IoTゲートウェイで動作するコンテナの異常検知手法の提案

西村　賢太[1,a)]　山本　萌花[1]　掛井　将平[1]　瀧本　栄二[2]　毛利　公一[2]　齋藤　彰一[1]

概要：
IoT の広がりとともにデバイスを狙ったサイバー攻撃が増加しており，IoT の安全性向上が重要性を増している．現在，利便性と安全性の観点から IoT ゲートウェイでコンテナを利用するモデルが注目されている．しかし，IoT ゲートウェイがサイバー攻撃を受けやすい点と、コンテナ提供者とホスト OS 提供者が異なる点からホスト OS が信頼できるとは限らないという懸念が存在する．そこで本稿では，IoT ゲートウェイの信頼できないホスト OS 上で動作するコンテナのセキュリティを向上させるために，コンテナの異常検知機構を提案する．提案手法では，コンテナの発行するシステムコールを監視し，ハードウェアによる保護領域内で異常検知を行う．さらに，異常検知手法として隠れマルコフモデルを用いた手法とオートエンコーダを用いた手法を複数検討し，コンテナの異常検知機構の提案を行う．

キーワード：IoT, コンテナ, Intel SGX, HIDS, 異常検知

Anomaly Detection System for Container on IoT Gateway

Kenta Nishimura[1,a)]　Moeka Yamamoto[1]　Shohei Kakei[1]　Eiji Takimoto[2]　Koichi Mouri[2]
Shoichi Saito[1]

Abstract: Cyber attacks targeting IoT devices are increasing with a spread of IoT, hence improving the safety of IoT is becoming important increasingly. Currently, containers in IoT gateways is to attract attention from the viewpoint of convenience and safety. However, because IoT gateway is infectable with cyber attacks, and container provider and OS providers are different, there is concern that the host OS is not always reliable. In order to improve the security of containers running on untrusted host OS, we propose a mechanism for detecting anomalies in containers. In the proposed method, system calls issued by the container are monitored, and anomalies are detected in a hardware protected area. In addition, we examined several methods using hidden Markov model and methods using Auto Encoder as anomaly detection methods.

Keywords: IoT, Container, Intel SGX, HIDS, Anomaly Detection

1. はじめに

近年，IoT を狙ったサイバー攻撃が増加しており，IoT の安全性向上が重要性を増している [1]．また，利便性と安全性の観点から，IoT 機器がインターネットに接続する際の中継を担う IoT ゲートウェイの機能をコンテナで提供するモデルが注目されている [2]．コンテナを利用することで，IoT ゲートウェイの機能をホスト OS に依存せずに実現できる他，アプリケーションごとに隔離された環境での運用が可能であるため，ホスト OS 上でアプリケーションを運用するモデルと比べ，安全性の向上が期待できる [3]．

しかし，IoT ゲートウェイでコンテナを運用する場合，コンテナが従来から抱えている安全性の問題の他，新たな問題も考慮する必要がある．コンテナが抱えている安全性の問題として，不正アクセスによる情報窃取などランタイムの安全性に関わる問題が存在する．また，ホスト OS が信頼できるとは限らない場合，ホスト OS 上で動作するコンテナも信頼できるとは限らないという問題が存在す

[1] 名古屋工業大学
Nagoya Institute of Technology
[2] 立命館大学
Ritsumei University
[a)] ckb15106@nitech.jp

る [4]．次に，IoT ゲートウェイでコンテナを運用する場合には，一般的に IoT はインフラ構成が複雑である点，管理のためのガイドライン及びフレームワークが不足している点から，サイバー攻撃を受けやすいという問題が存在する [5]．このことから，IoT ゲートウェイはホスト OS が攻撃を受ける可能性が高く，ホスト OS が信頼できるとは限らない．そのため，コンテナの安全性を保証するためにはホスト OS による安全性検証のみでは不十分であるという問題が存在する．

信頼できないホスト上のコンテナの安全性検証手法として，TapCon[4] が提案されている．TapCon では，ハードウェアによる保護領域を起点としたトラストチェイン機構により，コンテナイメージの安全性とデプロイの安全性を検証することを可能にしている．文献 [6] では，コンテナのランタイムの安全性の向上手法として，コンテナの発行するシステムコールを監視し，コンテナの異常検知を行う手法が提案されている．この手法では，クラウド上で動作するコンテナが発行するシステムコールを観測し，観測されたシステムコールを Bag of System Call (BoSC) としてまとめ，正常動作時の BoSC を機械学習により学習し，運用段階では BoSC の変化を観測することでコンテナの異常動作を検知している．これらの手法はコンテナをクラウドで運用することを想定した研究であるが，IoT ゲートウェイ上のコンテナの安全性の向上にも応用できる手法である．本稿ではこれらの手法を応用して IoT ゲートウェイ上のコンテナの安全性を向上させるシステムを提案する．

