DLL Injection & Hooking

ترجمه و گرد آوری :آرش تابع
Arash.tabe@gmail.com

فهرست

مقدمه

در این مقاله شما در مورد DLL Injection و نحوه Hooking و کنترل آن از راه دور مطالبی را یاد خواهید گرفت .

در ویندوز هر فرایندی دارای فضای آدرس مجازی می باشد که می تواند در هر زمان DLL دلخواهی را در آن بارگذاری و یا خالی کند.توضیح اینکه تخلیه و بارگذاری DLL توسط خود فرایند آغاز و صورت می گیرد. گاهی اوقات ما می خواهیم یک DLL را داخل فرایندی بدون اینکه در مورد آن فرایند اطلاعاتی داشته باشیم بارگذاری کنیم . دلایل بسیاری برای این کار وجود دارد . بعنوان مثال نویسنده یک بدافزار ممکن است بخواهد برای مخفی کردن فعالیتهای فایل های مخرب خود با استفاده از بارگذاری یک DLL به فرایند خود ، آن فرایند را معتبر سازد و یا ممکن است بخواهد برای دور زدن دستگاههای امنیتی ار آن استفاده کند.در حالیکه از سوی دیگر ممکن است یک برنامه نویس برای ارتقا و یا گسترش قابلیتهای برنامه خود از DLL استفاده می کند. اما برای هردو فعالیت این مراحل یکسان است.

در این مقاله ما با راههای مختلف و بصورت عملی کنترل از راه دور کد و یا DLL خود را در یک فرایند تزریق کرده و سپس آنرا گسترش خواهیم داد تا تابع API در درون فرایند وظایف محوله ما را به نحو احسنت انجام دهد.

تزریق DLL

اگر اشتباه نکنم حدود ۴۵ الی ۵۰٪ بدافزارها ار این روش برای تزریق کد برای اجرای فعالیت های مخرب استفاده می کنند .پس درک مفهوم تزریق DLL برای یک تحلیل گر نرم افزار های مخرب بسیار حیاتی و مهم است.

نکته :

- این روش ها با استفاده از زبان برنامه نویسی اسمبلی انجام و نوشته خواهد شد.
- این روش ها در نسخه های بالای vista یعنی 7, 8 کاربرد ندارد.

مدلهایی از روشهای تزریق DLL

1- Window hooks (SetWindowsHookEX)
2- CreateRemoteThread
3- App_Init registry key
4- ZwCreateThread or NtCreateThreadEx Global method (works well on all versions of windows)
5- Via APC (Asynchronous procedure calls)

در این مقاله ما از یک روش منطقی و ساده بنام CreateRemoteThread برای این کار استفاده می کنیم (۱) روش CreateRemoteThread از نسخه ویندوز ویستا به بعد با توجه به جدایی و جداسازی جلسه ها کار نمی کند(۴) در این صورت شما می توانید از عملکرد مشابه اما ثبت نشده NtCreateThread استفاده کنید، (۲) در واقع این مشکل CreateRemoteThread نیست بلکه Ntdll از CsrClientCallServer بصورت درست بر نمی گردد. اگر این برگشت موفقیت آمیز بود ما براحتی می توانستیم DLL را در هر فرایندی با استفاده از CreateRemoteThread تزریق کنیم.

- در اینجا ما در CreateRemoteThread بر روی ویندوز XPتمرکز می کنیم.

تزریق DLL با استفاده از CreateRemoteThread :

در حالت اول دو موقعیت وجود دارد:

۱- تزریق DLL به یک فرایند در حال اجرا
۲- ایجاد یک فرآیند و تزریق DLL به آن

بنابراین CreateRemoteThread به این معنی است که می توانید یک موضوع را با یکی دیگر از فرایند ایجاد و یا می توان گفت که می توان یک تابع را با یکی دیگر از فرایندها اجرا کرد.

به این syntax نگاه کنید:

```
HANDLE WINAPI CreateRemoteThread(
  __in   HANDLE hProcess,                                          ?--
------ 1
  __in   LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in   SIZE_T dwStackSize,
  __in   LPTHREAD_START_ROUTINE lpStartAddress, ?---------2
  __in   LPVOID lpParameter,                                     ?---
------3
  __in   DWORD dwCreationFlags,                                ?---------
4
  __out  LPDWORD lpThreadId
);
```

پارامترهای ذکر شده برای کار ما حیاتی هستند چونکه :

۱- رسیدگی به فرآیندی که نخ ها در آن ایجاد می شوند مهم است.

