確率モデルを用いた異常検知手法に関する一考察

鑪　講平† 　　田端　利宏‡ 　　櫻井　幸一‡

†九州大学大学院システム情報科学府　情報工学専攻
〒812-8581 福岡市東区箱崎 6 丁目 10 番 1 号

tatara@itslab.csce.kyushu-u.ac.jp
‡九州大学大学院システム情報科学研究院

{tabata, sakurai}@csce.kyushu-u.ac.jp

あらまし　バッファオーバフローなどを利用した攻撃を防ぐため，計算機上でプログラムが正常に動作していることを，システムコールの発行履歴から検査する研究が盛んに行われている．以前，著者らは，ベイジアンネットワークを用いて，システムコールの履歴をモデル化する確率論的手法を提案した．本論文では，著者らの手法に基づき，その精度や効率を改善するための手法を提案する．また，提案手法の有効性を見るために，実験を行った結果についても述べる．

A Study on Probabilistic Method for Anomaly Detection

Kohei Tatara†　　Toshihiro Tabata‡　　Kouichi Sakurai‡

†Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581 Japan

tatara@itslab.csce.kyushu-u.ac.jp
‡Faculty of Information Science and Electrical Engineering, Kyushu University, Japan

{tabata, sakurai}@csce.kyushu-u.ac.jp

Abstract　In order to prevent an attack, such as a buffer overflow, there are many researches of checking an operation of a program for normal or abnormal based on a history of system calls emitted by it. Previously, the authors proposed a method of modeling the history of system calls in a Bayesian Network. In this paper, we propose a method of improving accuracy and efficiency of anomaly detection based on authors' method. Then, we also describe a result of some experiments to show validity of the method.

1　はじめに

インターネットの急速な普及と共に，ネットワークに接続される計算機は増加の一途を辿っている．それに伴い，悪意を持ったユーザによる計算機への不正侵入の事例が多数報告されており，そうした侵入行為を監視する侵入検知システムの重要性が高まっている．

侵入行為の多くは，バッファオーバフローと呼ばれるプログラムに内在する脆弱性を利用したものである．攻撃者は，プログラム実行中のスタックにおける内部バッファをオーバフローさせることにより，関数のリターンアドレスを書き換える．そうすることにより，攻撃者はプログラムの制御を乗っ取り，任意のコードを実行することが可能となる [1].

一方で，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把握し，それを監視することによって，バッファオーバフローを利用した侵入行為を検知しようとする異常検知の研究が盛んに行われている [2, 3, 4, 5, 6, 7, 8]．異常検知では，正常な動作を示す特徴が観測されなければ異常と判断するため，未知の侵入行為を検知することが可能であるという利点がある．しかし，異常と判断されたことと特定の原因とを関連付けることが難しく，また，システムが比較的複雑になるという欠点がオーバヘッドの増大を招いている．したがって，正常時の動作を示すデータとして何を選択するかということや，そのデータからプログラムの特徴をどのように抽出するかということが重要になる．

Forrest ら [2, 3] の研究に始まるシステムコール・シーケンスに基づいた異常検知では，正常時のイベントを N-gram として扱った．ここで，N-gram とは，システムコールに割り振られたユニークな番号を文字とおいた，長さ N の文字列を示す．N-gram に基づく既存の研究では，精度の面から最適な N の値を求めたり [9]，N の値を可変に決定したりする試みが行われている [5]．また，確率的・統計的手法を用いることにより，N-gram が失う情報量を可能な限り抑えようとする研究も行われている [6]．また，N-gram に基づく手法に，スタックの情報やシステムコールの引数の情報などを加えて，より精度の高い異常検知を目指す研究もある [11]．

以前，著者らは N-gram のうちの N 番目のシステムコールは，それ以前の $N-1$ 個のシステムコールの並びに必ずしも依存しないことを示し，その性質を利用して異常検知を行うことが可能であることを論じた [14]．著者らの手法では，N 番目のシステムコールについて，それ以前の $N-1$ 個の文字列が与えられたときの条件付確率を評価している．確率値は，N 個目のシステムコールが発行される頻度に基づいている．従って，システムコールシーケンスの履歴から，確率値に基づいて正確なモデル化を行うことができれば，より精度の高い異常検知を実現することができると考えられる．本論文では，著者らの手法を基に，異常検知の精度と効率を改善するための手法を提案する．また，提案手法の有効性を見るために，実験を行った結果についても述べる．

2 確率モデルを用いた異常検知手法

2.1 著者らの手法

システムコールは，アプリケーションプログラムがオペレーティングシステムの提供する機能を利用するための関数で，それぞれ名前に対応する番号が割り振られている．以後，システムコール S_i は，システムコール番号が i であるシステムコールを表し，X_i は，システムコールシーケンス X の i 番目のシステムコールを表すとする．