本稿では IoT ゲートウェイで運用されるコンテナの，特にランタイムの安全性を向上するために，コンテナの異常検知機構を提案する．提案手法では，コンテナの発行するシステムコールを監視し，ハードウェアによる保護領域内で異常検知を行う．保護領域内で異常検知を行うことで，異常検知プログラム自体の安全性を高めることができる．また，コンテナという多様なプロセスが動作する環境に対する異常検知方法を実現するにあたり，機械学習による検知を取り入れる．本検討の初期段階として，隠れマルコフモデルによる手法とオートエンコーダによる手法の評価を行う．評価ではオープンソース監視カメラアプリケーションの運用を想定したコンテナの異常検知を行う実験を行い，各手法の異常検知率について比較検討を行う．

以後，本稿では 2 章で関連研究について述べ，3 章で機械学習によるシステムコール列の異常検知の検討について述べ，4 章で提案手法について述べる．続いて 5 章で提案手法の実装について述べ，6 章で実装の評価及び検討について述べる．

2. 関連研究

本章では保護領域を作成する技術である Intel SGX について述べ，Intel SGX を起点とした信頼できないホスト上におけるコンテナの安全性検証の関連研究について述べる．次に，システムコールから異常を検知する研究について，IoT アプリケーションを対象とした関連研究について述べる．

2.1 ハードウェアによる保護技術

ハードウェアによる保護領域を作成し，安全なコード実行を可能にする技術として，Intel SGX[7] がある．Intel SGX は Intel CPU の命令セットの一つであり，プロセスがメモリ上に Enclave と呼ばれる保護領域を作成することができる．Enclave はメモリ上では暗号化されており，実行時には CPU 内で復号されるため，平文がメモリ上にロードされることはない．

また，Intel SGX を用いてコンテナの安全性を高める研究に SCONE [8] がある．SCONE では，コンテナのインスタンスを Enclave 内に置くことにより，信頼できないホスト上におけるコンテナの安全性を向上させることを可能にしている．

2.2 信頼できないホスト上におけるコンテナの安全性検証に関する研究

信頼できないホスト OS 上におけるコンテナの安全性検証手法として，TapCon[4] が提案されている．TapCon では，信頼できないホスト OS 上においても SCONE は安全性が保たれることを利用し，SCONE を安全性の保証の起点としたコンテナの安全性検証手法を提案している．

TapCon では，まず，SCONE からコンテナが運用されるホスト OS を認証する．次に，認証されたホスト OS からコンテナイメージとデプロイ情報を保護領域に取得し，イメージ及びデプロイの安全性を検証する．これにより，信頼できないホスト OS 上におけるコンテナイメージ及びデプロイの安全性を検証する．

TapCon で提案されたハードウェアによる保護領域を起点としたコンテナの安全性検証は，信頼できないホスト OS 上で運用されるランタイムのコンテナの異常検知にも応用できる．

2.3 システムコールによる異常検知に関する研究

ホスト内の処理を監視することでランタイムのプロセスまたはアプリケーションの異常検知及び攻撃検知を行うシステムとして，Host-based Intrusion Detection System (HIDS)[9] がある．本節ではシステムコールを監視して異常検知を行う Bag of System Call (BoSC)[10] を用いた関連研究について述べる．

BoSC は，システムコールを監視し，異常検知を行う手法に用いられるアプローチである．BoSC では，監視対象のシステムコール列をウィンドウに区切り，ウィンドウ内に含まれるシステムコールのシーケンスをシステムコールの頻

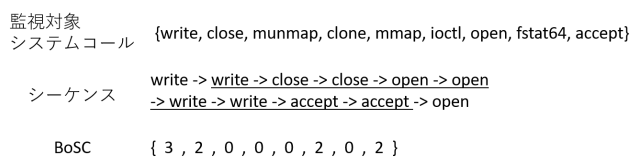

図 1 BoSC の例
Fig. 1 Example of BoSC.