۲- اشاره به تابعی که در فرآیندی وارد نخ شده و باعث اجرای آن می شود.

۳- پارامتر به تابع

٤- حالت ایجاد یک نخ

همه ما می دانیم که kernerl32.dll کتابخانه های API را در زمان اجرا لود و بارگذاری می کند همچنین kernerl32.dll هر فرآیندی را بصورت پیش فرض بارگذاری می کند . بنابراین ما می توانیم آدرس LoadLibrary را به 2# و پارامتر به LoadLibrary را به 3 #منتقل کنیم . هنگامی که آرگومان ما پس از دستور CreateRemoteThread خواهد بود LoadLibrary با پارامتر خود را در یکی دیگر از فرایندها اجرا و از این رو لود DLL در فرایند دیگری صورت خواهد گرفت.

تنها مشکل این روش این است که پارامتر LoadLibrary باید در فرایند هدف باشد .به عنوان مثال اگر ما از LoadLibrary (2 #) با (3 #) "mydll.dll"به عنوان پارامتر Loadlibrary استفاده می کنیم "mydll.dll"باید در فرآیند مورد نظر ما باشد.

خوشبختانه ویندوز ، API را برای این کار آماده کرده است. ما می توانیم با استفاده از WriteProcessMemory به هر فرآیندی ارسال و می توانیم فضا را به هر فرایندی با استفاده از

VirtualAllocEx API تخصیص بدهیم.اما قبل از این کارها ما نیاز داریم تا به فعالیت فرایندها رسیدگی کنیم این کار را می توانیم با استفاده از OpenProcess یا CreateProcess API انجام دهیم.

این کار بدین صورت خواهد شد :

۱- استفاده از OpenProcessیا CreateProcess API برای دریافت فرایندهای سیستم قربانی .
۲- استفاده از VirtualAllocEx برای اختصاص فضا برای فرایندهای سیستم قربانی.
۳- استفاده از WriteProcessMemory برای ارسال نام DLL به فرایند سیستم قربانی.
٤- استفاده از CreateRemoteThread برای تزریق DLL به فرایند سیستم قربانی .

مراحل بالا به اندازه کافی برای تزریق DLL به یک فرایند می باشد .اگر چه برای تزریق به یک فرایند سیستم ما برای اولین بار باید به مجموعه امتیاز se_debug در فرایند برسیم .(به معنی روندی است که DLL را به یکی دیگر از فرایندها تزریق می کنیم)

اگر شما به یاد داشته باشید "دو موقعیت " در آغاز این بخش داشتیم ما نیاز به یک کمی کار بیشتر برای 2 # یعنی ایجاد یک فرآیند و تزریق DLL را در آن داریم.

ما اول باید یک فرایند ایجاد کنیم و پس از آن برای تزریق DLL از این فرایند استفاده کنیم.

به این syntax نگاه کنید

```
BOOL WINAPI CreateProcess(
  __in_opt     LPCTSTR lpApplicationName,
  __inout_opt  LPTSTR lpCommandLine,
  __in_opt     LPSECURITY_ATTRIBUTES lpProcessAttributes,
  __in_opt     LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in         BOOL bInheritHandles,
  __in         DWORD dwCreationFlags,              ?-------- 1
  __in_opt     LPVOID lpEnvironment,
  __in_opt     LPCTSTR lpCurrentDirectory,
  __in         LPSTARTUPINFO lpStartupInfo,
  __out        LPPROCESS_INFORMATION lpProcessInformation
);
```

در اینجا dwCreationFlags یک پارامتر مهم می باشد.اگر به معنای MSDN نگاه کنید خواهید دید که از آن برای کنترل ایجاد یک فرآیند استفاده شده است . ما می توانیم با استفاده از مجموعه "CREATE_SUSPENDED" فرایندی را ایجاد کنیم و آنرا به حالت تعلیق در آوریم.

با استفاده از پرچم CREATE_SUSPENDED CreateProcessمی توانیم اجرای فرایند اصلی را در نقطه ورود متوقف کنیم برای شروع فرایند ما می توانیم از ResumeThread API استفاده کنیم.

