Detecting Intrusions on Windows Operating Systems by Monitoring System Services

島本 大輔 (新M1)

2005年3月28日

1

現在のセキュリティ事情

- インターネットの成長によるattackによる危険性の増大
  - ウィルス、不正侵入、スパイウェア、など
- 対策も発展途上
  - ウィルス対策ソフト、ファイアウォールなども完璧ではない
  - 新種も検出できることが望ましい
    - いつゼロデイアタックが起きてもおかしくない

2

Intrusion Detection System (IDS)

- 侵入検知システム
  - ホストやネットワークの様々な情報を監視し、侵入されたことなどを検知するシステム

3

IDSの種類 (1)

- Signature based vs Anomaly based
  - Signature based
    = ウィルスなどの signature とパターンマッチ
  - Anomaly based
    = 通常と異なる、異常な動作を検知

4

IDSの種類 (2)

- Host based vs Network based
  - Host based
    = ホストの情報 (プロセス、ファイル、など) を監視
  - Network based
    = ネットワーク上のパケットを監視

5

既存のIDS

- 既存のWindows用IDS
  - Signature based が主流
    ＝新種への対応が困難
  - 研究は少数
- UNIX系OSにおけるIDS
  - 様々な先進的な研究が多数

6

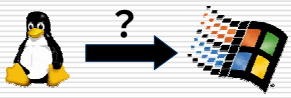
UNIX系OSにおけるIDS

- プロセスの様々な情報を監視
  - [Forrest 96] の system call の監視
  - [Feng 03] のスタックの監視
  - [Sekar 01] のプログラムカウンタの監視

7

Windowsへの応用

- UNIX系の技術をWindowsへ応用？
  ⇒ 困難
  - ブラックボックスなOS
  - 公式に提供されているモジュールでは不十分



8

目標

- 『UNIX系OSにおける先進的なIDSの研究をWindowsへ応用したい』

➡ Windowsでsystem callなどの情報を使って異常を検出

9

研究内容

- Windowsにおける先進的なIDS
  - System Service (Windows版system call) を利用して、異常を検知

10

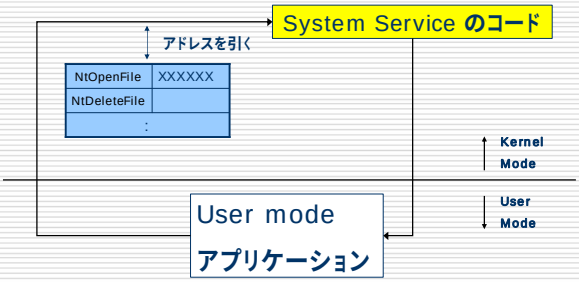
System Service

- UNIX系OSのsystem callに対応
- Windowsの根本的な機能を提供
  - ファイル、レジストリ、プロセス、スレッド
    タイマー、mutex、GDI、など

  例：NtWriteFile — ファイルへの書き込みはすべてこれを利用
- System callに比べ、数は非常に多い
  - 286個 (Windows 2000)
  - 991個 (　〃　XP)

11

System Service の動作



12

System Service の Interception

- 2通りの手法が存在
  1. SSDT Patching
  2. Interrupt Hooking

13

SSDT Patching

- System Serviceのアドレステーブル (System Service Descriptor Table) を書換
- System call tableの書き換えと同様

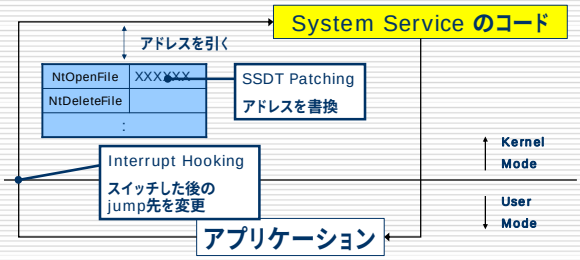
14

Interrupt Hooking

- Kernel mode へスイッチする瞬間に intercept
  - Windows 2000 以前ではソフトウェア割り込み (int 2e)
  - Windows XP 以降は SYSENTER

15

System Service の Interception



16

System Serviceの Interception

- Interrupt hooking を採用
  - 既存研究が少数
  - 一箇所において監視可能
    - SSDT patchingでは監視コードが散在
  - 監視する System Service の追加･削除が容易

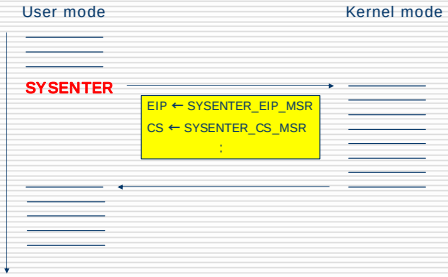
17

SYSENTER 命令

- Fast System Call
  - System call をするために必要な権限の移行に特化した命令
    - セグメントセレクタ (SS)、インストラクション ポインタ (IP)、スタックポインタ (SP) を1命令で あらかじめ設定された値に変更

