情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

システムコールの発行履歴が表す情報量の機微に基づく異常検知手法

鑪　講平[†1,†3]　堀　良彰[†2,†3]
竹内　純一[†2,†3]　櫻井　幸一[†2,†3]

近年，計算機に侵入するマルウェアによる被害が多数報告されている．侵入過程ではプロセスの制御が奪われるため，システムコールの発行履歴を監視し，異常動作を検出する研究が行われてきた．本研究では，一定期間に発行されたシステムコールから次に発行されるシステムコールの確率に着目し，マルコフ過程と見なす．システムコールの発行がもたらす情報量と，このマルコフ情報源から得られる情報量の期待値を比べ，この差が大きい場合に侵入行為とする検定手法を導入する．既存手法との比較により，提案手法の有効性を実験的に示す．

An Anomaly Detection of Process Behavior by analyzing Sensitivity on Information Entropy of System call

Kohei Tatara,[†1,†3] Yoshiaki Hori,[†2,†3]
Junichi Takeuchi[†2,†3] and Kouichi Sakurai[†2,†3]

Recently, there are many reports of malwares which intrude and cause damages to computer systems. Many researchers work on monitoring a process and its history of emitting system calls and detecting its anomalous behavior because malwares often take away control from the process in their process of intrusion. In this paper, we focus on conditional probability of a system call came from a set of system calls executed at fixed periods and construct a Markov model by using its conditional probability. Moreover, we compare an information entropy of the system call and an expected value of the entropy from the Markov source. If its difference is higher than a threshold, we can decide anomalous behavior of the process. We experimentally show effectiveness of our proposal by comparing with the existing method.

1. はじめに

今日，計算機の不正利用の事例が数多く報告されている．計算機が乗っ取られると，個人情報の漏洩やさらなる攻撃への踏み台に利用される危険性があるため，侵入を検知する手法の重要性は高い．侵入行為の多くは，バッファオーバフローを引き起こし，プロセスの制御を奪う[1]．そのため，制御の主体が替わる前後で変化する特徴を，効率よく観察することが必要である．

近年，プログラムの制御フローをシステムコールシーケンスとして表現する試みが，異常検知システムの研究で広く行われてきた[2)–6),8)–10)]．異常検知では，訓練期間に監視対象から得たデータを用いて，監視対象が正常に動作しているかどうかを監視する．このため，対象の正常な動作に基づいた特徴抽出が肝要である．利用者に供与される学習データの量が検知精度に影響を与える．また，異常と判断されたイベントと特定の原因とを関連付けることが難しいという問題もある．

システムコールの発行履歴は，オペレーティングシステム側で取得でき，プログラムのソースコードは必要としない．このため，監視対象のプログラムを再コンパイルする必要がなく，利便性が良いという特徴がある．また，システムコールシーケンスに基づいて検知したプロセスの異常は，バッファオーバフロー攻撃による侵入行為の発生と結びつけることができる．

我々は以前，システムコール間の相関性に基づきベイジアンネットワークを形成し，これを用いて異常を検知する手法を提案した[12)]．この手法では，一定期間に発行されたシステムコールから次に発行されるシステムコールの確率を時系列上に並べ，正常時と異常時とにおける平均の差異をノンパラメトリック検定により調べた．本研究では，同じ確率を用いてマルコフモデルとする見直しを行う．さらに，システムコールの発行がもたらす情報量と，このマルコフ情報源から得られる情報量の期待値を比べ，この差が大きい場合に侵入行為と

†1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan
†2 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University, Japan
†3 財団法人九州先端科学技術研究所
Institute of Systems, Information Technologies and Nanotechnologies (ISIT)

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

する検定手法を導入する．このモデルの見直しと検定手法の変更により，既存手法よりも検知精度が向上したことを実験的に明らかにする．

本論文の構成は以下の通りである．2 章で，バッファオーバフロー攻撃による侵入行為と，システムコールの発行履歴に基づく異常検知に関する研究を紹介する．次に，3 章で，システムコール間の相関性に基づいた異常検知手法について述べる．4 章では，提案手法と既存手法との実験的な比較を行い，結果を述べる．5 章では，誤検知が発生する原因を考察する．最後に，6 章でまとめとする．

