卒業論文に向けて(5)

学部4年生
島本 大輔
2004年12月14日

1

概要

- 卒論内容
- 様々な Hook
- SYSENTER で Hook
- 今後の予定

2

卒論内容

- Windows 版 IDS
  - System Service の記録で検出
    - System Service = UNIX 系の System Call
  - 「どのように Hook するか」が問題

3

概要

- 卒論内容
- 様々な Hook
- SYSENTER で Hook
- 今後の予定

4

様々な Hook

- User-mode
  - Win32 の API を Hook [1]
    - 演習3の研究内容
- Kernel-mode
  - Native API を Hook

5

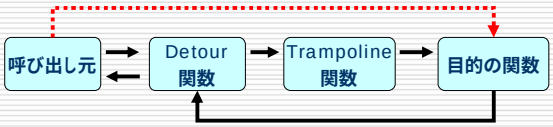
User-mode Hooking

- Proxy DLL
  - DLL を置き換える
- Function Patching
  - 関数の中身を書き換える
  - 例：Detours [2]
- IAT Patching [2,3]
  - Import Address Table の値を書き換える
  - Detours にもこの機能あり

6

Function Patching (Detours)

- 自分のコード(detour)を実行後、本当の API を呼び出す



7

Function Patching (Detours)

- 利点
  - ユーザー定義の API を Hook できる
- 欠点
  - API 内に「jmp *関数* =5 byte 分の容量が必要
    ⇒ 5 byte 未満の API は置き換えられない
  - Win32 の API は難しい
    ⇒ DLL 側で可能かもしれない

8

IAT Patching

- IAT = Import Address Table
- Import Address Table を書き換える
- Detours にもこの機能あり
- この手法の文献は多い

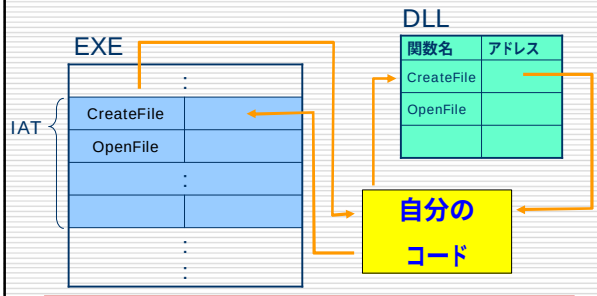
9

Import Address Table (IAT) [4]

- 外部ライブラリ(主に DLL)で呼び出す関数アドレスのテーブル
  - 当然、一定値ではない
- Windows Loader が DLL のアドレスをテーブルに書き込む
- 1つのバイナリに必ず1つある
  - エントリ数が0もあり(例 ntdll.dll)
- 逆の機能は Export Address Table

10

Using IAT for Hooking



11

IAT Patching

- 回避策がある
  - Getprocaddress で動的にライブラリの関数アドレスを引ける
- 実際、ウィルスやRootkitはそうしている
  - OS のバージョンによって、関数アドレスが異なるため

12

Kernel-mode Hooking

- Windows NT の System Service を hooking [5]
  - 過去に研究例あり [6]
- セキュリティへの応用の論文は少ない (と思われる)
- Rootkit では多々使われている

13

System Service

- Linux の System Call みたいなもの
- NT Executive (ntoskrnl.exe の一部) により提供される
- まさしく、Windows の核をなす
  - 例:Win32 CreateFile() と POSIX open() は NTCreateFile() を呼んでいる
- ほとんど公式にはドキュメント化されていない!

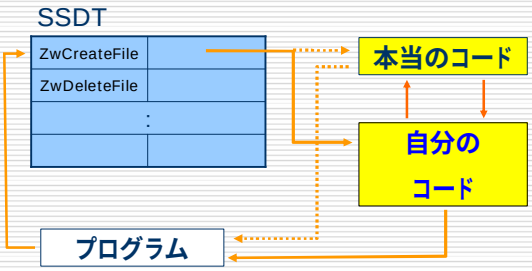
14

System Service Hooking [5]

- System Service はリスト (System Service Table (SST)) で管理 (UNIX系の System Call Table みたいなもの)
- SST は Service へのポインタを持つ
- このリストの値を変えれば良い
  ⇒ 自分の関数に書き換える

15

System Service Hooking



16

問題点

- OS のバージョンにより、ntoskrnl.exe が微妙に異なる
  ⇒ 各OSに対応させるのが面倒
- Hook する数が多くなると実装が大変

17

概要

- 卒論内容
- 様々な Hook
- SYSENTER で Hook
- 今後の予定

18

SYSENTER で Hook

- System Service は数が多い
  ⇒ まとめて監視できないか？
  つまり、共通の通過点はないのか？
- User-mode から System Service を利用するときに、Kernel-mode への移行が必要
  ⇒ Windows 2000 以前は int 2e
  Windows XP 以降は SYSENTER

19

SYSENTER命令とは [7]

- 1997年から加わっている命令
- Fast System Call
  System Call において必要な権限の移行に特化した命令
- セグメントセレクタ、IP、スタックポインタを変更する
- Linux でも 2.5 で採用（？）

20

SYSENTER で Hook

- SYSENTER_EIP_MSR から 次の IP を読み込む
  ⇒ この値を変更してしまう
- WRMSR 命令で変更可能
- RDMSR 命令で読み出し

21

コード（インラインアセンブリ）

```
push eax
push ecx
push edx
mov ecx, 174h  /* SYSENTER_CS_MSR */
rdmsr
mov SYSENTER_CS_MSR_H, edx
mov SYSENTER_CS_MSR_L, eax
mov ecx, 175h  /* SYSENTER_ESP_MSR */
rdmsr
mov SYSENTER_ESP_MSR_H, edx
mov SYSENTER_ESP_MSR_L, eax
mov ecx, 176h  /* SYSENTER_EIP_MSR */
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
pop edx
pop ecx
pop eax
jmp endasm
```

```
stub:
    pushad
    cmp eax, 30h  /* とりあえず CreateProcess だけ表示 */
    je log
normal:
    popad
    jmp [SYSENTER_EIP_MSR_L]
log:
    push eax
    push offset logMessage
    call DbgPrint
    add esp, 8
    jmp normal
endasm:
```

22

実演

23

概要

- 卒論内容
- 様々な Hook
- SYSENTER で Hook
- 今後の予定

24

今後の予定

- UIの作成
  - DbgView を用いずに結果を表示
  - ファイル出力等のログ機能
- SYSENTER での Hook を利用した IDS の作成
  - UNIX系での System Call を利用した例を参考

25

参考文献 (1)

1. API Spying Techniques
   http://www.internals.com/articles/apispy/apispy.htm
2. Detours
   http://research.microsoft.com/sn/detours/
3. Process-wide API spying – an ultimate hack
   http://www.codeproject.com/system/api_spying_hack.asp
4. An In-Depth Look into the Win32 Portable Executable File Format (Part 1 & 2)
   http://www.msdn.microsoft.com/msdnmag/issues/02/02/PE/default.aspx
   http://www.msdn.microsoft.com/msdnmag/issues/02/03/PE2/default.aspx

26

参考文献 (2)

5. Hooking Windows NT System Services
   http://www.windowsitlibrary.com/Content/356/06/1.html
6. A Host Intrusion Prevention System for Windows Operating Systems
   Roberto Battistoni, Emanuele Gabrielli, Luigi V. Mancini
   ESORICS 2004
7. IA-32 Intel® Architecture Software Developer's Manual

27