18

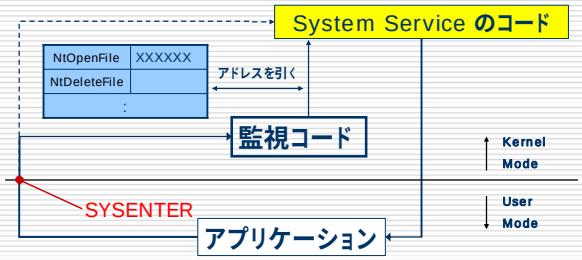
SYSENTER の動作図



SYSENTER で Interception

- SYSENTER 実行後の jump 先を監視コードのアドレスへ変更
  - 自分のコードが終了後に元のSystem Serviceのコードへ jump
    ⇒引き続き元の動作を実行
  - 具体的には SYSENTER_EIP_MSR レジスタを WRMSR 命令で書き換え

System Service の Interception



実装内容

- デバイスドライバ部分
  - Interception コードの挿入等を担当
  - Kernel mode で動作
- GUI プログラム
  - デバイスドライバの操作を担当
  - User mode で動作

デバイスドライバ部分

- 基本的にC言語で記述
  - SYSENTER_EIP_MSR の書き換えは インラインアセンブリ
- 挿入するコードもデバイスドライバの中に存在

以前のコード例

```
sti
mov ecx, 176h
rdmsr
mov SYSENTER_EIP_MSR_H, edx
mov SYSENTER_EIP_MSR_L, eax
cli
mov ecx, 176h
xor edx, edx
mov eax, stub
wrmsr
sti
jmp endasm
```

```
stub:
   pushad
   cmp eax, 30h  /* CreateProcess の Service ID */
   je log
normal:
   popad
   jmp [SYSENTER_EIP_MSR_L]
log:
   jmp normal
endasm:
```

ここにコードを追加することにより拡張可能

比較部分の変更

- 変更前
  - cmp命令 ⇒ 監視対象が増えると比例して時間がかかる
  - そもそもどうやって、追加するのか
- 変更後
  - 監視するものに関してフラグを立てる
    - 定時間で比較が終了
    - 監視対象の変更が容易
- 要は if-else の列を表引きに変更

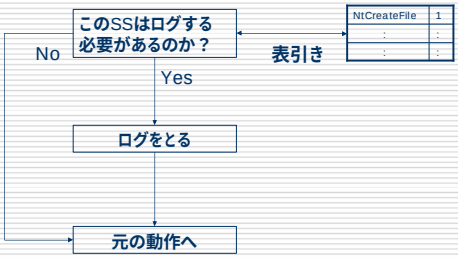
25

変更後のコード

```
if (sid >= 0x1000) {
 ssindex = sid-0xEE5;
}
else {
 ssindex = sid;
}
if (ssflags[ssindex] == 0x01) {
 // fs を 0x30 にセットして Process ID を取得する
 __asm {
      push fs
      mov ax, 0x30
      mov fs, ax
      mov eax, fs:[0x124]
      pop fs
      mov eax, [eax+0x44]
      mov pid, eax
 }
 // Service IDとProcessIDを保存する
 callLog[numLogged].ServiceID = sid;
 callLog[numLogged].ProcessID = pid;
 numLogged = (numLogged+1)%MAXLOG;
}
```

26

変更後のフロー



27

GUIプログラム

- デバイスドライバの操作
  - ロード、アンロード
- デバイスドライバとの通信
  - ログをデバイスドライバから読み出し、出力
    - バイナリとテキスト出力に対応
  - InterceptするSystem Serviceの変更をデバイスドライバに伝達

28

問題点

- Interrupt hooking の弱点
  - Kernel mode内からのSystem Serviceの呼び出しを監視不可能

⇒
- SSDT patchingの導入？
- ターゲットをuser modeプログラムに限定？

29

関連研究 (1)

- System call の監視を利用したもの
  - [Forrest et al. '96]
    - パイオニア的存在
  - [Sekar et al. '01]
    - プログラムカウンタを監視対象として導入
    - オートマトンを導入
  - [Feng et al. '03]
    - Return address を call stack から抽出し、監視

30

関連研究 (2)

- System Service 関連
  - [Russino '97]
    - SSDT Patching をはじめて紹介
  - [Battis et al. '04]
    - SSDT Patching を利用
    - モニタプロセスで System Service を監視
  - Strace for NT
    - http://www.bindview.com/
    - 同じく SSDT Patching を利用

31

関連研究 (3)

- 「未知のウィルス」対策
  - Bloodhound (Symantec)
    - Symantec がアンチウィルスに採用
    - VM内で動作させ、その挙動を signature とマッチング
  - TruPrevent (Panda Software)
    - プログラムの振る舞いを監視
    - 他のアンチウィルスとの共存を前提
    - doesn't generate false alarms ！

32

関連研究 (4)

- 「未知のウィルス」対策 (続)
  - 日立
    - 2005/03/18 の発表
    - 重要なファイルを並べた定義ファイルを作成
    - それらのファイルに対する変更・差し替えなどを監視

33

これからの予定

- プログラムの改良
  - IDS以外にも利用できるように開発

  例：ログを読みやすいように整形するソフト、など
- パターンの収集
  - ウィルスのパターン
  - 侵入のパターン
  - 通常の操作によるパターン
- IDSを完成させる

34

まとめ

- モチベーション
  - Windows における先進的なIDSの実装
- 今回の研究内容
  - Interception の手法を調査・実装
  - ウィルスなどのパターンを取得
- 今後の予定
  - IDS への拡張

35