2. 問題背景と関連研究

2.1 バッファオーバフロー脆弱性を利用する侵入行為

プログラムが実行されると，スタックと呼ばれる逐次入出力データを一時的に貯えるための記憶領域が確保される．スタックは，データの追加と取出しを一方の端だけで行う First In Last Out（FILO）形式のデータ構造を持ち，サブルーチンや関数を呼び出す際に，処理中のデータや戻りアドレスなどを一時的に退避する場合に使うことが多い．これらの値は，関数が呼び出されるとスタックに積まれ，関数から処理が戻るとスタックから破棄される．

一方，C，C++等のプログラミング言語では，ローカルバッファの確保と管理はプログラマに委ねられている．そのため，プログラマは，自らが決めたサイズのローカルバッファが適切に使用されるようにプログラミングを行う責任がある．ここで，予め確保したローカルバッファ（local buffer）のサイズを超えて値を書き込むことを許してしまうと，フレームポインタ（fp）やリターンアドレス（ret）の値が書き換えられてしまう．これをバッファオーバフローと呼ぶ．侵入行為の過程では，このバッファオーバフローを故意に引き起こして，リターンアドレスや関数ポインタの値をシェルコードやライブラリのアドレス（attack code）で置き換える．図 1 に，バッファオーバフローにより改ざんされたスタックの内部状態を示す．攻撃者は，シェルを起動するように制御フローを変更をすることで，任意のコマンドを実行することができる．本論文では，これを侵入行為と呼ぶ．

2.2 システムコールの発行履歴に基づく異常検知に関する研究

Forrest らは，システムコールの発行履歴から N-gram のシーケンスを取り出し，プロセスの動作を検証する指標として扱った[2),3)]．ここで，N-gram とはシステムコール番号を

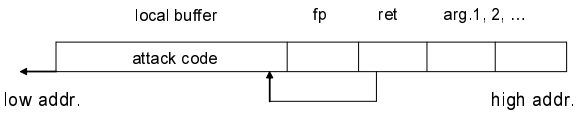


図 1 バッファオーバフロー攻撃が行われた際のスタック内部の状態

文字とおいた長さ N の文字列を表す．監視期間の発行履歴から，学習データに存在しない N-gram の数を調べ，閾値を超えた場合に異常と判定する．この研究は，発行履歴における N 個の連続したシステムコールの局所性を明らかにした．既存研究において，最適な N の値を求めたり[7)]，N の値を可変にする試みが為されている[5)]．さらに，確率的手法を用いることにより，N-gram が失う情報量を抑える取り組みや[6)]，スタック内部の情報を付加して精度を高める研究もある[10)]．これらの研究は，アルゴリズムの特徴が検知精度に与える影響を検証した，

Lee らは，RIPPER と呼ばれるルール学習型プログラムを用いた異常検知の精度を評価した[8)]．“正常” もしくは “異常” とラベル付けされた N-gram を RIPPER の入力として，If-then 型のルールセットを生成する．ルールセットは “if $X_2 = s_{104}$ and $X_7 = s_{112}$ then the sequence is *normal*”(ここで，$X_i = x_j$ は N-gram における i 番目のシステムコール番号が j であることを意味する．) のような形式を取る．Lee らはまた，同時に N-gram における N 番目と $(N+1)/2$ 番目のシステムコールを予測するという試みを行い，異常検知が可能なことを実験的に示した．彼らは，N-gram における個々のシステムコールは特定の位置にある他のシステムコールとの間に相関性を持つことを明らかにした．

Eskin らは，sparse Markov transducers というモデルを用いた検知手法を提案した[6)]．複数の N-gram において重複する部分をワイルドカードに置き換え，枝の数を減らした．そうして得られた sparse Markov tree と，葉の部分に対応する条件付確率により異常検知を行う．結果的に，Forrest らの手法[2),3)] に比べて精度が高いことを示し，確率的な閾値を用いた手法の有効性を実証した．

本論文では，Lee らや Eskin らと同様にシステムコール間の相関性を利用する異常検知手法を提案する．具体的には，N-gram における N 番目のシステムコールの決定に，その前の $(N-1)$-gram が関係する割合を確率として求め，これをマルコフ過程と見なした．シ