روش کار بدین صورت خواهد بود :

۱- ایجاد فرآیند در حالت تعلیق
۲- تزریق DLL به فرآیند با استفاده از مراحل بالا
۳- سوابق فرایند

در اینجا برنامه کامل است و شبیه مراحل بالا می باشد.

```
.386
.model flat, stdcall
option casemap:none

include windows.inc
include msvcrt.inc
include kernel32.inc

includelib kernel32.lib
includelib msvcrt.lib

.data
greet   db      "enter file name: ",0
sgreet  db      "%s",0
dreet   db      "enter DLL name: ",0
dgreet  db      "%s",0
apiname db      "LoadLibraryA",0
dllname db      "kernel32.dll",0

.data?
processinfo     PROCESS_INFORMATION <>
startupinfo     STARTUPINFO <>
fname   db      20      dup(?)
dname   db      20      dup(?)
dllLen  dd      ?
mAddr   dd      ?
vpointer        dd      ?
lpAddr  dd      ?


.code
start:

invoke crt_printf,addr greet
invoke crt_scanf,addr sgreet,addr fname
invoke crt_printf,addr dreet
invoke crt_scanf,addr dgreet,addr dname
invoke LoadLibrary, addr dllname
ov mAddr,eax
```

```
invoke GetProcAddress,mAddr,addr apiname
mov lpAddr,eax

;create process in suspended state
invoke CreateProcess,addr fname,0,0,0,0,CREATE_SUSPENDED,0,0,addr
startupinfo,addr processinfo
invoke crt_strlen,addr dname
mov dllLen,eax

; Allocate the space into the newly created process
invoke
VirtualAllocEx,processinfo.hProcess,NULL,dllLen,MEM_COMMIT,PAGE_EXECUTE_REA
DWRITE
mov vpointer,eax

; Write DLL name into the allocated space
invoke WriteProcessMemory,processinfo.hProcess,vpointer,addr
dname,dllLen,NULL

; Execute the LoadLibrary function using CreateRemoteThread into the
previously created process
invoke
CreateRemoteThread,processinfo.hProcess,NULL,0,lpAddr,vpointer,0,NULL
invoke Sleep,1000d

; Finally resume the process main thread.
invoke ResumeThread,processinfo.hThread
xor eax,eax
invoke ExitProcess,eax

end start
```

برنامه کنسول را در WinAsm انتخاب کنید و کد بالا را در آن تجمیع کنید . این کد باید یک فرایند ایجاد و DLL را درآن تزریق کند.

برای مثال :

شما می توانید فرایند calc.exe را ایجاد و urlmon.dll را در آن تزریق کنید ، به طور پیش فرض calc.exeدر urlmon.dll لود نمی شود.

Hooking :

در برنامه نویسی کامپیوتر،Hook (اتصال) ، مدت را پوشش می دهد که طیف وسیعی از تکنیک های مورد استفاده برای تغییر یا تقویت رفتار یک سیستم عامل، برنامه های کاربردی، و یا از دیگر اجزای نرم افزار با متوقف کردن فراخوانی های توابع و یا پیام و یا رویدادهای گذشته بین اجزای نرم افزار ها می باشد. که کد دسته مانند فراخوانی تابع ، رویدادها و یا پیام ها به نام Hook می باشد.

Hooking یک روش قدرتمند موجود در نرم افزار های کامپیوتری است .که یک فرد می تواند تقریبا همه چیز را در یک سیستم با استفاده از hook در محل مناسب انجام دهد.

همانطور که در تعریف آمده است که Hooking فراخوانی تابع یا پیام و یا حوادث را رهگیری می کند.دلیل این است که با در نظر گرفتن نحوه جریان اجرای یک فایل اجرایی می توانیم با این کار در مکان های مختلف از فایل اصلی با سیستم ارتباط برقرار کنیم.

در ابتدا Hook را می توان به دو دسته تقسیم کرد.

۱- حالت کاربر

- IAT جدول آدرس های وارد شده
- Inline Hooking
- فراخوانی Patching در باینری ها و ...

۲- حالت هسته

- اتصال IDT
- اتصال SSDT و ...

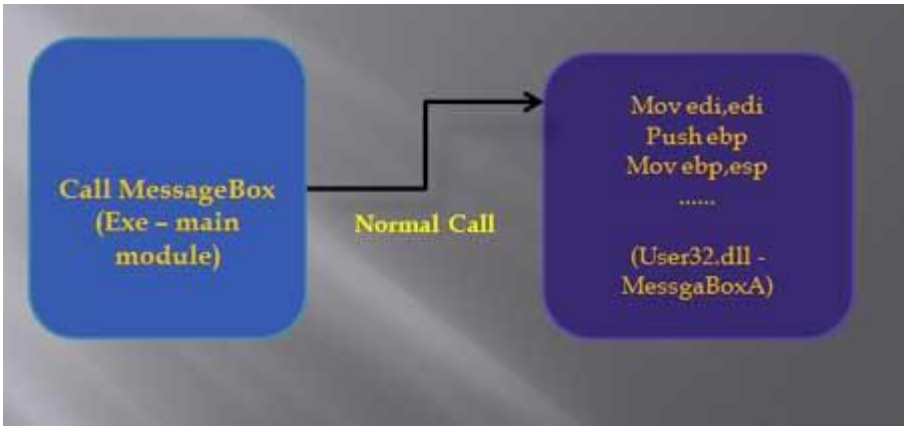
در این مقاله ما در مورد روش Inline hooking که یکی از روش های موثر Hooking است صحبت خواهیم کرد.

