L-012

スマートホームシステムにおける Dempster-Shafer Theory に基づく異常検知手法の提案
Proposal of Intrusion Detection Method based on Dempster-Shafer Theory for Smart Home Systems

竹田 拓哉[1)]　河野 浩之[2)]　石原 靖哲[2)]
Takuya Takeda　Hiroyuki Kawano　Yasunori Ishihara

1 はじめに

現在，IoT デバイスの増加に伴い，不正アクセスや乗っ取りといった事件が発生しセキュリティ強化が課題とされている．2016 年 9 月には IoT デバイスを用いた大規模な DDoS 攻撃が実行された．また，スマートホームシステムの脆弱性も課題として挙げられており，2016 年 8 月に行われた「DEF CON」では 23 製品から 47 の脆弱性が発見された．

本研究では，スマートホームシステムに対し不正アクセスが発生した場合に Dempster-Shafer Theory(DST) に基づいた侵入検知システム (Intrusion Detection System, IDS) の検知手法を提案する．この研究での不正アクセスは，IoT 機器を遠隔から不正に操作する攻撃を指す．こうした不正アクセスの例として，ユーザーが操作するアプリケーションと IoT 機器との通信をスヌーピングし，得た情報から攻撃スクリプトを作成し，実行するなどがある．この時，平均パケット到着間隔，要求パケットバイト数，応答パケットバイト数の 3 つのパラメータの値が正常な通信と異なることが，Nobakht ら [5] によって報告されている．

そこで，3 つのパラメータそれぞれで危険度を判定し，独立した判定結果を DST により組み合わせることで，偽陰性率・偽陽性率を低い値に抑え，精度の高い検知手法を提案する．以下，パラメータ毎の危険度を統合する提案手法を RP-DST(Risk Parameters-Dempster-Shafer Theory) 手法とする．

2 章では，先行研究として，IoT 機器の脆弱性，IDS のアルゴリズム，DST に基づく IDS について述べる．3 章では証拠理論について，また DST の組み合わせ理論について述べる．4 章では，本研究の想定する環境について説明し，提案する RP-DST 手法について説明する．5 章では，性能評価に用いるデータセット，RP-DST 手法の実装，実験環境を述べ，RP-DST 手法の性能を議論する．6 章ではまとめを行う．

2 IoT 機器と異常検知アルゴリズム

2.1 節では，IoT 機器の脆弱性について述べる．2.2 節では，IDS の異常検知アルゴリズムについて述べる．2.3 節では DST に基づく IDS について説明する．

2.1 IoT 機器の脆弱性

IoT 技術の拡大の妨げとなっているのが，脆弱性である [10]．Denning らは，スマートホームシステムの潜在的な脆弱性を調査し，今後のセキュリティニーズの提案を行なっている [2]．

IPA の報告によると，スマートハウスにおける各機器の情報を不正に取得された場合，個人のプライベートデータの漏えいという脅威に加えて，取得したデータ（消費電力や施錠状況）を悪用して，不在と分かった場合に遠隔操作で施錠解除し，家屋に侵入される恐れがある．スマートホームシステムの脆弱性により，利用者に不利益が被る可能性があるため，こうした攻撃を防ぐ必要がある [9]．

2.2 異常検知アルゴリズム

IDS では，大量のデータを分析することで，通常のパターンとは異なる振る舞いを検知する．

Pajouh らは，U2R や R2L などの攻撃に対し，次元削減モジュールのコンポーネント分析と線形判別分析を使用して，高次元データセットを低次元に変更し，Naive Bayes と K-Nearest Neighbor を使用して，疑わしい行動を識別する 2 段階分類モジュールを適用している [6]．

スマートホームシステムの環境において，OpenFlow を用いた IDS の研究も進められている．Nobakht らは，スマートホームシステムへの不正アクセスの事例を紹介し，IoT-IDM と呼ばれる異常検知，軽減フレームワークを提案している．不正アクセスには，コマンドパケット（要求パケット）バイト数，応答パケットバイト数，平均パケット到着間隔が有効であることを説明し，ロジスティック回帰と非線形分類モデルでの異常検知の精度比較を行っている．

このように多くの IDS アルゴリズムの提案がされている中，誤検知が少ない事で知られる DST について 2.3 節で述べる．

2.3 DST に基づく異常検知アルゴリズム

DST は 3 値論理を採用し，Data Fusion 技術の 1 つとして複数のデータを組み合わせる事で新たな知見を見出す理論である．異常検知アルゴリズムに DST を用いることで不確実さを定量的に表すことができるため，FP(False Positive) と FN(False Negative) を減らし，低い偽陽性率と偽陰性率を実現できる．近年の IDS に Data Fusion を採用した異常検知アルゴリズムの研究は，Li らがまとめている [3]．

