System Service 監視による Windows 用異常検知システムの開発

島本大輔[†]　大山恵弘[†]　米澤明憲[†]

1. はじめに

近年，計算機に対する大きな脅威として，多層型ウィルスなどの悪質なソフトウェアが日増しに数を増やしており，単純なシグネチャの比較では対処しきれていない場合がある．その対抗策として，UNIX 系 OS においては，system call 列など，プログラムの振る舞いを監視することにより異常検知を行う研究が多数なされている．具体的な手法は同じ system call 列を利用するものでも多岐に渡る[1)2)3)]．

Windows OS ではこのような先進的な手法を適用した研究は少ない．Windows OS は UNIX 系 OS と異なる特徴を持つため，UNIX 系 OS 向けに提案されてきた既存の異常検知手法が Windows OS 上でも有効かどうかは自明ではない．例えば，Windows OS の特徴として，大半のプログラムモジュールの API や内部構造が非公開であることや，OS サービスのインタフェースが低レベルであり，インタフェースの関数の数が非常に多いことがある．

本研究の目的は，Windows OS 上でも有効に機能するような，一般化された異常検知手法を構築することである．本研究では，まず，Windows OS に対して，UNIX 系 OS 向けの既存手法が有効かどうかを検証する．有効でない場合には，既存の異常検知手法の問題を解決した新しい異常検知手法を構築することを目指す．

2. System Service とその監視

Windows OS では，System Service が UNIX 系 OS の system call に対応しており，Windows OS のカーネルの機能を提供している．本研究では，これを監視することにより異常を検知する．この手法により，従来のパターンマッチでは検出できない未知のウィルスなども検出することが可能になると思われる．

System Service の監視や動作の改変は，不正侵入後にシステムを操作するためのルートキットなどに多く使われている．既存の手段としては，System Service のアドレスが格納されている System Service Descriptor Table (SSDT) を書き換える手段[4)] がある．SSDT は UNIX 系 OS における system call table に相当するものであり，SSDT を書き換えることは system call table を書き換えることに相当する．system call table を書き換える手法を利用した既存の研究は多く存在する．しかし，Windows の次期バージョンである Windows Vista では SSDT を書き換えられないようにされるため，この手法は利用できなくなる．

本研究では，SSDT の書き換えではなく，カーネルモードに遷移する際に呼ばれる SYSENTER 命令[5)] の動作を変更することで監視する．SYSENTER 命令は，System Service や system call に特化した x86 アーキテクチャの命令であり，カーネルモードへの遷移だけでなく，スタックポインタ (SP) の変更，インストラクションポインタ (IP) の変更も同時に行われる．SYSENTER 命令が呼ばれたときには SYSENTER_EIP_MSR レジスタに書き込まれているアドレスに実行が移る仕組みになっており，本研究ではこのレジスタを書き換え，用意した監視コードへ実行を移すようにしている．ユーザーモードから呼び出された System Service はすべて SYSENTER を実行し，呼び出された System Service によらず，一度共通のアドレスへ実行が移るため，このアドレスを変更し，自分のコードに実行を移すことにより，システム全体を監視することが可能となる．また，SYSENTER が実行された際に元の Service Service の ID が eax レジスタに格納されるため，この値を利用し，System Service の識別を容易に行うことが可能となる．

3. System Service 列を利用した異常の検知

本研究では上記の方法で System Service のログを取得し，まずは単純にその列を監視することにより，異常を検知する．

具体的には，監視対象のプログラムを実行し，正常動作時の System Service 列を固定長部分列すべてを記録しておき，監視する際に記録と比較し，記録に含ま

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻

れていない列が生じた場合に異常を通知する．この方法は 1 つのプログラムに対する異常を監視する場合であるが，将来的には，悪意あるプログラムや異常時の System Service 列を調査し，あらゆるプログラムから，未知の悪意あるプログラムを検知するよう拡張する予定である．

4. 実　　装

本研究で開発したソフトウェアはデバイスドライバと GUI の 2 つの部分からなる．

デバイスドライバは，SYSENTER 命令のジャンプ先の変更および，監視ログの生成に利用される．デバイスドライバを実行するとユーザーモードから System Service 内の SYSENTER 命令が実行されるとデバイスドライバ内に監視コードへ実行が移る．

現段階では，呼び出し元のプロセスが監視対象である場合のみ，呼び出し元の Thread ID と System Service を識別する Service ID を記録する．その後，デバイスドライバを適用する以前のジャンプ先に実行を移し，通常の実行を継続させる．図 1 は本ソフトウェア適用前と適用後の動作フローを示している．

図 2GUI 部分はこのデバイスドライバの操作，デバイスドライバからのログ読み出しや保存を行う．プロセスは新規に作成する場合と実行中のものを監視する場合の 2 種類が選べるようになっている．System Service の記録は列をすべて記録するものとユーザーが設定した長さのコール列を記録する．列をすべて記録するログファイルはテキストとバイナリの 2 つから選べるようになっている．

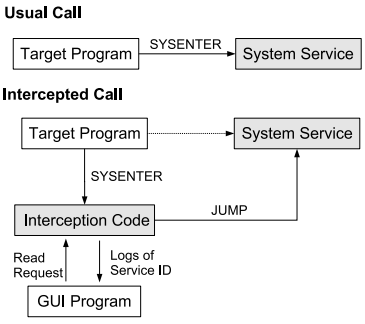

図 1　本ソフトウェアの動作図

5. 現在の状況および今後の予定

現在，任意の長さの System Service 列を記録することが可能となっており，今後はあるプログラムに対し，正常動作時の記録を用いて監視状態に置き，実際

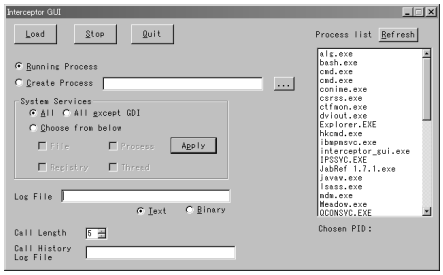

図 2　GUI 部分

に脆弱性などを突いた攻撃を検知できるか確認していく予定である．

また，正常なものから悪意ある様々なプログラムの記録を収集し，それらを解析することにより，System Service 列のみから異常を検知が可能か確認する予定である．

今後はその結果に応じて，スタックの状態などの追加の情報が必要と思われた場合は機能を拡張していく．さらに，それらを利用した異常検知の機能もソフトウェアに組み込み，異常検知システムとして完成させる予定である．

参　考　文　献

1) Forrest, S., Hofmeyr, S. A., Somayaji, A. and Longstaff, T. A.: A Sense of Self for Unix Processes, *Proceedinges of the 1996 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, pp. 120–128 (1996).
2) Feng, H., Kolesnikov, O., Fogla, P., Lee, W. and Gong, W.: Anomaly detection using call stack information (2003).
3) Warrender, C., Forrest, S. and Pearlmutter, B. A.: Detecting Intrusions using System Calls: Alternative Data Models, *IEEE Symposium on Security and Privacy*, pp. 133–145 (1999).
4) Russinovich, M. and Cogswell, B.: Windows NT System-Call Hooking, *Dr. Dobbs Journal* (1997).
5) Intel(R): *IA-32 Intel(R) Architecture Software Develope's Manual Volume 2B: Instruction Set Reference, N-Z* (2004).