Inline hooking

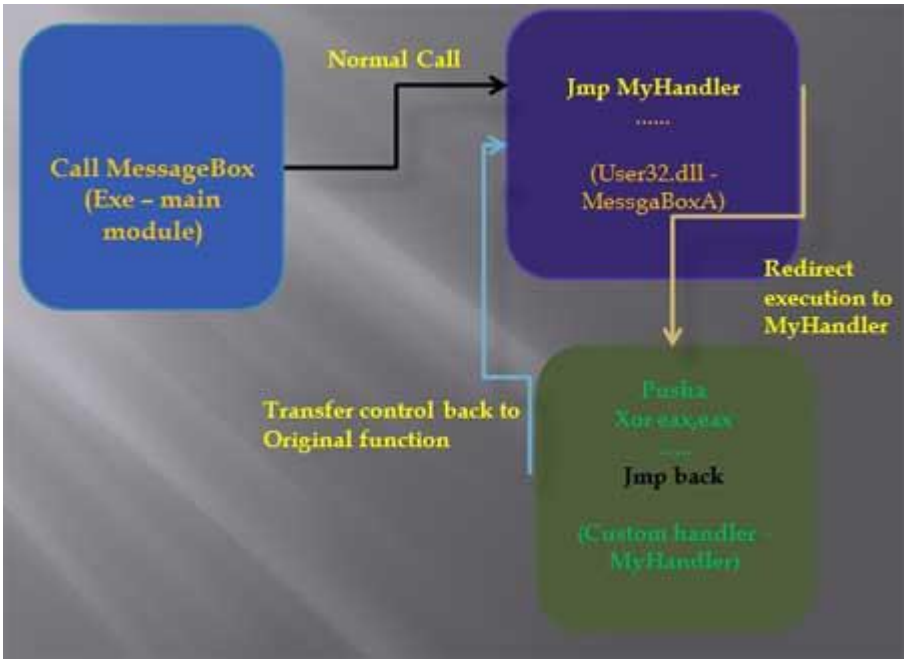
در Inline hooking ما ۵ بایت اول تابع یا API را که برای تغییر اجرای مسیر فایل اجرایی ما است را بازنویسی می کنیم . این 5بایت می تواند JMP، PUSH RETو دستور CALL باشد.

بصورت تصویری در زیر توضیح می دهیم:

تصویر ۱ : فراخوانی عادی (بدون hooking)



تصویر ۲ : فراخوانی بعد از hooking



همانطور که در تصویر بالا می بینیم توابع MessageBox با شروع کلمه با JMP درون تابع MyHandler رونویسی (overwritten) می شود. در تابع MyHandler ما مسائلی را انجام داده و سپس آنرا برای کنترل MessageBox ارسال می کنیم.

در این قسمت ما یک DLL می سازیم که در MessageBox API پیام ما را بجای پیام واقعی نمایش دهد.

برای ایجاد یک DLL ما به موارد زیر نیاز داریم :

- آدرس های اشاره گر MessageBox API
- تابع با کد آدرس اشاره گر

نکته :ما می توانیم آدرس MessageBoxA API با استفاده از GetProcAddress دریافت کنیم

در اینجا مراحل عبارتند از:

۱- دریافت آدرس MessageBoxA
۲- دریافت کد سفارشی و یا آدرس تابع
۳- بازنویسی کلمه در ادامه متن در 1 # با JMP به 2 #
۴- تغییر پارامتر از فراخوان
۵- انتقال کنترل به 1 #

در اینجا کد کامل deomonstrating تابع های Hooking MessageBox است.

```
.386
.model flat,stdcall
option casemap:none

include windows.inc
include kernel32.inc
include msvcrt.inc
include user32.inc


includelib kernel32.lib
includelib msvcrt.lib
includelib user32.lib


.data
```

