

SVMを用いたシステムコール履歴に基づく異常検知システムの BitVisor への実装

伊波靖[†]HENDRA GUNTUR[‡][†] 沖縄工業高等専門学校メディア情報工学科

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マルウェアの出現頻度や伝搬速度が深刻な脅威となっている。それに伴い、従来のシグネチャベースのマルウェア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プログラムを実行させ呼び出されたシステムコールの履歴に基づいて検知する手法が提案されている。我々も Windows の動作に影響を与えるクリティカルなシステムコールをルールベースで検出し、検出したクリティカルなシステムコールが悪意を持ったプログラムによって実行された危険なシステムコールなのかを、システムコールの履歴から生成した特徴ベクトルを用いて SVM(Support Vector Machine) で判別するシステムを提案した [1]。しかし、例えば異常検知システムがカーネルモードで動作していても、カーネルに感染することで自らを秘匿したり、マルウェア検知システムを無効化するルートキットと呼ばれるマルウェアによって異常検知システムが無効化されるリスクが存在している。

そこで、本研究では、セキュア仮想計算機モニタ (VMM) である BitVisor を機能拡張することで、VMM において Windows のシステムコールをフックしてシステムコールの履歴を取得し、N-gram 法により特徴ベクトルを生成し、SVM によりマルウェアを検知する異常検知システムを提案する。本システムにより、ルートキットからの攻撃に対して堅牢で検知率の高い異常検知システムを構築する。

2 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も高い識別率を示す。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 [2]。

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。なお、本研究では SVM として SVMlight を採用している [3]。

3 BitVisor の概要

本研究では、筑波大学が中心となって開発した準パススルー型 VMM である BitVisor をベースとして開発を行っている [4]。BitVisor 上では一つのゲスト OS しか実行できないが、ゲスト OS に依存しないデバイス単位でのアクセス制御やディスク、ネットワークパケットの暗号化を行うことが出来る。BitVisor をベースとして Windows のマルウェア解析を行うためのシステムとして実装されたものとしてマルウェアアナライザ Alkanet がある。本研究では、Alkanet のシステムコール取得方法を参考に Windows のシステムコール履歴を取得している [5]。

4 提案手法の概要

4.1 Windows システムコール履歴の取得方法

本研究では対象となる Windows を Windows XP SP3 とし、BitVisor 上のゲスト OS として稼働させ、ゲスト OS 上のプログラムが発行したシステムコールを取得するために、システムコール呼び出しに使用される SYSENTER 命令の開始アドレスに INT3(0xCC) をセットし、SYSENTER 命令が呼び出されたときにゲスト OS から VMExit によって BitVisor へ制御が移るようにした。BitVisor では、VMExit 時点のゲスト OS の CPU レジスタ EAX と EDX を取得する。EAX にはシステムコールの番号が、EDX には SYSENTER 命令を実行した際のスタックポインタの値、すなわちシステムコールの引数の格納場所が格納されている。次にゲスト OS 上の KPRCB(Kernel Processor Control Block) 構造体をゲスト OS 上から取得し、現在実行しているスレッドを示す CurrentThread の値から現在実行されているプロセスの情報を取得してプロセス単位

Implementation of BitVisor of an anomaly detection system based on the history of system calls using SVM.

[†]Yasushi IHA (yasuc@okinawa-ct.ac.jp)

[‡]HENDRA GUNTUR

[†]Department of Media Information Engineering, Okinawa National College of Technology

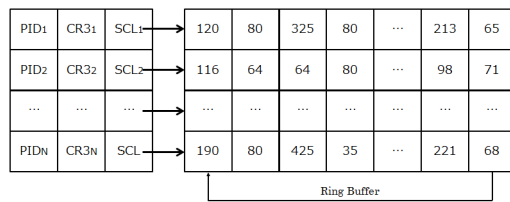


図 1: システムコール履歴を管理するデータ構造

で一意に割り当てられる UniqProcessID を取得する。取得した EAX, と UniqProcessID を図 1 に示すシステムコール履歴を管理するデータ構造によって管理する。なお、図 1 においてシステムコール履歴を管理する部分はリングバッファで管理している。