ベイジアンネットワーク [12] は，確率変数間の定性的な依存関係を非循環有向グラフで表したもので，不確実性を含んだ事象や因果関係を表すのに適している．X_i と X_j の間の関係を，ベイジアンネットワークでは有向リンクで $X_i \rightarrow X_j$ と表す．X_i は親ノード，X_j は子ノードと呼ばれ，X_i と X_j の定量的因果関係は条件付確率 $P(X_j|X_i)$ で表される．また，親ノードが複数あるとき，子ノード X_j の親ノードの集合を $\pi(X_j)$ と表す．子ノード X_j について，親ノードの全ての値を条件とする条件付確率 $P(X_j|\pi(X_j))$ を求めたものは条件付確率表（CPT）と呼ばれる．著者らの手法 [14] では，これを正常に動作していることを示すデータとして用いて，プログラムが発行するシステムコールシーケンスから得られる N-gram の正当性を検証する．

プログラムがシステムコール X_i を発行した場合，訓練期間に得られた CPT を用いて条件付確率 $P(X_j|\pi(X_j))$ を求め，一方で，$P(S_i|\pi(X_j))$ が最大になるような i を選ぶ．これを，N-gram のそれぞれの文字に対して行う．最後に，得られた 2 つの条件付確率の集合に対して，マン・ホイットニーの U 検定と呼ばれる統計的手法を適用して異常かどうかを判断する．マン・ホイットニーの U 検定は，2 群の代表値に差があるかどうかを検定する上で有用な手法である [13]．

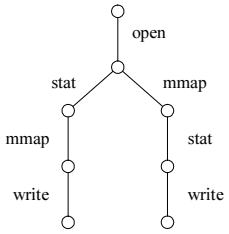

図 1: sparse Markov tree の例 (4-gram の場合)

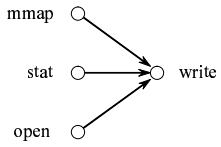

図 2: ベイジアンネットワークの例 (4-gram の場合)

2.2 関連研究

Eskin らは，sparse prediction tree に基づいた sparse Markov transducers を用いたモデルを提案した [6]．彼らの手法では，N-gram における重複する部分をワイルドカードに置き換え，枝の数を減らしている．そうして得られた sparse prediction 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．Eskin らは，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証明した．

一方で，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ている [10]．彼らの手法では，“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 $p_2 = 104$ and $p_7 = 112$ then the sequence is *normal*” (ここで， $p_i = j$ は N-gram における i 番目の文字が 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っている．実験では，University of New Mexico のデータセット [4] を用いて，異常検知が可能なことを示した．

これらの手法は，システムコール間の相関関係に基づいて異常検知を行っている．著者らは以前，この相関関係について，さらに詳細な解析と実験を行った [14]．著者らが提案した手法と，Eskin らの手法との違いは，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かどうかである．例えば，4-gram として，{open mmap stat write} と {open stat mmap write} というものが与えられたとき，Eskin らの手法では，図 1 のような sparse Markov tree が生成される．一方で，著者らの手法では，2 つの 4-gram を同一のものとして扱う（図 2）．Eskin らの手法に対して情報量が減少している．実験では，同様に逐次的な N-gram を扱う Hofmeyr らの手法 [3] に比べて，True positive や False positive の数にわずかな差異が見られる程度で，異常検知を行うには十分であった．ここで，True positive とは侵入行為が起こった際に得られる N-gram の異常が検知されたことを表し，False positive とは，侵入行為が行われなかった場合において，異常と判定されたことを表している．

確率論に基づく手法が，他の N-gram に基づく手法と異なる点は，N-gram や状態の決定に確率値を用いるという点である．プログラムが発行したシステムコールシーケンスから，N-gram として特徴を抽出する場合，システムコールの “順序” と “種類” という要素が用いられる．著者らの手法 [14] では，この “種類” という要素のみに焦点を絞って評価を行った．一方で，確率論に基づく手法は，これにシステムコールが発行される “頻度” という要素を加えることによってより精度の高い異常検知を行うものである．侵入検知としての異常検知システムを実現するためには，さらに精度と効率を改善する必要がある．そのため，我々は確率論に基づいて，それらを改善する手法を 2 つ提案する．

3 提案手法

3.1 適応的異常検知

図 1 に対応するような確率モデルを考える．図では，open の後には stat もしくは mmap が続く．一方，{open, stat} が発行された場合，mmap のみの可能性が残る．著者らの手法や，