度のベクトルに変換する．BoSC の例を図 1 に示す．監視するシステムコールとして write，close，munmap，clone，mmap，ioctl，open，fstat64，accept を選択した場合を考える．システムコールが図中のシーケンスのように観測された場合，下線部のシーケンスに対する BoSC はウィンドウ中で現れるシステムコール数のベクトルとなる．

文献 [11] では，IoT アプリケーションの発行するシステムコールを取得し，正常動作時の BoSC を k 平均法で学習し，運用段階では観測された BoSC と初期学習で得られた BoSC の平均値のマハラノビス距離を計算することで異常検知を行っている．また，IoT アプリケーションとして，オープンソース監視カメラアプリケーションである motion[12] を想定し，アプリケーションの動作による BoSC の変化を評価している．

3. 機械学習によるシステムコール列の異常検知の検討

本章では HIDS によるコンテナの異常検知方式における機械学習の活用について検討する．異常検知に用いるアルゴリズムとして隠れマルコフモデルによる異常検知とオートエンコーダによる異常検知を検討する．本章では隠れマルコフモデルによる異常検知とオートエンコーダによる異常検知について概要を述べる．

3.1 隠れマルコフモデルによる異常検知

隠れマルコフモデルによる HIDS が提案されている [9]．隠れマルコフモデルとは，時系列データの確率モデルであり，多くのアプリケーションで使用されてきた機械学習手法である．隠れマルコフモデルは有限個の隠れ状態を持ち，隠れ状態の遷移は状態遷移確率として定義され，有限確率オートマトンのようなマルコフ鎖を形成する．それぞれの状態では，その状態に依存する出力が観察される．観察される出力の出力確率は，それぞれの状態において確率分布として定義される．

隠れマルコフモデルでは，まず学習系列から時系列データを発生させているモデル M を作成する．モデルの作成にはバウムウェルチアルゴリズム [13] として知られる EM アルゴリズム [14] による最尤推定が用いられる．モデル M を用いる事により，観測系列の時系列データがどれほどモデル M に則しているかを尤度として計算することができる．これは，観測系列の時系列データがモデル M から生成されたと仮定した場合の出力確率を計算することで行われる．隠れマルコフモデルを異常検知に用いる場合，モデル M に対する対象の時系列データの出力確率を計算するために尤度が用いられる．対象の時系列データの尤度が高ければ，モデル M から対象の時系列データが得られる確率が高いと判断し，正常であると判定する．また，対象の時系列データの尤度が低ければ，モデル M から対象の時系列データが得られる確率が低いと判断し，異常であると判定する．

3.2 オートエンコーダによる異常検知

オートエンコーダによる HIDS が提案されている [15]．オートエンコーダとは，教師なし学習を行う深層学習手法であり，学習データと近い値の再構築を行うモデルを生成する．オートエンコーダは式 (1) で表されるエンコーダと式 (2) で表されるデコーダの二つの要素から構築される．

$$h = \sigma(W_{xh}x + b_{xh}) \tag{1}$$

$$z = \sigma(W_{hx}h + b_{hx}) \tag{2}$$

$$\| x - z \| \tag{3}$$

式 (1) で表されるエンコーダは入力ベクトル x を隠れ層にマッピングする．このとき，入力ベクトル x はアフィン変換により隠れ層ベクトル h に変換される．式 (2) で表されるデコーダは隠れ層ベクトル h をエンコーダと同じアフィン変換を用いることで入力ベクトル x と同じベクトル空間に変換する．もとの入力ベクトル x と再構築されたベクトル z の差 (3) は再構築誤差と呼ばれ，オートエンコーダは再構築誤差を最小にするように学習が行われる．

オートエンコーダを異常検知に用いる場合，まず，オートエンコーダに正常データを学習させ，正常データしか再構築できないモデル M を構築する．モデル M に異常データが入力された場合，再構築された出力データは入力データと大きく異なるものになるため，入力データと出力データの差を見ることで異常を判定することができる．

4. 提案

本章では提案手法についての概要を述べ，ハードウェア保護領域での異常検知方式，機械学習によるシステムコール列の異常検知方式について述べる．

4.1 概要

コンテナの発行するシステムコールを監視し，Intel SGX による保護領域内で異常検知を行う機構を提案する．また，異常検知アルゴリズムの比較検討を行う．IoT ゲートウェイ上のコンテナの異常検知を行う場合，三つの課題が存在する．一つ目に，IoT ゲートウェイがサイバー攻撃を受けやすい点から，異常検知システム自体の安全性を保証

する必要がある点が存在する．二つ目に，コンテナ提供者とホスト OS 提供者が異なる点からホスト OS が信頼できるとは限らないため，信頼できないホスト OS 上においてもコンテナの安全性を保証する必要がある点が存在する．三つ目に，IoT アプリケーションの運用を想定したコンテナの異常検知を行う際の異常検知アルゴリズムについて検討をする必要がある点が存在する．本提案では一つ目と二つ目の課題に対し，ハードウェアによる保護領域での異常検知方式を提案する．また三つ目の課題に対し，機械学習によるシステムコール列の異常検知方式を提案する．

4.2 ハードウェア保護領域での異常検知方式

コンテナの発行するシステムコールを取得し，Intel SGX による保護領域内で異常検知を行うモデルを提案する．Intel SGX は信頼できるリソースをハードウェアとハードウェアにより作成される保護領域内のプログラムに限定している．これにより，サイバー攻撃及び信頼できないホスト OS による異常検知システム自体の改変を防ぐことができるため，信頼できるとは限らないホスト OS 上においても異常検知システムの安全性を保証することができる．