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

ステムコールの発行がもたらす情報量と，このマルコフ情報源から得られる情報量の期待値を比べ，この差が大きいときに侵入行為とする検定手法を使って異常検知を行う．一方で，Eskin らの手法は N-gram における N 番目より前の $(N-1)$-gram を逐次的なものとして扱う[6]．例えば，4-gram として {mmap stat write open} と {stat mmap write open} が与えられたとき，Eskin らの手法では，図 2 のような sparse Markov tree が生成される．提案手法では，2 つの 4-gram を同一のものとして扱うため Eskin らの手法に対して必要なメモリ領域を削減できる．

3. システムコールの発行がもたらす報量の機微に基づく異常検知

3.1 訓練期間に得られた学習データを用いたマルコフモデルの構築

はじめに，訓練期間に監視対象のプログラムが発行したシステムコールシーケンスの履歴から N-gram を取り出し，この N-gram を用いてマルコフモデルを構築する．システムコールは，アプリケーションプログラムが OS の提供するサービスを利用するために用意された関数であり，それぞれにユニークな番号が割り振られている．以後，システムコール s_i という表現は，番号 i のシステムコールを表す．システムコールの全体集合は Σ $(s_i \in \Sigma)$ とする．$X_t = s_i$ は，時系列のある時点 t において，プログラムがシステムコール s_i を発行することを表す．訓練期間に得られた学習データからマルコフモデルを構築する手順は以下の通りである．$\pi(X_{t-1}^{t-D})$ は提案モデルにおける状態を表す．

システムコールシーケンス $X(= X_0, X_1, \ldots, X_t, \ldots)$ において，t 番目のシステムコール X_t と，過去 D 回にわたり発行されたシステムコールの集合 $X_{t-1}^{t-D}(= X_{t-1}X_{t-2}\ldots X_{t-D})$ の間に相関があると仮定する．D の値は依存度合を表す．

$$\pi(X_{t-1}^{t-D}) = (T_1, T_2, \ldots, T_{|\Sigma|})$$

$$T_i = \sum_{k=1}^{D} I_{s_i}(X_{t-k})$$

I_{s_i} は指示関数であり，$I_{s_i}(s_j) = \delta_{ij}$ とする．この手続きを，全てのシステムコールシーケンスに対して適用する．

図 3 は ftp クライアントプログラムが発行したシステムコールシーケンスに，上記の手続きを適用してマルコフモデルを構築した例である．図 3 における括弧内の数字はシステムコール番号を示す．

次に，全ての状態について $P(X_t|\pi(X_{t-1}^{t-D}))$ を計算する．条件付確率の値は全て，これら N-gram の出現頻度を基に計算する．Ξ は提案モデルにおける状態の全体集合を表す．$(\xi \in \Xi.)$

$$\Xi = \{\pi(x_1^D)|x_1^D \in \Sigma^D\}$$
$$|\Xi| = {}_{|\Sigma|}H_D = {}_{|\Sigma|+D-1}C_D$$

訓練期間に，イベント $X_{t-1}^{t-D} = x_{t-1}^{t-D} = \xi$ が m 回，$X_t = s$ が n 回，$N(= D+1)$-gram シーケンスが l 回観測されたとき，同時確率 $P(\pi(X_{t-1}^{t-D}) = \xi)$ と条件付確率 $P(X_t = s|\pi(X_{t-1}^{t-D}) = \xi)$ を次のように計算する．

$$P(\pi(X_{t-1}^{t-D}) = \xi) = \frac{m+1}{l+|\Xi|}$$
$$P(X_t|\pi(X_{t-1}^{t-D}) = \xi) = \frac{n+1}{m+|\Sigma|}$$

3.2 監視期間における異常検知の手順

監視期間に，プログラムがシステムコール s_i を発行した場合，条件付確率 $P(\pi(X_t = s_i)|\pi(X_{t-1}^{t-D}))$ を求める．バッファオーバフロー攻撃によりプロセスの制御が奪われると，システムコールの発行履歴が訓練期間に観測されたものと異なる[2),3)]．この時，条件付確率の値は小さい．