表 1: 実験で用いたデータセット

Program	# of seq. for training	# of seq. for testing	# of proc.
ftp	8,360	173,301(1,358)	13(5)
xlock	40,928	87,201(939)	44(2)
ps	1,879	6,711(2,458)	30(11)
login	2,112	11,629(4,847)	36(13)
sendmail	73,491	77,904(8,286)	53(34)

Eskin らの手法 [6] では，上記のような可能性を条件付確率の値を用いて定量的に扱っている．条件付確率の値は，訓練に用いた N-gram における，該当のシステムコールが発行された頻度により決定する．すなわち，条件付確率値は前者において大きく，後者においては小さい．

ここで，1)open の後に write が発行される場合と，2){open, stat} の後に open が発行される場合を考える．図 1 の確率モデルにおいて，1)，2) における条件付確率はともに小さな値をとる．しかし，1) と 2) では，次に発行されるシステムコールに対応する条件付確率が異なるため，異常と判断される可能性もまた異なる．そこで，異常検知システムの状態を下記のように 2 つに分類して，個々に適切な異常検知を行うことを考える．

sensitive 異常と判断する基準を下げた状態のことを指す．異常な状態を検知する数は上がる一方で，正常な状態を誤って異常と判断するため，誤検知が発生しやすい．

insensitive 異常と判断する基準を上げた状態のことを指す．誤検知の数は小さくなるものの，異常な状態，すなわち侵入行為を見逃す恐れがある．

提案手法では，これら 2 つの状態を条件付確率の値によって切り替えることにより，異常検知の精度を高める．

3.2 効率的異常検知

異常検知システムでは，正常な動作を示すデータから効率的にプログラムの特徴を抽出できることが望ましい．特にプログラムが発行するシステムコールシーケンスには，ループ処理や分岐構造に起因する冗長なデータが多数含まれている．

一方で，条件分岐が存在しない場合，逐次的に発行されるシステムコールが決定する．そのため，異常検知の効率を高めるためには，このようなシステムコールシーケンスにおいては，システムコール毎に異常検知を行わないほうがよい．これは，図 2 のような場合，これは {stat, mmap, write} を 1 つのシステムコールとして扱うことを意味する．そこで，我々は条件付確率が 1 かそれにに近い値を取る場合には，異常検知を行わないという制限を設ける．

4 実験

4.1 準備

提案手法を著者らの手法 [14] に適用したことを評価するために実験を行った．データセットは，University of New Mexico の Web サイトに公開されているもの[1]を利用した [4, 6].

データセットは，プログラムの正しい利用によって得られた “live data” と，プログラムオプションを注意深く選ぶことによって得られた “synthetic data” からなる．本実験では，これらのデータを 1 つにしたものから適当な数のデータを選び，訓練とテストに用いた．データセットの詳細を表 1 に示す．表において，第 2 列目は異常検知システムの訓練に，第 3 列目は異常検知のテストに用いた N-gram の数を示す．また，第 4 列目には，実験データにおけるプロセスの総数が記述されている．括弧内の数字は，侵入行為が行われた際に得られたデータに含まれる，N-gram およびプロセスの数を表している．

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

表 2: 実験結果（適応的異常検知）

Program	Our previous method [14]		Exp. 1		Exp. 2	
	True positive	False positive	True positive	False positive	True positive	False positive
ftp	0.02143	0.00062	0.06800	0.00066	0.07317	0.00147
xlock	0.10064	0.00000	0.10064	0.00000	0.15203	0.00000
ps	0.00000	0.00000	0.00000	0.00000	0.00053	0.00000
login	0.00621	0.00000	0.00766	0.00000	0.00745	0.00059
sendmail	0.05168	0.00000	0.05712	0.00000	0.06980	0.00000

表 3: 実験結果（効率的異常検知）

Program	Our method [14]		Exp. 3	
	# of exec.	True positive	# of exec.	True positive
ftp	166,718	0.02143	92,154	0.02564
xlock	40,918	0.10064	40,227	0.13333
ps	6,112	0.00000	3,310	0.00000
login	11,594	0.00621	6,949	0.00939
sendmail	77,904	0.05168	48,326	0.07121

これらのデータセットを用いて，それぞれシステムコールシーケンス毎の True positive や，False positive の割合を計測した．また，本実験におけるパラメータとして，$N = 6$ という値で実験を行い，$CriticalValue = 9$ を設定した [9, 14].