4.3 機械学習によるシステムコール列の異常検知方式

異常検知に用いられるアルゴリズムを複数検討し，異常検知率の評価を行い，IoT アプリケーションコンテナの異常検知アルゴリズムを比較する．検討手法として，隠れマルコフモデルを用いた手法とオートエンコーダを用いた手法のそれぞれについて，BoSC を用いる手法とシーケンスを用いる手法の比較検討を行う．

5. 実装

本章ではハードウェア保護領域での異常検知方式と機械学習によるシステムコール列の異常検知方式の実装について述べる．

5.1 システム構成

システムの構成を図 2 に示す．IoT ゲートウェイにおいて，アプリケーションの機能がコンテナによって提供されるモデルを想定する．また，信頼できるリソースをハードウェア及びハードウェアによって保護された領域とする．信頼できないリソースをホスト OS 及びハードウェアによって保護されていないコンテナとする．提案手法では，コンテナの発行するシステムコールを取得し，ハードウェアにより保護された領域で異常検知を行う．保護領域の作成及びシステムコールの取得，異常検知アルゴリズムの実装について本章で述べる．

5.2 保護領域

保護領域でのコンテナ作成には，SCONE[8] を用いる．

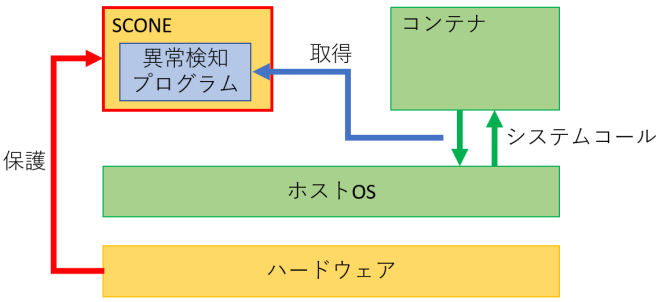

図 2 システム構成
Fig. 2 System Configuration.

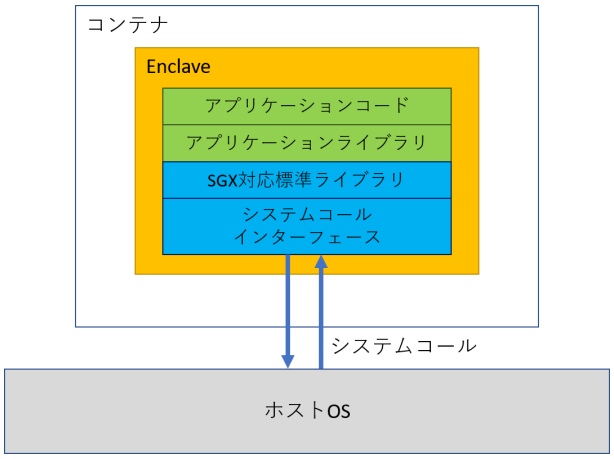

図 3 SCONE 概要
Fig. 3 Overview of SCONE.

本提案では，異常検知システムを SCONE コンテナ内で運用する．

SCONE の概要を図 3 に示す．SCONE はコンテナで運用されるアプリケーションとアプリケーションライブラリを Intel SGX による保護領域である Enclave 内に置くことで，コンテナで運用されるアプリケーションの安全性を向上させるシステムである．SCONE は Enclave 内のアプリケーションとアプリケーションのライブラリに対し，SGX 対応標準ライブラリとシステムコールインターフェースを提供する．通常，Enclave 内でのコード実行は標準ライブラリの使用が制限されており，アプリケーションを Enclave 内で実行するためにはアプリケーションのコードを改変する必要がある．しかし，SCONE により提供される SGX 対応標準ライブラリとシステムコールインターフェースを用いることにより，アプリケーションは Enclave 内においても通常と同じようにコードを実行することができる．

5.3 システムコールの取得

本節ではシステムコールの取得方式と監視対象のシステムコールクラスについて述べる．

5.3.1 システムコールの取得方式

IoT アプリケーションコンテナの発行するシステムコールとその引数を取得するために Sysdig Falco[16] を用いる．

表 1 監視するシステムコールクラス
Table 1 List of System call classes.

