192.168.56.101	2323	77/Telnet	0

マシンを新たに作成し IP アドレスを 192.168.56.102 とする．この仮想マシンに Apache Web サーバをインストールする．バージョンは 2.4.6 を利用した．通常の HTTP 通信と切り分けるため Web サーバのポートを 8080 に設定する．実験の概要を図 **18** に示す．

DPI プログラムへ判別できるプロトコルを追加するため図 **19** に示す記入例に従ってプロトコルの定義ファイルを記述する．今回は Web サーバへの接続を MainService として登録する．また検知を確認するため検知対象リストに MainService を記述する．

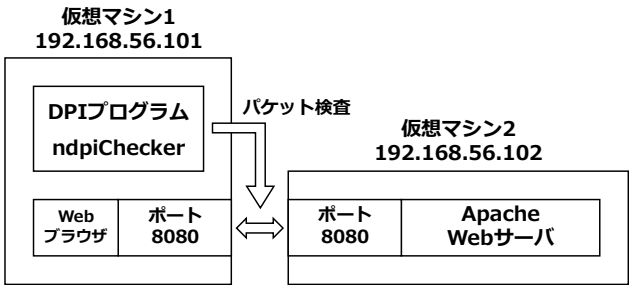

図 **18**　プロトコル定義を追加した実装プログラム評価実験の概要

```
# Subprotocols Format:
# host:"<address>",host:"<address>",.....@<subproto>
# <tcp|udp>:<port>,<tcp|udp>:<port>,.....@<subproto>
host:"192.168.56.102",tcp:8080@MainService
```

図 **19**　protos.txt の記述内容

```
sudo ./ndpiChecker -i eth1 -v 2
    -p protos.txt -w capture.csv -m allow.csv
```

図 **20**　作成したプログラムでキャプチャしたパケットを検査し，結果をテキストファイルへ保存するコマンド

記述したプロトコル定義ファイルを引数に与え図 **20** に示すコマンドを実行することで**表 3** を得た．新たに定義した任意のプロトコルの通信も検知できることが確認できた．

8.　まとめと課題

本論文では DPI を用いてパソコン，スマートフォン並びにその他のインターネット接続機器を包括的に保護するシステムの実装を行った．実装では，nDPI ライブラリを用いて DPI の簡易的な動作確認を行い，一般的な HTTP/HTTPS 通信でも利用サービスを特定できることが確認できた．また，本来特定のポートを利用する通信のポートを変更しても，ペイロード部の検査によりプロトコルを判別できることも確認できた．さらに，nDPI ライブラリを用いて特定のプロトコルによる通信を検知するシステムを実装し，新たに定義した任意のプロトコルの通信も検知できることが確認できた．

対応機器は急速に普及しているが比較的新しい分野であるため攻撃側有利な状況となっており，対策が急がれる．しかし，一般的に流通している IoT 機器製品は仕様が公開されることがないため，サービス提供だけでなく機器の正常動作のためにどの通信が必要なのかを判断することが困難である．通信制御にホワイトリスト制を用いると前述のように機器に必要な通信をすべて把握する必要があるが，ブラックリスト制を用いると日々進化する侵入や攻撃に対して後手の対応となり，セキュリティリスクが高まると考えられる．また多数の機器・サイトとの通信などによりパケットフィルタリングで用いるルールが複雑になり，スループットや管理性の低下を招く恐れがある．これらを踏まえ，通信フローは DPI で管理し，個別のパケットに対しては特徴量のようにプロトコルや暗号化の状態に依存しない手法を用いることによって，幅広い IoT 機器を包括的に保護可能なシステムが実現できると考えられる．本論文では通信検知による外部からのネットワークを介した侵入検知を目標としたが，今後の課題として，netfilter と他のラ

Copyright (c) 2017 by the Information Processing Society of Japan

イブラリやエンジンを利用することで，検知状況によってパケット転送を制御する侵入防御システムの開発が挙げられる．

参考文献

[1] Gartner: Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015 (2015). http://www.gartner.com/newsroom/id/3165317.

[2] 警察庁セキュリティポータルサイト「@police」: IoT 機器を標的とした攻撃の観測について (2015) https://www.npa.go.jp/cyberpolice/detect/pdf/20151215_1.pdf.

[3] JVN(JPCERT コーディネーションセンター 脆弱性対策情報): Mirai 等のマルウェアで構築されたボットネットによる DDoS 攻撃の脅威 (2016). https://jvn.jp/ta/JVNTA95530271/.

[4] cisco: IoT への脅威に関する状況 (2016). https://www.cisco.com/c/dam/global/ja_jp/products/collateral/se/internet-of-things/C11-735871.pdf.

[5] IPA: ネットワークカメラや家庭用ルータ等の IoT 機器は利用前に必ずパスワードの変更を (2016). https://www.ipa.go.jp/security/anshin/mgdayori20161125.html.

[6] 警察庁セキュリティポータルサイト「@police」: インターネット観測結果等（平成 28 年 9 月期） (2016). http://www.npa.go.jp/cyberpolice/detect/pdf/20161020.pdf.

[7] 高山祐磨，越島一郎，青山友美: IoT 時代における制御システムサイバーセキュリティ，信学技報 116(131), 65-70, (2016).

[8] 三浦功也，本田晋也，高田広章: 組込みシステム向け仮想環境を用いた汎用 OS の監視機能，情報処理学会　研究報告データベースシステム（DBS） 2014-DBS-160(3), 1-8, (2014).

[9] 仲小路博史，鬼頭哲郎，重本倫宏，寺田真敏，石山智祥: Winnyp 通信検知機能の実装および評価，情報処理学会論文誌 52(1), 2-13，(2011).

[10] 長嶺優大，矢内直人，岡村真吾，藤原融: MASQWARE: 通信プロトコルを偽装するマルウェア，情報処理学会コンピュータセキュリティシンポジウム 2016 論文集, pp.655-661，(2016).

Copyright (c) 2017 by the Information Processing Society of Japan