L-031

SVM を用いた Windows 向け異常検知システムの実装と評価
Implementation and Evaluation of Windows Anomaly Detection System Using SVM

伊波靖 † Yasushi IHA
高良富夫 ‡ Tomio TAKARA

1. はじめに

インターネットの急速な進展とともに，ウイルスやワーム等の不正なプログラムの出現頻度や伝搬速度が脅威となっている．それに伴い，従来のシグネチャベースの不正プログラム対策ソフトウェアでは，多岐にわたる亜種や 0-day ウイルスへのリアルタイムな対応が限界となりつつあり，新たな解決策としてビヘイビアベースの不正プログラム対策ソフトウェアの研究が盛んに行われている．

UNIX 系の OS においては，システムコール列の学習による侵入検知の研究が早くから行われ，多くの手法が提案されてきた [1][2].

一方，Windows 系の OS においては危険なシステムコールによる OS への攻撃を予め登録されたアクセス制御データベース (ACD: Access Control Database) に基づき検知し，実行を阻止する WHIPS(Windows Host Intrusion Prevention System) と呼ばれるシステムの提案が行われている [3]．しかし，このシステムは ACD に予め危険なプロセスとシステムコール及び引数の組をルールとして登録する必要があり，未知の危険なプログラムへの対応が困難であった．また，我々は Windows 系の OS において OS の資源に危険を及ぼすクリティカルなシステムコールを，システムコールと引数の組み合わせによるルールのみで検知し，検知したクリティカルなシステムコールが不正なプログラムによって発行された危険なシステムコールかどうかを SVM(Support Vector Machine) を用いて識別し阻止する異常検知手法を提案した [4]．本研究では，我々が提案した SVM を用いた異常検知手法を WHIPS へ実装することで，高い検知率と False Positive の割合を減少させた異常検知システムを開発し，評価実験によりその有効性を示すことを目的とする．

2. 危険なシステムコールの定義

Battistoni らは，システムコールとシステムコールの引数及びシステムコールを発行したプロセスの組み合わせにより危険なシステムコールを定義した [3]．ここでは Battistoni らの研究に基づき，危険なシステムコールを以下のように定義する．

定義 1. 危険な引数とは，システムの可用性に対して影響を与える可能性がある引数である．

定義 2. クリティカルなシステムコールとは，危険な引数を伴って発行されるシステムコールである．

定義 3. 危険なシステムコールとは，悪意を持ったプログラムによって発行されるクリティカルなシステムコールである．

例えばファイルを作成するシステムコール NtCreateFile は，それ自体は危険なシステムコールとは言えないが，システムディレクトリへのファイルの新規作成や上書きを意図した引数が与えられた場合に，クリティカルなシステムコールとなる．そして，それが，不正なプログラムによって発行された場合に危険なシステムコールとなる．また，NtWriteFile も，それ自体は危険なシステムコールとは言えないが，危険な NtCreateFile によって得られたファイルハンドル (この場合危険な引数となる) を引数として与えられた場合にクリティカルなシステムコールとなる．

3.WHIPS の概要

WHIPS は Windows 上で動作する侵入防止システムの一種で，予め ACD に危険なプロセスとシステムコール及び引数をルールとして登録しルールベースにより異常検知を行う．現在，オープンソースソフトウェアとして GPL で公開されている．WHIPS の動作概要を図 1 に示す．

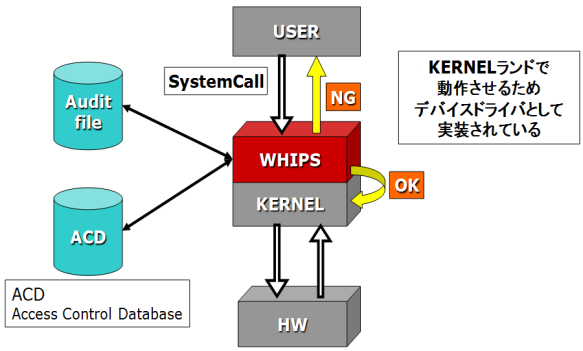

図 1: WHIPS の動作概要

WHIPS の特徴としては，ACD に設定されたルールに基づきシステムコールの実行を許可または拒否する．問題点としては，ルール設定が難しく拒否するシステムコールおよび引数をプロセスごとに設定する必要があるため高度な専門知識が必要となる．そのため，新しい不正プログラムへの対応が難しい．