DST を IDS に用いた例として，Madalina らは，クラウド環境で偽陰性率と偽陽性率を低下させ，フォルトツリーで原因を分析し，精度の高い IDS を実現している [4]．また Aparicio-Navarro らが提案した IDS では，PoL(Pattern-of-Life) という DST を含んだ手法を提案し，異常検知を行なっている [1]．

3 DST の概要

3.1 節で，DST の計算方法について記述し，3.2 節で DST の組み合わせ規則を説明する．

2) 南山大学理工学研究科機械電子制御工学専攻 Graduate Program of Mechatronics, Nanzan University
3) 南山大学理工学部 Department of Mechatronics, Nanzan University

3.1 証拠理論

ベイズ推定の拡張として，統計的推論，診断，リスク分析，決定分析に応用される．DST ではある命題を支持するとき，「情報や証拠の不確実さ」を定量的に表すことができる [11]．複数の独立した情報から得た情報が支持する命題を結合することで確信度を割り当てる．一つの証拠に基づく仮説を支持する確率を割り当てる機能を基本確率割当 (basic probability assignment) と呼び，m と表す．

通常は T または F の 2 属性に対し値が与えられるが，DST では $\emptyset$，T，F，$\{T,F\}$ の 4 属性となる．$\{T,F\}$ は T であるか F であるか判別できない *unknown* を表している．m の計算方法は，Siaterlis ら [8] の手法を用いた．

次のような命題の有限集合 $X=\{x_1,x_2,\cdots,x_n\}$ が次の 2 つの性質を持つとき，X を識別空間とよび，x_i を基本命題と呼ぶ．

1. $x_1,\cdots,x_n$ は互いに排他的である．
2. $x_1,\cdots,x_n$ のうち，1 つのみ T であり，それ以外は F である．

例えば $\{x_i,x_j\}\in 2^X$(2^X は X のべき集合) は，$x_i\vee x_j$ とする．これら命題に対し，確信度の度合いを表現するための関数 m:$2^X\rightarrow[0,1]$ を，(1)~(3) 式の公理を満たすように定義する．

$$m(A)\geq 0,\forall A\in 2^X \tag{1}$$

$$m(\emptyset)=0 \tag{2}$$

$$\sum_{\forall A\in 2^X} m(A)=1 \tag{3}$$

本来であれば，集合関数である m は $m(\{A\})$ と表記する必要があるが，誤解のない範囲で簡略化し $m(A)$ と表す．$m(F)$ は次式で求める．

$$m(F)=1-m(T)-m(T,F) \tag{4}$$

(4) 式より $m(T)$，$m(F)$，$m(T,F)$ の和が 1.0 になる．

3.2 DST の組み合わせ規則

DST では，複数の独立した知識源から信念構造を結合する事を情報統合と呼ぶ．$m_1(A):A\in 2^X$ と $m_2(B):B\in 2^X$ が存在するとき，情報を組み合わせる事で，二つの値の矛盾を確率で表すことができる．統合は $m=m_1\oplus m_2$ と表す．ここで予備割当関数 q により 2 つの情報を m_1，m_2 が支持する命題の論理積に対する，確信度を割り当てる．予備割当関数は (5) 式で表される．

$$q(E)=\sum_{A\cap B=E} m_1(A)m_2(B) \tag{5}$$

m_1 と m_2 の AND 演算にて m に確率を割り当てる．表 1 に m_1 と m_2 の論理積を示す．

次に q から m への変換を行う．$q(\emptyset)$ は 2 つの状態 m_1 と m_2 にどの程度矛盾があるかを示す．一般には $m(\emptyset)=0$ であるため，q が m と一致しない場合がある．そこで DST の規則を用いて，q から m に変更する必要がある．

表 1　m_1 と m_2 の論理積

AND	$m_1(T)$	$m_1(F)$	$m_1(T,F)$
$m_2(T)$	$q(T)$	$q(\emptyset)$	$q(T)$
$m_2(F)$	$q(\emptyset)$	$q(F)$	$q(F)$
$m_2(T,F)$	$q(T)$	$q(F)$	$q(T,F)$

$$m(C)=\frac{q(C)}{1-q(\emptyset)},C\neq\emptyset \tag{6}$$

(6) 式は，$q(C)$ を $1-q(\emptyset)$ で正規化することにより，$\emptyset$ に割り当てられた $q(\emptyset)$ を $\emptyset$ 以外に比例配分することを示している．