4.2 システムコール時系列から特徴ベクトルの生成

SVM により識別を行うためには、システムコール発行履歴の時系列が持つ特徴を素性ベクトルとして表現する必要がある。提案方式では、素性ベクトルとして、言語モデルとしての広く採用されている N-gram モデルを採用した。具体的には、システムコール時系列から得られた N-gram と N-gram の共起頻度によって特徴表現した。

図 2 に $N = 2$ の場合の N-gram モデルによる特徴表現の概要を示す。クリティカルなシステムコールが検知された場合、それ以前のシステムコール時系列より N-gram 法を用いて N-gram 集合を抽出する。抽出したそれぞれの N-gram は、ハッシュ関数を用いて要素番号 N_i に変換する。次に、変換した要素番号 N_i について重複するものの頻度 h_i を求める。素性ベクトルは、要素番号 N_i を昇順にソートし、対応する頻度 h_i をペアにすることで生成する。2 から生成された素性ベクトルは

$f = \{N1 : 1, N2 : 1, N3 : 1, N4 : 1, N5 : 1, N6 : 1, N7 : 1, N8 : 1, N9 : 1, N10 : 1, N11 : 3, N12 : 1, N13 : 1, N14 : 1, N15 : 1, N16 : 1\}$

となる。なお、特徴ベクトルの次元数が大きく、また疎で冗長なデータ空間となるため、次元数を減らして密なデータ空間に変換するためにハッシュ関数を用いている。

5 BitVisor への SVM の実装

BitVisor への SVMlight の実装には、浮動小数点演算および数値計算ライブラリとして pow, tanh, exp, sqrt 等が必要となる。BitVisor は浮動小数点演算の処理を使わずに実装されているため、浮動小数点演算を行った場合例外が発生しゲスト OS の実行が停止してしまう。これは CR0 レジスタの TS(Task Switch) ビットが立っていると浮動小数点演算が行われた際に例外が発生するためである。

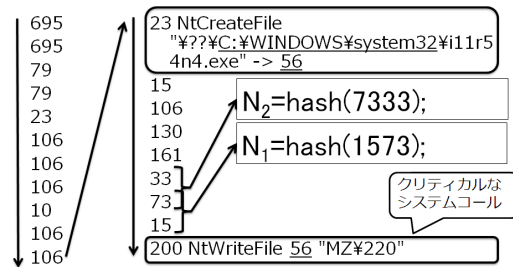


図 2: N-gram 法による素性データ

そこで、SVM の関数内では浮動小数点演算が可能になるように、関数の入り口で CR0 レジスタの TS ビットをリセットし、関数からリターンする際に TS ビットをセットしている。また BitVisor には数値計算ライブラリが用意されていないため、組み込み Linux 向けの小型標準 C ライブラリである uClibc から必要な数値計算関数をリンクすることで実装を行った [6]。

6 まとめと今後の課題

本研究では BitVisor に Windows のシステムコール履歴を取得する機能を実装し、取得したシステムコール履歴から N-gram 法により特徴ベクトルを生成した。また、SVM を BitVisor に実装するために数値演算ライブラリを組み込み数値計算関数の実行と浮動小数点演算を可能にした。

今後の課題としては、SVM のチューニングと実環境における性能評価と検知性能向上が挙げられる。

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] 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).
- [2] Cristianini, N. and Shawe-Taylor, J.: サポートベクターマシン入門, 共立出版 (2005).
- [3] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [4] BitVisor Project: BitVisor1.3, <http://www.bitvisor.org/> (2012).
- [5] 大月勇人, 瀧本栄二, 樫山武浩, 毛利 公一: マルウェア挙動解析のためのシステムコール実行結果取得法, Computer Security Symposium 2011, pp.95-100(2011).
- [6] uClibc: A C library for embedded Linux, <http://www.uclibc.org/> (2012)