4.Support Vector Machine(SVM)

SVM は統計的学習理論に基づく新しい 2 クラスのパターン認識手法であり，ニューラルネットワークなどの従来法と比較して汎化能力が高い点と最適解が求まる点に特徴があり，学習に用いていないデータに対して

†沖縄工業高等専門学校メディア情報工学科
‡琉球大学工学部情報工学科

も高い認識率を示す [5]．SVM がこのような特徴を示すのは，その学習に認識誤りと汎化性能の両面から最適化が行われ，これが 2 次の凸計画問題として定式化されているため最適解を求める事ができるためである．

SVM は線形しきい素子モデルにマージン最大化という基準で学習を行う．2 クラスの識別においてマージンとは，2 つのクラスと分離超平面との最小距離をいい，SVM では，このマージンを最大にする分離超平面が最も汎化能力が高いものと判断する．

SVM は学習の最適解として求められた分離超平面による線形識別を行っているが，学習用データを線形分離することが不適切な場合，学習データを元のパターン空間からより高次のパターン空間に非線形写像を行い高次元空間で分離超平面を構築し線形識別を行う．

SVM における分離超平面の概要を図 2 に示す．

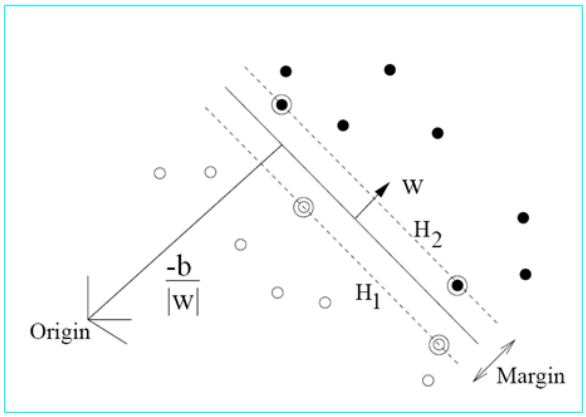

図 2: SVM における分離超平面

5. 危険なシステムコールに着目した Windows 向け異常検知手法

ここでは，我々が提案した異常検知手法の概要について説明する．

5.1. クリティカルシステムコールデータベースの構築

過去の不正なプログラムの分析から，クリティカルなシステムコールとなりうるシステムコールと引数のリストを予め作成し，クリティカルシステムコールデータベース (CSCDB) を構築してしておく．

5.2. システムコールの監視

システムコールの監視手法として SSDT(System Service Descriptor Table) の書き換えを使用する．SSDT をフックすることで，ユーザランドのプログラムからカーネルランドの OS の機能を呼び出す際に引数の取得や呼び出したプロセスの情報などを所得することが出来る．

5.3. クリティカルなシステムコールの検知

システムコールの監視を行い保護すべき資源に対して影響を与えるシステムコールが発行された際に，CSCDB と照合しクリティカルなシステムコールかどうかを判断する．なお，Battistoni らの研究では，検知の際にプロセスの名称も用いているが，本研究では，システムコールと引数のみでクリティカルなシステムコールの検知を行っている．

5.4.SVM による危険なシステムコールの認識

クリティカルなシステムコールと判断した場合は，そのシステムコールより過去に発行されたシステムコールの時系列について SVM を用いて危険なシステムコールかどうかを判断する．SVM で用いる素性データは，システムコール時系列データおけるシステムコールを要素番号，システムコールの頻度を要素の値とするペアを用いた．なお，SVM は予め正常なプログラム及び不正なプログラムの学習データによって学習を行っている．

6.WHIPS へ実装した機能

ここでは，WHIPS へ実装した機能について説明する．

6.1. ルールの拡張

WHIPS のルールを拡張しルールの汎用化を行った．プロセス名をワイルドカードで表現することで，クリティカルなシステムコールのみで検知できるようにした．また，複数システムコールの組合せによりクリティカルなシステムコールを検知できるようにした．これは，単一のシステムコールをルールでチェックするより，複数のシステムコールをブロックとして扱ってチェックした方が識別率が高くなるためである．