プロセスの異常を検知する手順を以下に示す．

(1) システムコールシーケンス $X(= X_0, X_1, \ldots, X_t, \ldots)$ において，$P(\pi(X_t)|\pi(X_{t-1}^{t-D}))$ を求める．

(2) 学習から得られたマルコフ情報源のエントロピーの期待値 A を求める．(条件付) エントロピーの定義を以下に示す．

$$\begin{aligned} H(X_t) &= E[-\log P(X_t)] \\ &= -\sum_{s\in\Sigma} P(X=s)\log P(X=s) \end{aligned}$$

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

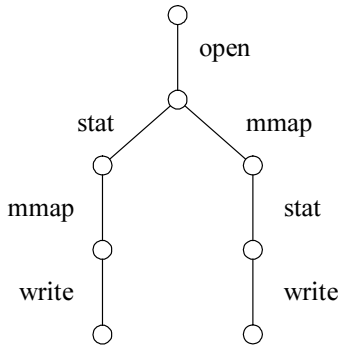
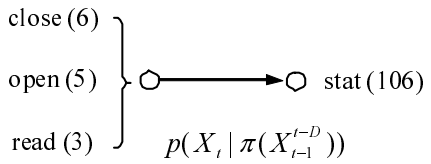


図 2 sparse Markov tree の例[6] (4-gram の場合)

next state	current state
stat	{ close, open, read }
mmap	{ open, read, stat }
close	{ read, stat, mmap }
mprotect	{ stat, mmap, close }
munmap	{ mmap, close, mprotect }
socketcall	{ close, mprotect, munmap }
gettimeofday	{ mprotect, munmap, socketcall }



$$\pi(X_{t-1}^{t-D}) = \{\text{close, open, read}\},$$

$$X_t = \text{stat}$$

図 3 依存度 $D = 3$ の場合におけるマルコフモデルの例（括弧内の数字はシステムコール番号）

$$H(X_t|\pi(X_{t-1}^{t-D})) = E[-\log P(X_t|\pi(X_{t-1}^{t-D}))]$$

ここで，A を定義する．

$$A = -\lim_{n\to\infty} \frac{1}{n+1} \log P(X_0 X_1 ... X_n)$$

定常マルコフ状態を仮定すると，A は以下のように近似できる．

$$\begin{aligned} A &= E[-\log P(X_t|\pi(X_{t-1}^{t-D}))] \\ &= -\sum_{X_t} \sum_{\pi(X_{t-1}^{t-D})} \{P(X_t|\pi(X_{t-1}^{t-D})) \\ &\quad \times P(\pi(X_{t-1}^{t-D})) \log P(X_t|\pi(X_{t-1}^{t-D}))\} \end{aligned}$$

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

表 1 実験に用いたデータセット
Table 1 Details of data sets for experiment

Program	# of seq.	# of seq. for training	# of seq. for testing	# of proc.
ftp	181,661	33,440	146,863 (1,358)	5
ps	11,047	4,112	4,477 (2,458)	11
login	18,586	6,128	7,611 (4,847)	13
sendmail	151,395	73,491	69,618 (8,286)	34
named	115,724	33,491	80,443 (1,790)	4

次に，過去 D 回分の条件付エントロピーの平均を計算する．

$$B_t = \frac{\sum_{k=1}^{D} -\log P(X_{t-k+1}|\pi(X_{t-k}^{t-k+1-D}))}{D}$$

(3) $|A - B_t| > T$ (T: 閾値) であるとき，このプロセスは異常であると判定する．

4. 提案手法による異常検知の実験

3 章の手順を用いて実験を行った結果を述べる．

4.1 実験方法とデータの説明

Forrest らは，プログラムの正常な動作時と侵入行為が発生した際のシステムコールシーケンスが異なることを実験的に検証した[2),3)]．さらに，N-gram を用いて異常検知を行う手法を幾つか選び，検知精度の比較を行った[4)]．この研究において実験に使用されたデータは，Web 上に公開されている[⋆1]．既存研究との比較のために，我々はこのデータを用いて実験を行った．

