Detecting operating systems without Microsoft Advanced Programming Interface

Author: Thomas Krue – Universitas Virtualis
The Assembly-Programming-Journal, Vol. 1, No. 1 (2004)
Tranz by: Benina

Tiếp theo bài tut PEB&TEB structure, tôi xin dịch bài tut này để cho các bạn thấy được rõ hơn những kiến thức mà ta đã tìm hiểu được trong tut vừa qua. Vì là bài dịch, nên câu cú có thể ko hòan hảo, thậm chí hơi khó hiểu, mong các bạn thông cảm và nhớ đọc lại nhiều lần tôi đảm bảo các bạn sẽ hiểu thôi.

Bài viết này sẽ chỉ cho bạn cách dò tìm version của OS Mirosoft : Windows 95,98,Me – hệ điều hành non NT-based và Windows NT4,2000,XP,2003 – hệ điều hành NT-based- mà ko cần dùng hàm APIs để tránh kỷ thuật đảo mã (Reverse Engineer) set breakpoint trên hàm APIs.

I. Introduction

Khi phân tích cấu trúc của TEB (cũng được biết đến như là TIB: Thread Information Block) trên các OS khác nhau, chúng ta tìm thấy data thêm vào sau cấu trúc này sau khi startup . Data thêm vào này dường như (trên OS dựa trên nền tảng NT 'NT-based') có cấu trúc ko logical hay ko có chiều dài định sẳn, mà ở đó (data thêm vào)-đối OS dựa trên nền tảng NT -chứa thông tin bên trong PEB. Vậy chỉ còn cách để mô tả ý nghĩa của data này cho OS ko dựa trên nền tảng NT là debug cùng một ứng dụng trên các OS khác nhau.

II. Application start:

Có nhiều cách để dò tìm ra OS. Có thể thực hiện bằng cách dùng hàm API GetversionEx và checking giá trị trả về của version trong cấu trúc OSVERSIONINFO(EX).Hoặc bằng cách truy xuất thanh ghi CS. Cách khác là phân tích các thanh ghi registers trong lúc start up ứng dụng. Có các quy tắc đặc biệt đối với các registers khi OS "sửa chửa" các thanh ghi registers sẽ như thế nào trước khi thực thi chỉ thị đầu tiên:

Giá trị startup cho Windows 95/98/ME

EAX== Entry point ứng dụng
EBX==00530000h, một giá trị cố định

Các giá trị startup cho Windows NT/2000/XP/2003

EAX==NULL
EBX==7FFDF000h, con trỏ trỏ đến PEB

Bởi biết được các quy tắc này, chúng ta có thể check OS dựa trên suốt thời gian startup.Và rồi chúng ta chứa các giá trị thanh ghi của EAX và EBX để dành sử dụng trong tương lai- hay là dùng để giải quyết các vấn đề khác.

A.Thread Environment Block

TEB được "sửa chửa" trong suốt thời gian startup ứng dụng và chứa các pointers đến thread liên quan đến dữ liệu thêm vào cấu trúc TEB. Cấu trúc TEB structure có mặt trên tất cả các OS. Nó có kích thước được định nghĩa là 34h Bytes. TEB address có thể được tính tóan bằng cách truy xuất segment register FS theo cách sau đây:

```
assume fs:nothing
mov eax,fs:[18h]
```

Thanh ghi EAX sẽ chứa base address của block này. TEB chứa - tại address 18h trong cấu trúc - một pointer trỏ đến chính nó :

```
pSelf DWORD ? ; 18h pointer to TEB/TIB
```

Thành phần sau cùng của TEB là con trỏ trỏ đến process database. Trên OS dựa trên nền NT giá trị này sẽ trỏ đến address của Process Environment Block (xem Section II-C)

B. Additional data following TEB

Như đã nói trong phần Section I, data thêm vào này ko có cấu trúc logical và chỉ tham vấn được trên OS ko dựa trên nền NT . Trên Windows NT, 2000, XP và 2003 data thêm vào này được định nghĩa như sau:

```
NT_TEB_ADDON struct
LastErrorValue DWORD ? ; 00h (34h TEB)
LastStatusValue DWORD ? ; 04h (38h TEB)
CountOwnedLocks DWORD ? ; 08h (3Ch TEB)
HardErrorsMode DWORD ? ; 0Ch (40h TEB)
NT_TEB_ADDON ends
```

Windows 95, 98, ME ko có một cấu trúc như vầy; data thêm vào đã bị phá rối .