4 RP-DST 手法の提案

4.1 節で IDS を含むスマートホームシステムの想定環境について説明し，4.2 節で RP-DST の提案を行う．

4.1 スマートホームシステムの環境

IDS を含むスマートホームシステムの想定環境の概要を図 1 に示す．ユーザーが携帯端末等を用いて遠隔から IDS を含む OpenFlow を介してスマートホームシステム内にある IoT 機器を操作する状態を表している．OpenFlow は，SDN(Software Define Network) の 1 つとして注目されており，制御部と転送部を分離したアーキテクチャを採用する事により, 柔軟にネットワーク構成変更できる技術である．制御部を OpenFlow コントローラ，転送部を OpenFlow スイッチと呼ぶ．

OpenFlow コントローラは，パケットの条件毎にアクション（破棄，送信など）を定義したフローテーブルを作成し，OpenFlow スイッチに書き込む．OpenFlow スイッチはパケットが到着するたびにフローテーブルを参照することで，OpenFlow コントローラに問い合わせする必要がなくなり高速に処理できる．

フローテーブルに条件がないパケットが到着した場合，OpenFlow コントローラはそのパケットデータを元に，新たに条件を作成しなくてはならない．パケットを正常な通信として IoT 機器へ送信するか，不正な通信として破棄するかという条件をフローテーブルに書き込まなくてはならないため，パケットデータを IDS を用いて分析し，異常か判定する必要がある．

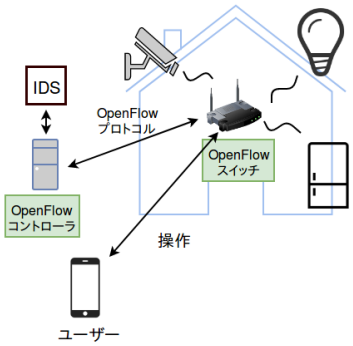

図 1　スマートホームシステムの概要図

この環境での IDS は，ホスト型 (HIDS) に分類され，暗号化された通信においても同様に異常検知を行うことができる．

4.2 RP-DST 手法の計算アルゴリズム

3.2 節で説明した組み合わせ規則を用いた CoMR 手法を用いる．平均パケット到着間隔を m_a，要求パケットバイト数を m_b，応答パケットバイト数を m_c とする．

はじめに m_a と m_b から (7)~(10) 式を用いて m_{ab} を計算する．

$$q_{ab}(\phi) = m_a(T)m_b(F) + m_a(F)m_b(T) \tag{7}$$

$$m_{ab}(T) = \frac{m_a(T)m_b(T) + m_a(T)m_b(T,F) + m_a(T,F)m_b(T)}{1 - q_{ab}(\phi)} \tag{8}$$

$$m_{ab}(T,F) = \frac{m_a(T,F)m_b(T,F)}{1 - q_{ab}(\phi)} \tag{9}$$

$$m_{ab}(F) = \frac{m_a(F)m_b(F) + m_a(F)m_b(T,F) + m_a(T,F)m_b(F)}{1 - q_{ab}(\phi)} \tag{10}$$

(7) 式より $q(\phi)$ を計算し，(8)~(10) 式では q を $q(\phi)$ で正規化することで，$q(\phi)$ 以外の焦点集合に値を分配したことを表している．同様に (11)~(14) 式を用いて m_{ab} と m_c から m_{abc} を計算する．

$$q_{abc}(\phi) = m_{ab}(T)m_c(F) + m_{ab}(F)m_c(T) \tag{11}$$

$$m_{abc}(T) = \frac{m_{ab}(T)m_c(T) + m_{ab}(T)m_c(T,F) + m_{ab}(T,F)m_c(T)}{1 - q_{abc}(\phi)} \tag{12}$$

$$m_{abc}(T,F) = \frac{m_{ab}(T,F)m_c(T,F)}{1 - q_{abc}(\phi)} \tag{13}$$

$$m_{abc}(F) = \frac{m_{ab}(F)m_c(F) + m_{ab}(F)m_c(T,F) + m_{ab}(T,F)m_c(F)}{1 - q_{abc}(\phi)} \tag{14}$$