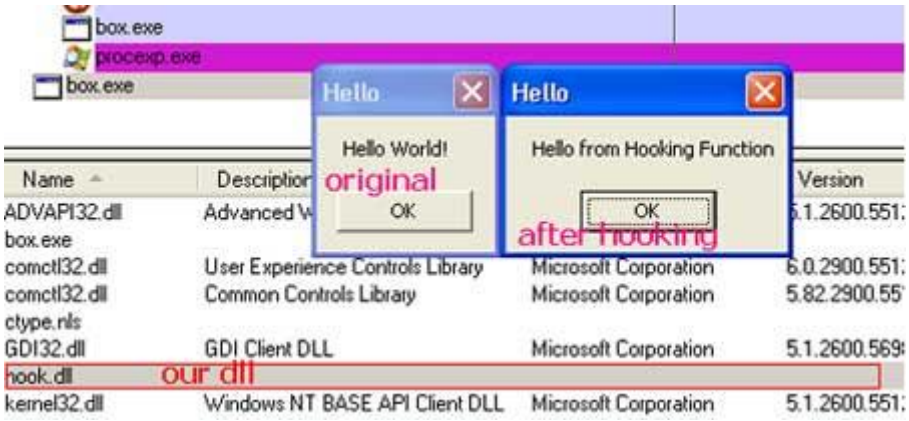
```
tszMsg          db      "Hello from Hooking Function",0
userDll         db      "user32.dll",0
msgapi          db      "MessageBoxA",0


.data?
oByte1  dd      ?
oByte2  dd      ?
userAddr        dd      ?
msgAddr dd      ?
nOldProt        dd      ?

.code
LibMain proc hInstDLL:DWORD, reason:DWORD, unused:DWORD
 .if reason == DLL_PROCESS_ATTACH
        invoke LoadLibrary,addr userDll
        mov userAddr,eax

        ; Get MessageBoxA address from user32.dll
        invoke GetProcAddress,userAddr,addr msgapi
        mov msgAddr, eax

    ; Set permission to write at the MessageBoxA address
        invoke VirtualProtect,msgAddr,20d,PAGE_EXECUTE_READWRITE,OFFSET
nOldProt

    ; Store first 8 byte from the MessageBoxA address
        mov eax,msgAddr
        mov ebx, dword ptr DS:[eax]
        mov oByte1,ebx
        mov ebx, dword ptr DS:[eax+4]
        mov oByte2,ebx

        patchlmessagebox:
                ; Write JMP MyHandler (pointer) at MessageBoxA address
                mov byte ptr DS:[eax],0E9h
                ; move MyHandler address into ecx
                mov ecx,MyHandler
                add eax,5
                sub ecx,eax
                sub eax,4
                mov dword ptr ds:[eax],ecx

    .elseif reason == DLL_PROCESS_DETACH
    .elseif reason == DLL_THREAD_ATTACH
    .elseif reason == DLL_THREAD_DETACH
    .endif
    ret
LibMain endp


MyHandler proc
                pusha
                xor eax,eax
                mov eax,msgAddr

        ; change the lpText parameter to MessageBoxA with our text
                mov dword ptr ss:[esp+028h],offset tszMsg

        ; Restore the bytes at MessageBoxA address
```

```
                mov ebx,oByte1
                mov dword ptr ds:[eax],ebx
                mov ebx,oByte2
                mov dword ptr ds:[eax+4],ebx

        ; Restore all registers
                popa

        ;jump to MessageBoxA address (Transfer control back to MessageBoxA)
                jmp msgAddr
MyHandler endp

end LibMain
```

در پروژه جدید DLL های استاندارد را در Win Sam انتخاب کنید و کد بالا را در جایی که ویرایش می کنید اضافه کنید.

اکنون ما DLL هایی داریم که می توانند به MessageBoxA اتصال پیدا کند و پارامترها و نوشته های مارا با پارمترها و نوشته های اصلی عوض کنند.ما با این برنامه می توانیم DLL را به برنامه های متفاوت تزریق کنیم.

به عنوان مثال در تصویر زیر ما عبارت " hello world "را با عبارت"hello from hokking " function را در خروجی برای ما نشان می دهد.



نتیجه :

هر دو روش تزریق DLL و Hooking تکنیک های قدرتمند و محبوب استفاده شده توسط برنامه نویسان برنامه های مخرب و همچنین برنامه نویسان نرم افزار معتبر می باشد.

مراجع :

1- http://www.codeproject.com/Articles/4610/Three-Ways-to-Inject-Your-Code-into-Another-Proces
2- http://securityxploded.com/ntcreatethreadex.php
3- http://research.microsoft.com/en-us/projects/detours/
4- http://research.microsoft.com/en-us/projects/detours/