公開されているデータは，5 つのプログラム（ftp, ps, login, sendmail, named）の正常動作時と異常動作時におけるシステムコールの発行履歴である．プログラムを普段通りに利用した際に記録された “live data” と，様々なオプションを選択して利用した際に得られた “synthetic data” からなる．実験では，これらのデータからシステムコールシーケンスを無作為に選び，それぞれ訓練と監視に割り当てた．表 1 は実験に用いたデータの詳細を示す．表中の値は公開データから得られた N-gram の数を表す．ここで，$N = 6$（$D = 5$）である[7)]．学習データは，バッファオーバフローの発生時におけるシステムコールの発行履歴を含まない．テストに用いたデータのうち，括弧外の数字は，正常動作時の発行履歴から得られた N-gram の数を，括弧内の数字は異常動作時に得られた N-gram の数を表す．また，表にはこれらのシステムコールを発行したプロセスの数を示した．

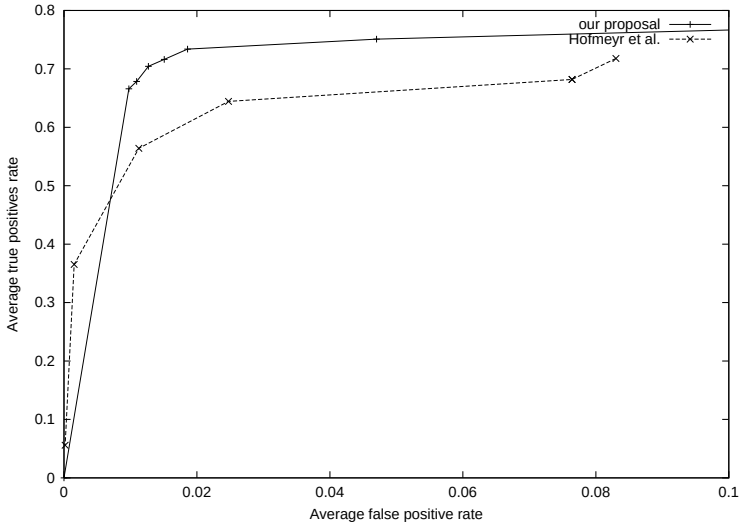


図 4 ftp に関するデータを用いた異常検知の結果（ROC 曲線）（$N = 6$, $D = 5$）
Fig. 4 The experimental result (ROC curve) using ftp data set($N = 6$, $D = 5$)

N-gram に基づく異常検知手法として．tide, stide[4)], や Hofmyer らの手法[3)] があり，高い検知精度を示した．本研究では，検知精度の比較対象として Hofmyer らの手法[3)] を選択した．Hofmyer らの手法では，N-gram 同士のハミング距離を計算して，その値が閾値を上回っていた場合にプロセスの動作を異常と判定する．

4.2 実 験 結 果

異常検知の精度を比較するために，提案手法と Hofmyer らの手法とで ROC 曲線を描いた．ROC 曲線の縦軸を True positive の割合，横軸を False positive の割合とする．ここで，True positive とは，侵入行為が行われる際の発行履歴から得られた N-gram が，異常と判断されることを表す．また，False positive は，正常動作時の発行履歴から得られた

⋆1 http://www.cs.unm.edu/~immsec/data-sets.htm

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

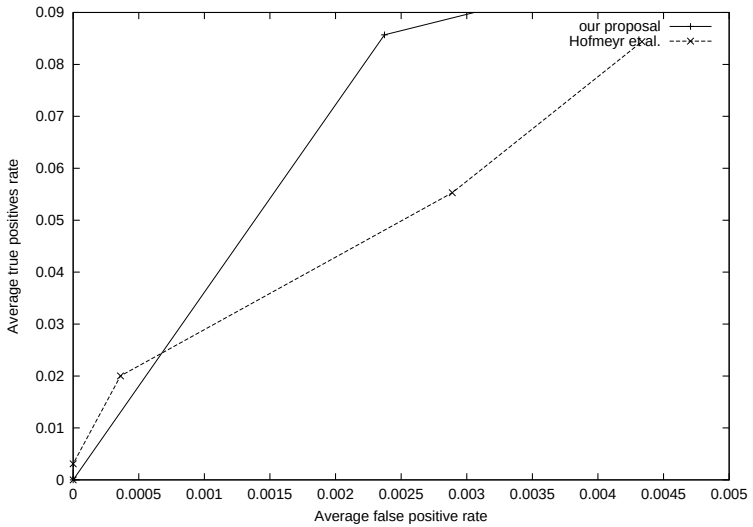


図 5 login に関するデータを用いた異常検知の結果（ROC 曲線）（$N = 6$, $D = 5$）
Fig. 5 The experimental result (ROC curve) using login data set($N = 6$, $D = 5$)