C. Process Environment Block

Hệ điều hành Windows dựa trên nền tảng NT chứa data liên quan đến process bên trong Process Environment Block. Address của cấu trúc này có thể lấy được bằng cách truy xuất vào segment register FS:

```
assume fs:nothing
mov eax,fs:[30h]
```

EAX sẽ chứa base address của PEB.

pProcess DWORD ? ; 30h pointer to process database

thông tin về Version được chứa bên trong cấu trúc PEB structure:

OSMajorVersion DWORD ? ; A4h <=4->NT / 5->2K/XP/2K3
OSMinorVersion DWORD ? ; A8h 0->2K / 1->XP / 2->2K3

D. NT-based definitions

Windows NT, 2000, XP và 2003 sử dụng fixed addresses (địa chỉ cố định) để chứa PEB và TEB. PEB luôn được chứa tại address 7FFDF000h và TEB bắt đầu tại 7FFDE000h. Bởi biết được 2 giái trị cố định này (two fixed values), chúng ta có thể dò tìm ra nền tảng của OS.

III. The Trick

Section II-B đã chỉ ra một cấu trúc add-on structure cho hệ điều hành NT-based . Dử liệu tồn tại trên hệ điều hành non NT-based cũng vậy. Nhưng nó được chứa theo cách khác. Để tính tóan các vị trí memory đúng đắn liên quan đến dò tìm OS – chúng ta dùng một trick phân tích data thêm vào .

Nếu chúng ta nhìn vào NT-TEB-ADD-ON structure chỉ ra trong Section II-B, chúng ta thấy thành phần *LastErrorValue*. Hầu như tất cả Windows API's sẽ return một giá trị error value mà nó có thể truy xuất bằng *GetLastError[1]*. Thêm vào đây ,ta có thể thao tác bằng tay đối với *LastErrorValue* bằng *SetLastError[1]* API. Bằng cách sử dụng hàm API này và hiển thị vùng memory area bên trên TEB,các vị trí của *LastErrorValue* là:

Windows 95 - TEB-base + 60h
Windows 98 - TEB-base + 60h
Windows ME - TEB-base + 74h

Bây giờ chúng ta có thể dò tìm Windows ME or Windows 95/98. Bước đầu tiên chúng ta tính tóan , nhưng ko phải là bước sau cùng . Có thể dò tìm sự khác nhau giữa Windows 95 and Windows 98.

Section II đã chỉ ra giá trị khởi đầu và các rules (quy tắc) của nó. Giá trị khởi đầu của EBX trên OS ko dựa trên NT là 00530000h. Một cách chính xác giá trị này sẽ được tìm thấy bên trong phần dữ liện thêm vào - close *LastErrorValue* đã tính tóan xong ngay bây giờ. Bằng cách phân tích nó ở vị trí này, Kết quả sẽ là:

Windows 95 - TEB-base + 58h
Windows 98 - TEB-base + 54h
Windows ME - TEB-base + 7Ch

Bây giờ chúng ta có thể tham vấn đến mỗi OS ko dựa trên nền NT.

IV. Code creation (cài đặt code)

Đúng lúc này, chúng ta có thể dò tìm thông tin về version cho mỗi OS. Chúng ta muốn một mã độc lập cho OS, chúng ta cần xây dựng các thông tin cho nó . Cũng vậy có thể tìm ra thông tin về version theo luồng độc lập (*workflow independent)*. Section VI sẽ chỉ cách giải quyết vấn đề hòan chỉnh trong Assembler. Trước hết , chúng ta lấy base addresses of PEB và TEB và tính tóan operating system base bằng cách phân tích chúng :

```
assume fs:nothing
mov ebx,fs:[18h] ; get self pointer from TEB
mov eax,fs:[30h] ; get pointer to PEB / database
.if eax==7FFDF000h && ebx==7FFDE000h
; WinNT based
.else
; Win9X based
.endif ; of base check NT/9X
```