6.2.SYSENTER フック

Windows 系 OS においてシステムコールの監視には SSDT を書き換えフックすることで，監視するシステムコールを個別に指定可能にする方法とユーザランドからカーネルランドへ遷移する際に使用される命令 SYSENTER をフックすることにより全ての System Call を同一のフック関数で監視可能にする方法がある．WHIPS では SSDT フックのみを使用していたが，システムコールの時系列を作るために，全てのシステムコールをプロセス ID 毎に作成したキューに格納する必要があり，SSDT によるフックだけでは，全てのシステムコールについて処理関数を用意する必要があるため SYSENTER でフックすることで，システムコール時系列の採取する処理関数を一つにまとめることが出来る．SYSENTER フックの概要を図 5 に示す．

6.3.SVM による危険なシステムコール識別

SVM で識別を行うためには特徴ベクトルを定義する必要がある．本研究では，ルールによってクリティカルなシステムコールを検知した際に，SYSENTER フックにより採取したクリティカルなシステムコールにいたるシステムコール履歴の時系列から各システムコールの N-gram における頻度を特徴ベクトルとしている．特徴ベクトルの生成方法を図 4 に示す．

生成した特徴ベクトルを用いて， SVM により危険なシステムコールかどうか識別する．なお，SVM には

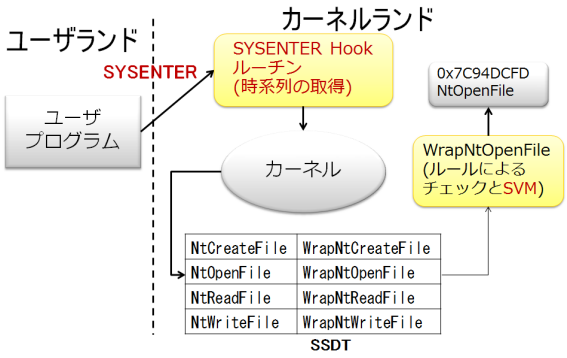

図 3: SYSENTER フックの概要

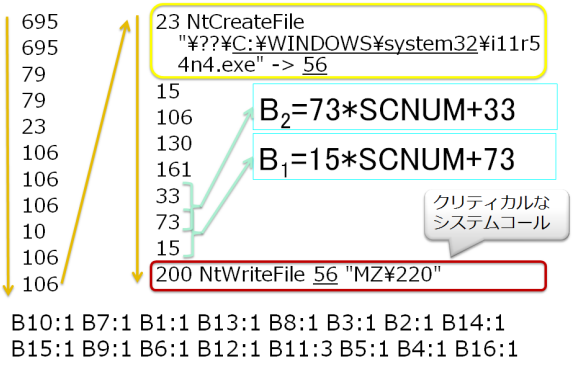

図 4: 特徴ベクトルの生成方法

SVMlight を用いた．

WHIPS へ実装した部分の概要を図 5 に示す．

7. 評価と考察

7.1. 学習データの収集方法

本提案方式の有効性を確認するために認識実験を行った．学習用の正常なプログラムは，ソフトウェア配布サイトから無作為にダウンロードした 140 種類のインストーラプログラムとした．また，不正なプログラムはウイルス及びワームを中心として 140 種類とした．正常及び不正なプログラムについて仮想環境にインストールされた Windows XP 上で実行し，システムコールの監視によりシステムコール列をログとして取得し，N-gram 法により特徴ベクトルを生成した学習データを用いて SVM を学習し，SVM のモデルを構築した．なお，予備実験の結果から時系列の長さ L = 200，N-gram の N = 4 とした．

7.2. クリティカルなシステムコール検知結果

取得したシステムコール列を用いて提案方式によるクリティカルなシステムコールの検知率を調べるために行った検知実験の結果を表 1 に示す．CSCDB に基づく検知により，不正なプログラムによるクリティカルなシステムコールを 100%検知することが可能となっている．なお，正常なプログラムとしてインストーラプログラムによるログを用いているため，正常なプログラムにおいても検知率が高くなっている．このことより不正なプログラムの挙動とインストーラプログラムの挙動は様々な面で類似性が高く，ルールベースのみで識別するのは困難であることが分かる．