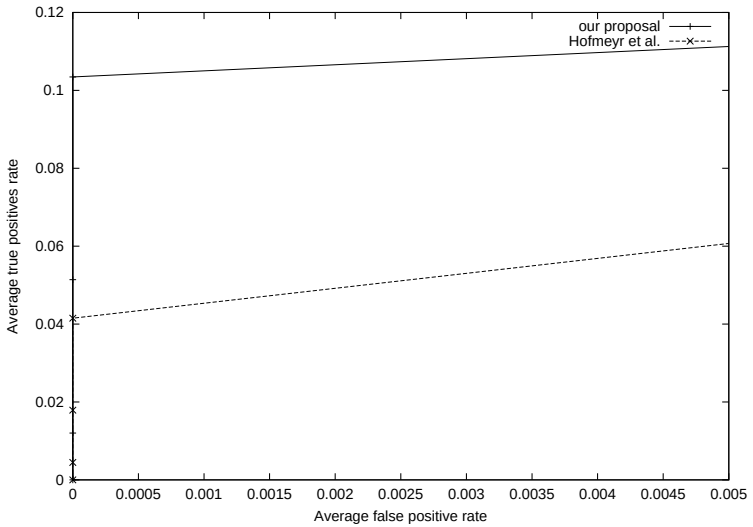


図 6 ps に関するデータを用いた異常検知の結果（ROC 曲線）（$N = 6$, $D = 5$）
Fig. 6 The experimental result (ROC curve) using ps data set($N = 6$, $D = 5$)

N-gram が，誤って異常と判断されることを表す．$N = 6$ として[7]，提案手法と Hofmyer らの手法の閾値を変化させて ROC 曲線を描いた．また，これらの割合は，“live data” と “synthetic data” とを一つに集めたものからのデータセットの無作為抽出を 10 回繰り返して実験を行い，値の平均をとったものを採用した．その結果，図 4, 5, 6, 7, 8 のようなグラフが得られた．提案手法は，全てのプログラムにおいて Hofmyer らの手法より精度の良い検知が可能である．

図 4，8 からわかるように，ftp と named に関しては，プロセスの異常な動作を高い確率で検知できる．一方で，login, ps の True positive の割合は小さい．これらのプログラムが正常動作時に発行する N-gram の種類が多く，異常動作時のデータを入力しても異常と判定し難くなってしまったと思われる．ネットワークを介した攻撃を受けやすいサーバプログラム（named, sendmail）において検知精度に改善が見られ，精度の高い検知が達成できた．

提案モデルでは，ある状態 $\pi(X_{t-1}^{t-D})$ からシステムコール X_t が発行される確率に着目した．$\pi(X_{t-1}^{t-D})$ 内のシステムコールの順序は考慮しない．Forrest, Hofmyer, Eskin らの手法では，最大で $|\Sigma|^N$ 通りの N-gram を記憶する必要がある．提案手法の場合，$N \cdot \frac{(|\Sigma|+N-2)!}{(N-1)!\cdot(|\Sigma|-1)!}$ 通りであり，これは $|\Sigma|^N$ より小さい．このため，N-gram を保存するためのメモリ領域を削減することができる．実験では，Hofmyer らの手法を上回る検知精度を達成したが，システムコールシーケンスを偽装して，検知を回避する攻撃も報告されているため，さらなる評価が必要である[11]．

5. 誤検知に関する考察と本研究の位置づけ

訓練期間において得られるデータが，プログラムの制御フローを全て辿った上で得られる完全なデータであるという保証はない．システムコールの発行履歴に基づく異常検知では，分岐やループ処理に応じた処理を行わない．このため，特定のシステムコールの発行と，プログラムの制御フローを関連付ける作業は非決定的なものとなり，誤検知を完全に無くすこ