4.2 実験結果

表 2 は適応的異常検知を行った実験結果を示している．表における Exp. 1，Exp. 2 はそれぞれ次のような条件のもとに実験を行った結果である．

Exp. 1 条件付確率が 0.5 以上の値を取る状態を sensitive，0.5 未満の値を取る状態を insensitive と置く．

Exp. 2 条件付確率が 0.5 未満の値を取る状態を sensitive，0.5 以上の値を取る状態を insensitive と置く

sensitive な状態では異常検知の基準（すなわち閾値）を 30%上げ，insensitive な状態では異常検知の基準を 30%下げて計測を行った．表 2 より，True positive の増加が顕著に見られるが，False positive の増加は，無視できるほどに小さい．提案手法を用いることで異常検知の精度が向上していることがわかる．また，Exp. 1 と Exp. 2 では Exp. 2 の方が，True positive と False positive の増加の割合がより大きかった．

表 3 は，異常検知の回数を減らして，効率的異常検知を行った結果を示している．表における Exp. 3 は，次のような条件のもとで実験を行った．

Exp. 3 条件付確率が 0.9 以上である場合には，異常かどうかの判断のための処理を行わない

表における第 3 列目と第 5 列目は，異常検知を行った回数である．提案手法の適用前後において，異常な振る舞いを見せるプロセスは全て検知可能であった．表より，異常検知を行う回数が減少したことによって True positive の割合が増加していることがわかる．

5 考察

異常検知に使用されるデータはしばしば膨大になるため，適切なデータのみを効率的に選り分ける作業が必要となる．冗長なデータを減らし，正常な動作を表すモデルを生成するため，さまざまな手法が提案されている [4, 5, 6, 7, 8].

本論文において，異常検知の精度と効率が改善されたことの背景には，著者らの提案した手

法 [14] が，システムコール間の相関関係の強さを確率値で表すという事実がある．すなわち，同義的に確率論を用いる手法であれば，提案手法は適用可能であることを意味する．

システムコールシーケンスの履歴から，有限状態マシンを生成する手法 [7, 8] と，提案手法が異なる点は，状態の決定を確率値に基づいて行ったという点である．それぞれの手法において，精度の優劣を論じるには，さらなる評価が必要である．

6 まとめ

本論文では，著者らが提案した異常検知手法 [14] を基に，確率モデルを用いた異常検知手法における精度や効率を改善するための手法を提案した．

実験では，提案手法が True positive の割合を高める様子や，異常検知を行う回数を削減することが可能であることがわかった．この結果により，システムコールシーケンスに基づく異常検知の可能性を模索することができた．今後の課題としては，本論文の結果を基にして，効率的かつ実用的な異常検知手法の考案などがある．

謝辞

本研究の一部は，財団法人 情報科学国際交流財団 SSR 産学戦略的研究フォーラム 海外連携型調査研究「オペレーティングシステムのセキュリティ機能に関する調査研究」の支援を受けている．

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 2003-09-05).

[2] S. Forrest, S. A. Hofmeyr, A. Somayaji, T.A. Longstaff. A sense of self for Unix processes. In the *1996 IEEE Symposium on Computer Security and Privacy.*

[3] S. Forrest, S. A. 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. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy,* 1999.

[5] C. Marceau. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the New Security Paradigms Workshop 2000*, 2000.

[6] E. Eskin, W. Lee, S. J. Stolfo. Modeling System Calls for Intrusion Detection with Dynamic Window Sizes, In *Proceedings of DISCEX II. June 2001.*

[7] A. P. Kosoresow, S. A. Hofmeyr, Intrusion Detection via System Call Traces, *IEEE Software, vol. 14, pp. 24–42, 1997.*

[8] R. Sekar, M. Bendre, P. Bollineni and D. Dhurjati, A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors, In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001.

[9] K. M. C. Tan, R. A. Maxion, "Why 6?" Defining the Operational Limits of stide, an Anomaly-Based Intrusion Detector. In *Proceedings of IEEE Symposium on Security & Privacy*, pp. 188–201, 2002.

[10] W. Lee, S. Stolfo, and P. Chan, Learning Patterns from Unix Process Execution Traces for Intrusion Detection, In *Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management*, 50-56, 1997.

[11] M. Oka, H. Abe, Y. Oyama, K. Kato, Intrusion Detection System Based on Static Analysis and Dynamic Detection, In *Proceedings of Forum on Information Technology (FIT 2003)*, Japan, Sep. 2003.

[12] Y. Motomura, I. Hara, User Model Construction System using Probabilistic Networks. http:// staff.aist.go.jp/ y.motomura/ipa/ (accessed 2003-09-05).

[13] W. J. Conover, Practical Nonparametric Statistics, John Wiley & Sons, Inc., New York, 1971.

[14] K. Tatara, T. Tabata, K. Sakurai, A Probabilistic Method for Detecting Anomalous Program Behavior, In Proceedings of The 5th International Workshop on Information Security Applications, Jeju, 2004.