このように組み合わせ規則を用いて，求めた m_{abc} を危険度の判定に用いる．

5 RP-DST 手法の実装と性能評価

この章では，4 章で提案した RP-DST 手法を実装し，性能評価を行う．5.1 節はデータセットについて，5.2 節はデータセットの前処理について説明する．5.3 節で実装した異常検知手法を説明し，5.4 節で実験結果を述べる．

5.1 異常検知データセット

ネットワーク異常検知にまつわるデータとして機械学習の評価セットに使用される NSL-KDD dataset を説明する．protocol_type や service などの 41 の特徴値とラベルが存在する．クラス (ラベル) は normal，Probe，DoS，U2R，R2L に分類されており，そのサブクラスが表記されている．例えば，Probe 攻撃は，ipsweep や nmap，postsweep などである．このデータセットの中からトレーニングデータは KDDTrain+.arff，テストデータは KDDTest+.arff を使用する．

本来であれば，スマートホームシステムへの不正アクセス時のパケットデータから作成されたデータセットを用いることが望ましいが，適切なデータセットが存在しなかったため，NSL-KDD dataset を用いる．

5.2 データセットの前処理

DoS 攻撃（back 攻撃，smurf 攻撃，neptune 攻撃）は，これまで説明してきた不正アクセスと同じように平均パケット到着間隔，要求パケットバイト数，応答パケットバイト数から異常検知を行うことができる．よって NSL-KDD dataset においてサブクラスが back，smurf，neptune に分類された anomaly データと normal データを合わせて，性能評価用のトレーニングデータとテストデータを作成する．

トレーニングデータは，KDDTrain+.arff から back 攻撃，smurf 攻撃，neptune 攻撃ごとに別々に anomaly データを抽出し，KDDTrain+.arff から抽出した normal データを合わせる事で 3 種類作成する．anomaly データ数は，back が 956，neptune が 10000，smurf が 2646 である．また，normal データと anomaly データの数の比が 1:1，2:1，3:2，となるように各攻撃ごとに 3 つずつ作成したため，トレーニングデータは計 9 ファイルである．

またテストデータは，KDDTest+.arff から，smurf 攻撃，back 攻撃，neptune 攻撃ごとに anomaly データを抽出し，KDDTest+.arff から抽出した normal データと合わせる事で攻撃毎の 3 つのファイルを作成する．

5.3 RP-DST 手法の実装

まず平均パケット到着間隔における危険度を測定する．測定方法は，5.1 節で述べたサブクラスをラベルとし，あるテストデータに対して，トレーニングデータをベイズ推定で分析し，$normal$ の確率と $anomaly$ の確率を算出する．3.1 節で述べた方法を用いて $m_a(normal)$，$m_a(anomaly)$，$m_a(normal,anomaly)$ に変換する．$normal$ と $anomaly$ の確率が 50% に近い値ほど，$m(normal,anomaly)$ が高くなるよう設定し，最大値は 0.25 とした．m_b，m_c も同様に，それぞれのパラメータにおける m を算出する．

4.2 節で説明した組み合わせ規則を用いて，m_a，m_b，m_c から m_{abc} を計算する．得られた $m_{abc}(normal)$，$m_{abc}(anomaly)$ の値を比較し，$m_{abc}(normal)$ の方が大きい値であれば通常の通信，$m_{abc}(anomaly)$ の方が大きい値であれば攻撃と判定する．

RP-DST 手法の実装は，java でコーディングされた 810 行の異常検出プログラム [7] の DST 部分（約 200 行）を用いて，合計で約 400 行のプログラムに書き変えた．

5.4 RP-DST 手法の性能評価

前処理したデータセットを使用し，RP-DST 手法の DoS 攻撃の性能評価を行なった．表 2 に実験環境を示す．

前処理したデータセットの中から使う特徴値は，count，dst_bytes，src_bytes である．これらは，NSL-KDD dataset において，平均パケット到着間隔，要求パケットバイト数，

表 2　実験環境

OS	Ubuntu14.04
プロセッサ	AMD FX-8350 Eight-Core
メモリ	16GB
使用言語	java

応答パケットバイト数に該当する特徴値である．表 3 に m と対応するデータセットの特徴値を示す．

表 3　m の表記と対応するデータセットの特徴値

m	データセットの特徴値
m_a	count
m_b	dst_bytes
m_c	src_bytes

データセットにおける選定した特徴値の概要を説明する．count は 2 秒間に同じホストへ接続された通信の回数を示し，dst_bytes と src_bytes はそれぞれ同一ホストから送受信されたパケットのバイト数を表す．RP-DST 手法の TP(True Positive)，FP(False Positive)，TN(True Negative)，FN(False Negative) を算出し，smurf 攻撃，back 攻撃，neptune 攻撃への性能評価を表 4，表 5，表 6 に示す．