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

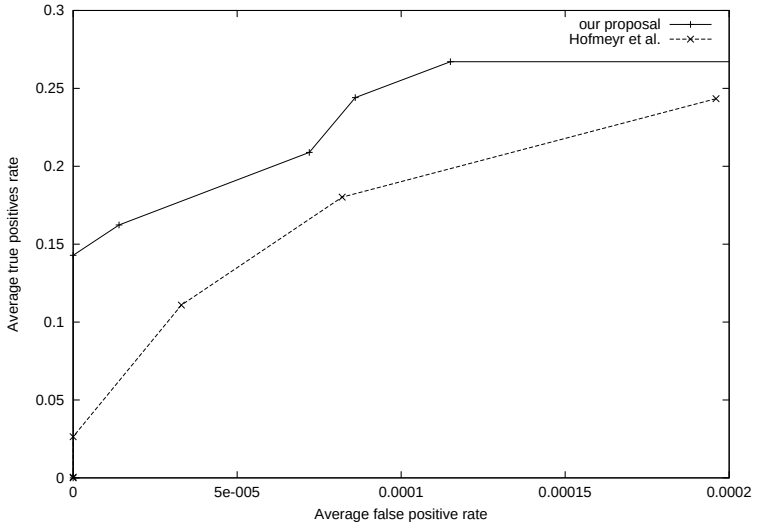


図 7 sendmail に関するデータを用いた異常検知の結果（ROC 曲線）（$N = 6$, $D = 5$）
Fig. 7 The experimental result (ROC curve) using sendmail data set($N = 6$, $D = 5$)

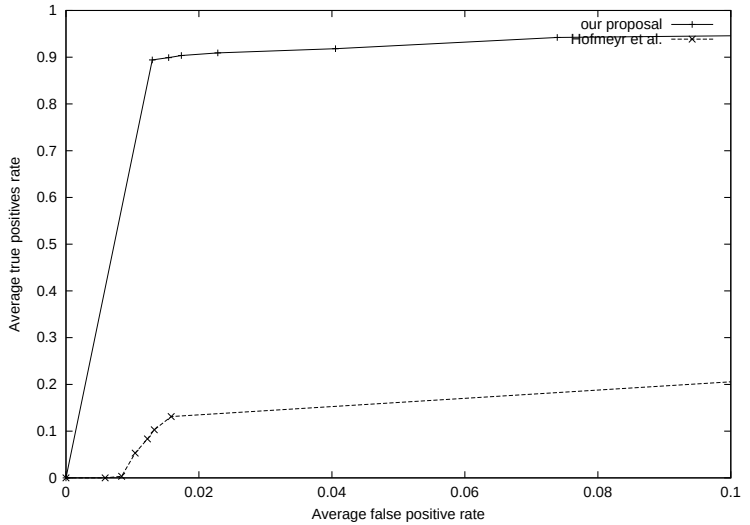


図 8 named に関するデータを用いた異常検知の結果（ROC 曲線）（$N = 6$, $D = 5$）
Fig. 8 The experimental result (ROC curve) using named data set($N = 6$, $D = 5$)

とが難しい．この問題に対して，訓練期間に得られるデータに新たな情報を付加することで，誤検知が発生しない異常検知を行うことができる．追加する情報は，侵入行為が行われた前後で変化し，何時でも検証可能であり，個々の情報はプログラムの制御フローの特定箇所と関連付けられるものでなければならない．

Wagner らは，ソースコードを静的解析することにより，プログラムの制御フローを表現する非決定プッシュダウンオートマトンを生成した[9]．岡らは，Wagner らの方式に，スタックの情報を加えた状態遷移図を生成して，誤検知を無くした[10]．他にも，上記の性質を満足する最も簡単な手法としては，プログラム中に記述されるシステムコールをそれぞれユニークな形式に変換し，予め記録しておくことが考えられる．プログラムの他の箇所で発行される同じシステムコールや，訓練データに含まれないシステムコールと区別するために，ユニークな番号を割り振ったり，引数を与えたりする．この手法により，変換したシステムコールはプログラムコードの特定箇所と関連付けることができるため，非決定性は表れない．これらの研究では，ソースコードを静的解析することによってシステムコールの制御フローを決定的に把握したが[10]，必ずしもソースコードが手に入るとは限らないため，本研究を含め，システムコールの発行履歴のみを用いて誤検知を減らす取り組みも重要である．