システムコール	引数による区別	クラス番号
write	“motion.log” を含む	0
	“netcam_init” を含む	1
	“172.17” を含む	2
	“image/jpeg” を含む	3
	“HTTP” を含む	4
	“html” を含む	5
	“/var/lib/motion” を含む	6
	“detected” を含む	7
	上記以外	8
close	なし	9
munmap	なし	10
clone	なし	11
unlink	なし	12
mmap	なし	13
ioctl	なし	14
open	“/etc/localtime/” を含む	15
	上記以外	16
fstat64	なし	17
accept	なし	18

Sysdig Falco は，コンテナのシステムコールが発生した際，カーネル空間内のドライバによりイベントを取得し，ユーザ領域にコピーし，ユーザ空間でシステムコールのパースを行う．このようにすることで，strace とは異なりコンテナの動作を妨げることなくシステムコールを取得することができる．また，システムコールのパース処理は SCONE コンテナ内でも行うことができることを確認した．Sysdig Falco は，システムコールの取得をカーネル空間内で行うため，ホスト OS の信頼性が問題となるが，この問題の解決は今後の課題である．

本提案では，コンテナの動作を妨げることなくコンテナのシステムコールを取得することができる点，システムコールのパースを SCONE コンテナ内で行うことができる点から，コンテナのシステムコールの取得に Sysdig Falco を用いる．

5.3.2 監視対象のシステムコールクラス

監視対象のシステムコールは IoT アプリケーション向け IDS[11] を参考に 10 種類とした．また，システムコールの引数を考慮して異常検知を行うことでより細かいアプリケーションの挙動まで監視可能 [17] であることから，システムコールの引数による区別を含め，監視対象のシステムコールクラス数を設定した．監視カメラアプリケーションである motion[12] が動作するコンテナのシステムコールクラスを表 1 に示す．

引数による区別の理由として，motion アプリケーションが特定のシステムコールを多く発生させていたことが挙げられる．特に write システムコールの発生頻度は高く，ログファイルへの書き込み，カメラ画像の保存，カメラとの通信，ウェブ UI の操作を行う際に多く発生した．そこで，write システムコールの引数からアプリケーションの挙動のより細かい解析を行うために，表 1 に示す 19 種類のシステムコールクラスの区別方式を定めた．

write システムコールの引数による区別の対象となるシステムコールクラスの特徴について述べる．クラス番号 0 のシステムコールクラスは motion.log ファイルにログを出力する際に発生する．クラス番号 1,2,3 のシステムコールクラスは，コンテナとカメラが通信を行う際に発生する．クラス番号 4,5 のシステムコールクラスはコンテナが Web UI を提供する際に発生する．クラス番号 6 のシステムコールクラスはカメラ画像を/var/lib/motion/に保存する際に発生する．クラス番号 7 のシステムコールクラスは，カメラ画像から動体検知が行われた際に発生する．このように，引数による区別を行うことで write システムコールにより行われる処理まで判別することができるため，アプリケーションの挙動のより細かい解析を行うことができる．

5.4 隠れマルコフモデルを用いた異常検知

隠れマルコフモデルを用いた異常検知の実装について述べる．隠れマルコフモデルは python の hmmlearn ライブラリを使用して実装する．モデルは離散値を扱うクラスである MultinomialHMM を使用し，隠れ状態数は文献 [18] を参考に 3 とした．

オープンソース監視カメラアプリケーション motion をコンテナで運用する場合を想定し，Sysdig Falco により取得されるシステムコール列を表 1 に示すクラス番号に変換したものを用いる．初期学習では，学習系列に対し MultinomialHMM クラスにおいて初期学習を行う fit メソッドを用いてシステムコール番号列から学習モデル M を作成する．運用段階では，観測系列に対し MultinomialHMM クラスにおいて観測系列の尤度を計算する score メソッドを用いて観測系列の尤度を計算する．score メソッドにおいて観測系列の尤度は負の値として与えられるため，正負を反転したものを異常値スコアとして扱う．

観測系列の尤度は学習モデル M から観測系列が得られる確率を対数尤度で表したものであり，異常値スコアが高いほどモデル M から観測系列が得られる確率は低いため，異常であるとする．異常値スコアに対し，しきい値を定め，観測系列の異常値スコアがしきい値を上回った場合，異常であると判定する．

5.4.1 シーケンスを用いる場合

初期学習では，学習データとして正常実行時のシステムコール番号列を学習する．運用段階では，観測されたシステムコール番号列に対し異常値スコアを計算する．一度の異常値スコアの計算に用いるデータのウィンドウサイズはシステムコールクラス数と同じ 19 とした．しきい値につ

いては，正常実行時に観察される異常値スコアから，誤検知率が低くなるように調節する．

5.4.2 BoSC を用いる場合

初期学習では，学習データとして正常実行時のシステムコール番号列を BoSC に変換したものを学習する．システムコール番号列を BoSC に変換する際，データの次元数が 1 から 19 になるが，MultinomialHMM クラスでは多次元シーケンスの学習ができないため，1 次元に変換したものを学習する [18]．運用段階では，観測されたシステムコール番号列を BoSC に変換したものに対し異常値スコアを計算する．一度の異常値スコアの計算に用いるデータのウィンドウサイズはシステムコールクラス数と同じ 19 とした．しきい値については，正常実行時に観察される異常値スコアから，誤検知率が低くなるように調節する．