表 4　smurf 攻撃の性能評価

normal:anomaly	偽陽性率	偽陰性率
1:1	0.0%	2.1%
3:2	36.2%	0.0%
2:1	100.0%	0.0%

表 5　back 攻撃の性能評価

normal:anomaly	偽陽性率	偽陰性率
1:1	0.6%	0.1%
3:2	0.3%	0.5%
2:1	4.7%	0.1%

表 6　neptune 攻撃の性能評価

normal:anomaly	偽陽性率	偽陰性率
1:1	0.0%	5.7%
3:2	100.0%	1.5%
2:1	100.0%	0.0%

表 7　Recall と Precision の性能評価

精度	smurf	back	neptune
Recall	100.0%	100.0%	100.0%
Precision	99.2%	99.9%	95.2%

どの攻撃においても normal と anomaly の比が 1:1 であるとき，最も良い精度を示した．back 攻撃のみ 2:1 の比率であっても良い精度を示したが，その他の攻撃では，偽陽性が 100%となった．

normal:anomaly の比が 1:1 のトレーニングデータを用いたときの Recall と Precision の結果を表 7 に示す．Recall は smurf 攻撃，back 攻撃，neptune 攻撃いずれの攻撃においても 100%になった．Precision は 95%以上となった．

6　おわりに

本研究では，RP-DST 手法により，平均パケット到着間隔，応答パケットバイト数，要求パケットバイト数からそれぞれの危険度を測定し，測定結果を組み合わせることで低い偽陰性と偽陽性を実現した．トレーニングデータの normal と anomaly の比は，1:1 の場合がいずれの攻撃でも高い精度だったが，トレーニングデータの normal データの数が増えると精度は低下した．ベイズ推定において事前確率が変化し，検知に影響を与えた可能性がある．

今後の課題として，実際のスマートホームシステム環境で RP-DST 手法を用いた IDS を構築し，Nobakht ら [5] の構築した IoT-IDM との異常検知における性能評価を行う予定である．Raspberry Pi を IoT 機器として設置し，スマートフォンからの遠隔操作時に，RP-DST 手法を採用した IDS を含む OpenFlow のネットワークで接続する環境での有効性を示す実験を行う．

謝辞

本研究は JSPS 科研費 JP17K00432 の助成を受けたものである．

参考文献

[1] Aparicio-Navarro, F. J. et al.: Using Pattern-of-Life as Contextual Information for Anomaly-Based Intrusion Detection Systems, *IEEE Access*, Vol.5, pp.22177–22193 (2017).

[2] Denning, T., Kohno, T. and Henry M., L.: Computer Security and the Modern Home, *Commun. ACM*, Vol.56, No.1, pp.94–103 (2013).

[3] Li, G. et al.: Data Fusion for Network Intrusion Detection: A Review, *Security and Communication Networks* (2018).

[4] Lonea, A. M. et al.: Detecting DDoS Attacks in Cloud Computing Environment, *International Journal of Computers Communications Control*, Vol.8, pp.70–78 (2012).

[5] Nobakht, M. et al.: A Host-based Intrusion Detection and Mitigation Framework for Smart Home IoT using OpenFlow, *2016 11th International Conference on Availability, Reliability and Security*, pp.147–156 (2016).

[6] Pajouh, H. H. et al.: A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks, *IEEE Transactions on Emerging Topics in Computing*, pp.1–1 (2016).

[7] Shakti, S.: shakti365/IDS-PRFCM-Clustering (accessed Jun. 2018). https://github.com/ shakti365/IDS-PRFCM-Clustering.

[8] Siaterlis, C. et al.: A Novel Approach for a Distributed Denial of Service Detection Engine, *NETMODE Lab Department of Electrical and Computer Engineering*, National Technical University of Athens (2003).

[9] 技術本部セキュリティセンター: IoT 開発におけるセキュリティ設計の手引き, 技術報告, 独立行政法人情報処理推進機構 (accessed Jun. 2018). https://www.ipa.go.jp/security/iot/iotguide.html.

[10] 井上博之: IoT（つながる組込み機器）における脅威の現状, 精密工学会誌, Vol.83, No.1, pp.46–49 (2017).

[11] 稲垣敏之, 伊藤誠: 証拠理論, 日本ファジィ学会誌, Vol.10, No.3, pp.445–450 (1998).