6. 結　　論

本論文では，プロセスの挙動を監視し，異常を検知する手法を提案した．これまで，プログラムの制御フローをシステムコールシーケンスとして表現する研究が盛んに行われてきた．それに加え，確率モデルの導入や，スタック内部に保存されている値を付加して精度を高める手法が提案されている．誤検知を減らすため，学習データが不足してもプログラムの制御フローを表現しなければならない．提案手法では，一定の期間に発行されたシステムコールの組み合わせから，次に発行されるシステムコールの確率に着目し，これをマルコフ過程と見なした．従来手法では，一定の期間に発行されたシステムコールの順序を記憶した

Vol.2009-SE-164 No.28
Vol.2009-EMB-13 No.28
Vol.2009-CSEC-45 No.28
2009/5/29

上で，マルコフ過程の状態と見なしたため，提案手法の方が状態数が少ないという利点がある．さらに，システムコールの発行がもたらす情報量と，このマルコフ情報源から得られる情報量の期待値を比べ，この差が大きいときに侵入行為とする検定手法を新たに導入した．実験により既存手法を上回る結果が得られ，提案手法の有効性を明らかにした．

異常検知が広く受け入れられるためには，誤検知を完全に無くし，利便性を高めなければならない．提案手法を基に，より実用的な異常検知システムを設計することが今後の課題である．

謝　　辞

本研究を行うにあたって，第一，第二，および第三筆者は情報通信研究機構（NICT）委託研究「インシデント分析の広域化・高速化技術に関する研究開発」の支援を受けた．

参 考 文 献

1) Beyond-Security's SecuriTeam.com, "Writing Buffer Overflow Exploits - a Tutorial for Beginners", http://www.securiteam.com/securityreviews/5OP0B006UQ.html (accessed 2009-03-10).
2) S. Forrest, S. Hofmeyr, A. Somayaji, T. Longstaf, "A sense of self for Unix processes", Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, pp. 120–128, 1996.
3) S. Forrest, S. Hofmeyr, and A. Somayaji, "Intrusion detection using sequences of system calls", Journal of Computer Security, Vol.6, pp. 151–180, 1998.
4) C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions using System Calls: Alternative Data Models", Proceedings of the IEEE Symposium on Security and Privacy, pp. 133–145, 1999.
5) C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams", Proceedings of the 2000 Workshop on New Security Paradigms, pp. 101–110, 2000.
6) E. Eskin, W. Lee, and S. Stolfo, "Modeling System Calls for Intrusion Detection with Dynamic Window Sizes", Proceedings of the 2001 DARPA Information Survivability Conference & Exposition, pp. 165–175, Jun. 2001.
7) K. M. C. Tan, R. Maxion, ""Why 6?" Defining the Operational Limits of stide, an Anomaly-Based Intrusion Detector", Proceedings of IEEE Symposium on Security & Privacy, pp. 188–201, 2002.
8) W. Lee, S. Stolfo, and P. Chan, "Learning Patterns from Unix Process Execution Traces for Intrusion Detection", Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management, pp. 50–56, 1997.
9) D. Wagner, D. Dean, "Intrusion Detection via Static Analysis", Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp. 156–169, May 2001.
10) M. Oka, H. Abe, Y. Oyama, K. Kato, "Intrusion Detection System Based on Static Analysis and Dynamic Detection", Proceedings of Forum on Information Technology (FIT 2003), Sep. 2003.
11) D. Wagner, P. Soto, "Mimicry Attacks on HostBased Intrusion Detection Systems", Proceedings of 9th ACM Conference on Computer and Communications Security, Nov. 2002.
12) K. Tatara, T. Tabata, K. Sakurai, "The Design and Evaluation of Anomaly Detection System Based on System Call", IPSJ Journal, Special Issue on Research on Computer Security Characterized in the Context of Social Responsibilities, Vol.46, No.8, pp.1967–1975, Aug. 2005.