5.5 オートエンコーダを用いた異常検知

オートエンコーダを用いた異常検知の実装について述べる．オートエンコーダは Keras を用いて実装する．モデルは Model クラスを使用し，エンコーダ，中間層，デコーダの 3 層構造とした．過学習を防ぐため，隠れ層のユニット数は入力次元数に対し十分に多い 160 とし，中間層の正則化パラメータは 10e-7 とした．

motion をコンテナで運用する場合を想定し, Sysdig Falco により取得されるシステムコール列を表 1 に示すクラス番号に変換したものを用いる．初期学習では，学習系列に対し Model クラスにおいて初期学習を行う fit メソッドを用いてシステムコール番号列から学習モデル M を作成する．運用段階では，観測系列 x に対し Model クラスにおいて観測系列のラベルを推定する preict メソッドを用いて推測系列 $\hat{x}$ を計算し，x と $\hat{x}$ の平均二乗誤差を異常値スコアとして扱う．

観測系列 x が初期学習時に発生した値と近い値であった場合，predict メソッドにより得られる推測系列 $\hat{x}$ は観測系列 x と近い値となるため，x と $\hat{x}$ の平均二乗誤差は小さくなる．異常値スコアが大きいほど観測系列 x が初期学習時に発生した値とは遠い値であると判断し，異常であると判定する．異常値スコアに対し，しきい値を定め，観測系列の異常値スコアがしきい値を上回った場合，異常であると判定する．

5.5.1 シーケンスを用いる場合

初期学習では，学習データとして正常実行時のシステムコール番号列を学習する．運用段階では，観測されたシステムコール番号列に対し異常値スコアを計算する．一度の異常値スコアの計算に用いるデータのウィンドウサイズはシステムコールクラス数と同じ 19 とした．しきい値については，正常実行時に観察される異常値スコアから，誤検知率が低くなるように調節する．

表 2 評価環境
Table 2 Evaluation environment.

OS	Ubuntu16.04
CPU	Intel core i5-7500 @ 3.40GHz
Python	Version2.7

5.5.2 BoSC を用いる場合

初期学習では，学習データとして正常実行時のシステムコール番号列を BoSC に変換したものを学習する．運用段階では，観測されたシステムコール番号列を BoSC に変換したものに対し異常値スコアを計算する．一度の異常値スコアの計算に用いるデータのウィンドウサイズはシステムコールクラス数と同じ 19 とした．しきい値については，正常実行時に観察される異常値スコアから，誤検知率が低くなるように調節する．

6. 評価検討

本章では提案手法の評価検討を行う．評価環境について表 2 に示す．隠れマルコフモデルを用いた異常検知手法，オートエンコーダを用いた異常検知手法のそれぞれについて，シーケンスを用いた場合と BoSC を用いた場合について異常検知率の評価を行う．監視対象アプリケーションとしてオープンソース監視カメラアプリケーションの motion が動作するコンテナの異常検知を行う．異常動作を生起させるコンテナの操作として不審コマンドを定義し，監視対象のコンテナにおいて不審コマンドが実行された際の異常値スコアの変化を観察する．

6.1 不審コマンドの定義

IoT アプリケーションコンテナにおける不審コマンドの検知を評価する．正常実行時において，アプリケーションの設定が変えられることは無いと仮定し，アプリケーションの設定の書き換えを行う操作を不審なコマンドとして検知を試みる．不審コマンドとして，以下の一連の操作を定義する．

- エディタによる設定ファイルの書き換え
- ps コマンドによるプロセス番号確認
- kill コマンドによるプロセス停止と再起動

6.2 初期学習

初期学習として，motion アプリケーションの正常動作時のシステムコール列を学習する．学習に用いるデータは motion アプリケーションを用いた IDS の研究 [11] を参考に，15 分の正常実行で観測された 3404 ステップのシステムコール列を学習し，学習モデルを作成する．正常実行として，以下の操作を定義する．

- 操作なし
- ストリーミング

表 3 隠れマルコフモデルを用いた場合の異常値スコア
Table 3 Anomaly Score of Hidden Markov Model approrch.