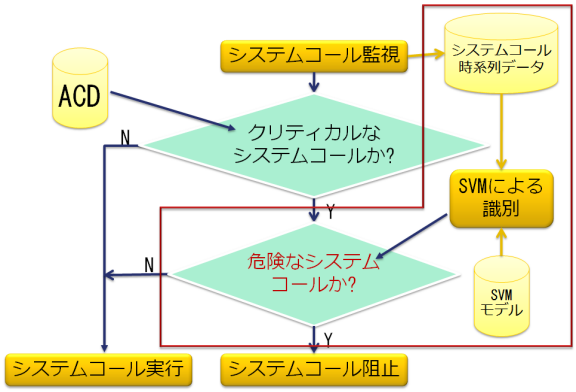

図 5: WHIPS への実装

表 1: クリティカルなシステムコール検知結果

不正なプログラム		正常なプログラム	
学習用	認識用	学習用	認識用
100.0%	100.0%	95.0%	100.0%

7.3. 危険なシステムコール認識結果

クリティカルなシステムコールとして検知されたシステムコールについて，システムコール時系列を素性として SVM により認識した結果を表 2 に示す．

なお，表中の認識率は，検知したクリティカルなシステムコールを正常なプログラムによるものと不正なプログラムによるものとに正しく認識できた結果である．FP(False Positive) は，正常なプログラムによるものを不正なプログラムとした結果で，FN(False Negative) は，不正なプログラムによるものを正常なプログラムとした結果である．

認識結果から一部のシステムコールを除き提案方式による危険なシステムコールの認識率が高く，FN が低いことが分かり，提案方式の有効性が高いことが言える．

表 2: 危険なシステムコール検知結果

NtWriteFile			NtSetValueKey		
認識率	FP	FN	認識率	FP	FN
96.3%	11.2%	1.4%	97.0%	66.7%	0.3%
NtMapViewOfSection			NtReadFile		
認識率	FP	FN	認識率	FP	FN
97.0%	50.0%	1.5%	83.2%	9.0%	55.2%

7.4. 検知実験の結果

通常のプログラム (インストーラー) と実際のマルウェアをそれぞれ 5 種類リアルタイムで実行させ検知可能かどうか実験を行った．検知実験の結果を表 3 に示す．実験結果からインストーラーはクリティカルなシステムコールを発行しているため，ルールベースでは不正となるが SVM により通常のプログラムとして認識を行っていることが分かる．また，マルウェアについてはルールベースおよび SVM の両方で不正なプログラムとして検知が行われている．この結果から，リアルタイムでの検知が可能となっていることが言える．

表 3: 検知実験の結果

	プログラム名	ルールベース	SVM 識別
通常	A-Downloader320	不正	正常
	CravingExplorer-0-18-8	不正	正常
	GOMPLAYERJPSETUP	不正	正常
	Lhaca124	不正	正常
	XeloPDF201	不正	正常
マルウェア	Antinny	不正	不正
	Bagle	不正	不正
	Gibe	不正	不正
	NetSky-b	不正	不正
	Sasser	不正	不正

8. まとめ

ルールベースでクリティカルなシステムコールを検知し，SVM により危険なシステムコールを識別する異常検知システムを WHIPS に実装した．実装したシステムでクリティカルなシステムコールをルールベースにより高い認識率で検知することを可能とした．また，システムコールの時系列を素性とした SVM により，従来のルールベースによる検知では困難であった，インストーラーと不正なプログラムを識別することを可能とした．今後の課題としては，性能評価実験によるオーバーヘッドの測定とさらなるチューニングと One Class SVM の実装により，正常なプログラムのみの学習データで SVM のモデルを構築できるようにすることが上げられる．

謝辞

本研究は科研費 23500106 の助成を受けたものである．

参考文献

[1] Forrest, S., Hofmeyr, S. A., Somayaji, A. and Longstaff, T. A.: A sense of self for Unix processes, *IEEE Symposium on Security and Privacy*, pp. 120–128 (1996).

[2] Sekar, R., Bendre, M., Dhurjati, D. and Bollineni, P.: A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors, *IEEE Symposium on Security and Privacy*, pp. 144–155 (2001).

[3] Battistoni, R., Gabrielli, E. and Mancini, L. V.: A Host Intrusion Prevention System for Windows Operating Systems, *Proceedings of the 9th European Symposium On Research in Computer Security (ESORICS 2004)*, pp. 352–368 (2004).

[4] 伊波靖，高良富夫：危険なシステムコールに着目した Windows 向け異常検知手法，情報処理学会論文誌，Vol. 50 No. 9, pp. 2173–2181 (2009)

[5] Cristianini, N. and Shawe-Taylor, J.: サポートベクターマシン入門，共立出版 (2005).