Thông tin về version cho NT-based operation systems được chứa trong PEB. Chúng ta chỉ phải phân tích các giá trị *OSMajorVersion* và *OSMinorVersion*:

```
mov ebx,[eax+0A8h] ; get OSMinorVersion
mov eax,[eax+0A4h] ; get OSMajorVersion
.if eax==5 && ebx==0 ; is it Windows 2000?
.elseif eax==5 && ebx==1 ; is it Windows XP?
.elseif eax==5 && ebx==2 ; is it Windows 2003?
.elseif eax<=4 ; is it Windows NT?
.endif
```

Hệ điều hành non NT-based có thể được dò tìm bằng cách phân tích vùng data thêm vào trên TEB, tìm giá trị 00530000h:

```
mov edx,00530000h ; the value to search
mov eax,fs:[18h] ; get the TEB base address
mov ebx,[eax+58h] ; TEB-base + 58h (W95)
mov ecx,[eax+7Ch] ; TEB-base + 7Ch (WME)
mov eax,[eax+54h] ; TEB-base + 54h (W98)
.if ebx==edx ; is it Windows 95?
.elseif eax==edx ; is it Windows 98?
.elseif ecx==edx ; is it Windows ME?
.endif
```

V. Conclusions

Tìm operating system (hệ điều hành) bằng cách dùng kỹ thuật này chỉ để tránh dùng các hàm APIs. Các hàm khác , ví dụ *GetCommandLine[1]*, *IsDebuggerPresent[1]* hay các hàm được có tên trong trong bài này có thể viết lại giống như cách đã trình bày . Lời cuối , reverse engineer (kỷ thuật đảo mã) ko thể set *breakpoints* trên các hàm gọi API, bởi vì chúng ko tồn tại trong chương trình của chúng ta như bạn đã thấy. Và sự pha trộn các OS khác lại với nhau sẽ làm cho quá trình đảo nghịch mã khó khăn hơn.

References

[1] Microsoft Corporation, *Microsoft Developer Network*, http://msdm.microsoft.com
[2] Yuschuk, O., *Olly Debugger*, http://home.t-online.de/home/ollydbg
[3] Hutchenson, S., *MASM V8*, http://www.masmforum.com

VI. Appendix

```
.const
;-- return values from OS_GetOS
OS_UNKNOWN equ -1
OS_WIN95 equ 1
OS_WIN98 equ 2
OS_WINME equ 3
OS_WINNT equ 4
OS_WIN2K equ 5
OS_WINXP equ 6
OS_WIN2K3 equ 7
.code
OS_GetOS proc
local _theReturnValue:DWORD
pushad ; store all registers
mov _theReturnValue,OS_UNKNOWN
assume fs:nothing
mov ebx,fs:[18h] ; get self pointer from TEB
mov eax,fs:[30h] ; get pointer to PEB / database
.if eax==7FFDF000h && ebx==7FFDE000h ; WinNT based
mov ebx,[eax+0A8h] ; get OSMinorVersion
mov eax,[eax+0A4h] ; get OSMajorVersion
.if eax==5 && ebx==0 ; is it Windows 2000?
mov _theReturnValue,OS_WIN2K
.elseif eax==5 && ebx==1 ; is it Windows XP?
mov _theReturnValue,OS_WINXP
.elseif eax==5 && ebx==2 ; is it Windows 2003?
mov _theReturnValue,OS_WIN2K3
.elseif eax<=4 ; is it Windows NT?
mov _theReturnValue,OS_WINNT
.endif
.else ; Win9X based
```

```
mov edx,00530000h ; the magic value to search
mov eax,fs:[18h] ; get the TEB base address
mov ebx,[eax+58h] ; TEB-base + 58h (W95)
mov ecx,[eax+7Ch] ; TEB-base + 7Ch (WME)
mov eax,[eax+54h] ; TEB-base + 54h (W98)
.if ebx==edx ; is it Windows 95?
mov _theReturnValue,OS_WIN95
.elseif eax==edx ; is it Windows 98?
mov _theReturnValue,OS_WIN98
.elseif ecx==edx ; is it Windows ME?
mov _theReturnValue,OS_WINME
.endif
.endif ; of base check NT/9X
popad ; restore all registers
mov eax,_theReturnValue
ret ; return to caller
OS_GetOS endp
```

http://masm32vn.com (thanz NTA)
http://h1.ripway.com/benina/
http://benina.250free.com
Benina 07/02/2006
Update 07/02/2006
Mail: benina@walla.com