	正常実行	異常コマンド実行時
シーケンス	13-70	32-126
BoSC	18-30	17-32

- スナップショット取得
- Web UI の操作

6.3 検知率と誤検知率の定義

異常コマンド実行中に観測されるシステムコール列から異常値スコアを計算し，異常値スコアがしきい値を超える割合を百分率で表したものを検知率と定義する．また，正常実行中に観測されるシステムコール列から異常値スコアを計算し，異常値スコアがしきい値を超える割合を百分率で表したものを誤検知率と定義する．

6.4 異常値スコアのしきい値調節

異常値スコアのしきい値を調節するにあたり，各手法において正常実行時と異常コマンド実行時の異常値スコアの取る値の範囲を観察する．観察された異常値スコアより複数のしきい値を検討し，異常検知率及び誤検知率を評価する．

6.5 隠れマルコフモデルを用いた異常検知の評価

隠れマルコフモデルによる異常検知に用いる異常値スコアを表 3 に示す．また，異常値スコアにしきい値を定めた場合の異常検知率を表 4 に示す．隠れマルコフモデルでシーケンスを用いる場合はしきい値を 60 に定めることで検知率が 50%を超え，また誤検知率は 12%となった．また，しきい値を 65 に定めることで検知率 49%を維持しつつ，誤検知率は 0%となった．しかし，BoSC を用いる場合は，しきい値を 35 に定めることで誤検知率を 1%にできたが，異常を検知することができなかった．この理由として，BoSC を用いる場合，ウィンドウ内に特定のシステムコールクラスが多く観測された場合，モデルに対する入力値が”0”となる入力が多くなるためであると考えられる．例えば，ウィンドウサイズが 19 であるシーケンスに対する BoSC を考える．その中のシステムコールクラスが全て 16 であった場合，BoSC は $[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 19, 0, 0]$ となる．hmmlearn で多次元データを学習する場合，１次元に変換して学習するため，上記 BoSC では「0 が 15 回続いた後，19 が入力され，また 0 が 2 回続く」というシーケンスを評価する．初期学習において「BoSC 中で 0 が続く確率」が高い場合，観測系列中で 0 が続いた場合でも異常値スコアは低くなる．このような状態を防ぐ手段として，観測するシステムコールクラスを頻出のシステムコールに限定することが考えられるが，今後の課題である．

表 4 隠れマルコフモデルを用いた場合の異常検知率
Table 4 Detection rate of Hidden Markov Model approrch.

	異常値スコアのしきい値			
シーケンス		55	60	65
	検知率	53%	51%	49%
	誤検知率	15%	12%	0%
	異常値スコアのしきい値			
BoSC		25	30	35
	検知率	13%	0%	0%
	誤検知率	79%	47%	1%

表 5 オートエンコーダを用いた場合の異常値スコア
Table 5 Anomaly score of Auto Encoder approrch.

	正常実行	異常コマンド実行時
シーケンス	34-51	43-60
BoSC	6-9	8-17

表 6 オートエンコーダを用いた場合の異常検知率
Table 6 Anomaly score of Auto Encoder approrch

	異常値スコアのしきい値			
シーケンス		40	45	50
	検知率	95%	79%	11%
	誤検知率	87%	40%	0%
	異常値スコアのしきい値			
BoSC		8	9	10
	検知率	99%	87%	83%
	誤検知率	0%	0%	0%

6.6 オートエンコーダを用いた異常検知の評価

オートエンコーダによる異常検知に用いる異常値スコアを表 5 に示す．異常値スコアにしきい値を定めた場合の異常検知率を表 6 に示す．オートエンコーダでシーケンスを用いる場合はしきい値を下げることで検知率は向上したが，誤検知率が上昇した．BoSC を用いた場合，しきい値を調節することで高い検知率と低い誤検知率を実現することができた．この理由として，シーケンスの推測がウィンドウ中に現れる各システムコールクラスの数とその順番を推測する必要があるのに対し，BoSC の推測はウィンドウ中に現れる各システムコールクラスの数のみの推測であり，その順番を推測する必要が無い分，学習及び入力の再現が容易であることが挙げられる．

6.7 異常検知アルゴリズムの比較

隠れマルコフモデルでシーケンスを用いる場合，しきい値を下げることで検知率と誤検知率が共に上昇したが，検知率を上回ることはなかった．これに対し，隠れマルコフモデルで BoSC を用いる場合，異常検知を行うことができなかった．原因として，監視対象のシステムコールクラス数が多く，BoSC を十分にモデリングすることができなかったことが考えられる．オートエンコーダでシーケンスを用いる場合，しきい値を下げることで検知率を上げることが

できたが，誤検知率が上昇した．これに対し，オートエンコーダで BoSC を用いる場合，高い検知率と低い誤検知率を実現することができた．これらの結果より，コンテナ環境におけるシステムコールに基づく異常検知では，システムコールの正確な順番に影響されない点で BoSC がシーケンスよりも優れていること，システムコールクラス数の多い BoSC を学習することができる点でオートエンコーダが隠れマルコフモデルよりも優れていることが言える．

本稿における評価は motion アプリケーションが動作するコンテナの，表 1 に示すシステムコールクラスを監視した際の評価であり，他のアプリケーションが動作するコンテナの異常検知を行う場合，改めてシステムコールクラスの定義，しきい値の調整を行う必要がある．

7. おわりに

IoT ゲートウェイ上の，信頼できるとは限らないホスト OS 上で動作するコンテナの異常検知を目的に，コンテナのシステムコールを監視し，ハードウェアによる保護領域内で異常検知を行うモデルを提案した．異常検知に用いるアルゴリズムとして隠れマルコフモデルを用いる手法とオートエンコーダを用いる手法を検討し，それぞれシーケンスを用いる場合と BoSC を用いる場合を実装して評価を行った．異常検知アルゴリズムの検討にあたり，オープンソース監視カメラアプリケーションの動作するコンテナの異常検知を行い，それぞれの手法について異常検知率を評価した．今後の課題として，対象アプリケーションを増やす事，システムコール取得プロセスの安全性向上，隠れマルコフモデルを用いる方式における異常検知率の向上が挙げられる．

謝辞

本研究の一部は，JSPS 科研費基盤研究 (C)19K11962 による助成を受けたものです．

参考文献

[1] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, Shiuhpyng Shieh.: IoT Security: Ongoing Challenges and Research Opportunities, 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications (2014).

[2] Bukhary Ikhwan Ismail, Ehsan Mostajeran Goortani, Mohd Bazli Ab Karim, Wong Ming Tat, Sharipah Setapa, Jing Yuan Luke, Ong Hong Hoe.: Evaluation of Docker as Edge Computing Platform, IEEE Conference on Open Systems (2015).

[3] Mostafa Kahla, Mohamed Azab, Ahmed Mansour.: Secure, Resilient, and Self-configuring Fog Architecture for Untrustworthy IoT Environments, 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (2018).

[4] Yan Zhai, Qiang Cao, Jeffrey Chase, Michael Swift: TapCon: Practical Third-Party Attestation for the Cloud, HotCloud'17 Proceedings of the 9th USENIX Conference on Hot Topics in Cloud Computing (2017).

[5] Mahmoud Ammar, Giovanni Russello, Bruno Crispo: Internet of Things: A survey on the security of IoT frameworks, Journal of Information Security and Applications 38 (2018).

[6] Amr S. Abed, Charles Clancy, David S. Levy.: Intrusion Detection System for Applications using Linux Containers, STM 2015. LNCS, vol. 9331, pp. 123-135. Springer, Heidelberg (2015).

[7] Intel, C：INTEL SOFTWARE GUARD EXTENTIONS HOME, https://software.intel.com/en-us/sgx.

[8] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O 'Keeffe, Mark L Stillwell, David Goltzsche, David Eyers, Rdiger Kapitza, Peter Pietzuch, Christof Fetzer.: SCONE: Secure Linux Containers with Intel SGX, 12th USENIX Symposium on Operating Systems Design and Implementation (2016).

[9] Dit-YanYeung, YuxinDing.: Host-based intrusion detection using dynamic and static behavioral models, Edwin Hancock: Pattern Recognition Volume 36, Issue 1 (2003).

[10] Dae-Ki Kang, D.Fuller, V.Honavar.: Learning classifiers for misuse and anomaly detection using a bag of system calls representation, Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (2005).

[11] Man-Ki Yoon, Sibin Mohan, Jaesik Choi, Mihai Christodorescu, Lui Sha.: Learning Execution Contexts from System Call Distributions for Intrusion Detection in Embedded Systems, Proceedings of the Second International Conference on Internet-of-Things Design and Implementation (2017).

[12] Motion, https://motion-project.github.io/.

[13] L.E. Baum, T. Petrie, G. Soules, N. Weiss.: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, Annals of Mathematical Statistics, Annals of Mathematical Statistics 41 (1970).

[14] A.P. Dempster, N.M. Laird, D.B. Rubin.: Maximum likelihood from incomplete data via the EM algorithm (with discussion), J. Roy. Statist. Soc. Ser. B 39 (1977).

[15] Jinwon An, Sungzoon Cho.: Variational Autoencoder based Anomaly Detection using Reconstruction Probability, SNU Data Mining Center Special Lecture on IE (2015).

[16] Sysdig: Sysdig Falco, https://falco.org/.

[17] Gaurav Tandon, Philip Chan.: Learning Rules from System Call Arguments and Sequences for Anomaly Detection, Department of Computer Sciences Technical Report CS-2003-20 Florida Institute of Technology (2003).

[18] 2010-present, hmmlearn developers.: hmmlearn, https://hmmlearn.readthedocs.io/en/latest/tutorial.html.