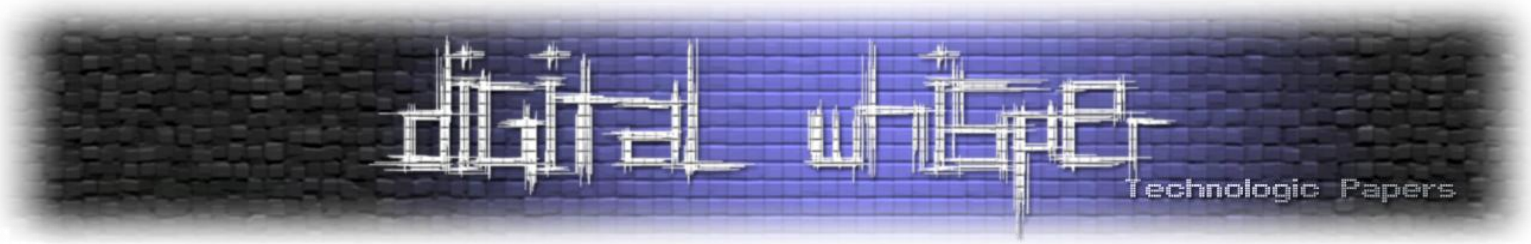


User-Land Hooking

מאת אורי (Zerith)

הקדמה

לאחרונה נתקלתי בשיחה אשר התרחשה בערוץ IRC בין שניים מחברי הקהילה, אותם החבר'ה שוחחו ביניהם על הדרך הטובה ביותר לביטול ה-Nag (שהוא חלון ה-"Evaluation") של Mirc. אחד מהם הציע שימוש ב-Hook לפונקציה של יצירת החלון.

עד כאן הכל טוב, פתרון לגיטימי, אך לאחר מעט זמן ובלבול גדול התברר שהבחור שמימש את ה-Hook כנראה לא הבין כל כך מה זה Hook, ורק הכיר את השימושים שלו. מה שהוא התכוון לממש זה בכלל קריאה לפונקציה SetWindowsHookEx!

שימוש בפונקציה זו לא תהווה פתרון למצב ואי אפשר בכלל להשתמש בה עבור Hook-ים מקומיים לתהליך (כפי שנדגים בהמשך.).

במאמר זה ארצה להבהיר את התפיסה השגוייה בעניין ה-Hooking ואף אסביר על שימוש ומטרת ה-Hooking, סוגי ה-Hooks השונים ומימושם. בנוסף, אדגים שימוש של Hooking על מנת לעקוף את ה-Nag של mIRC ☺

מהו Hook?

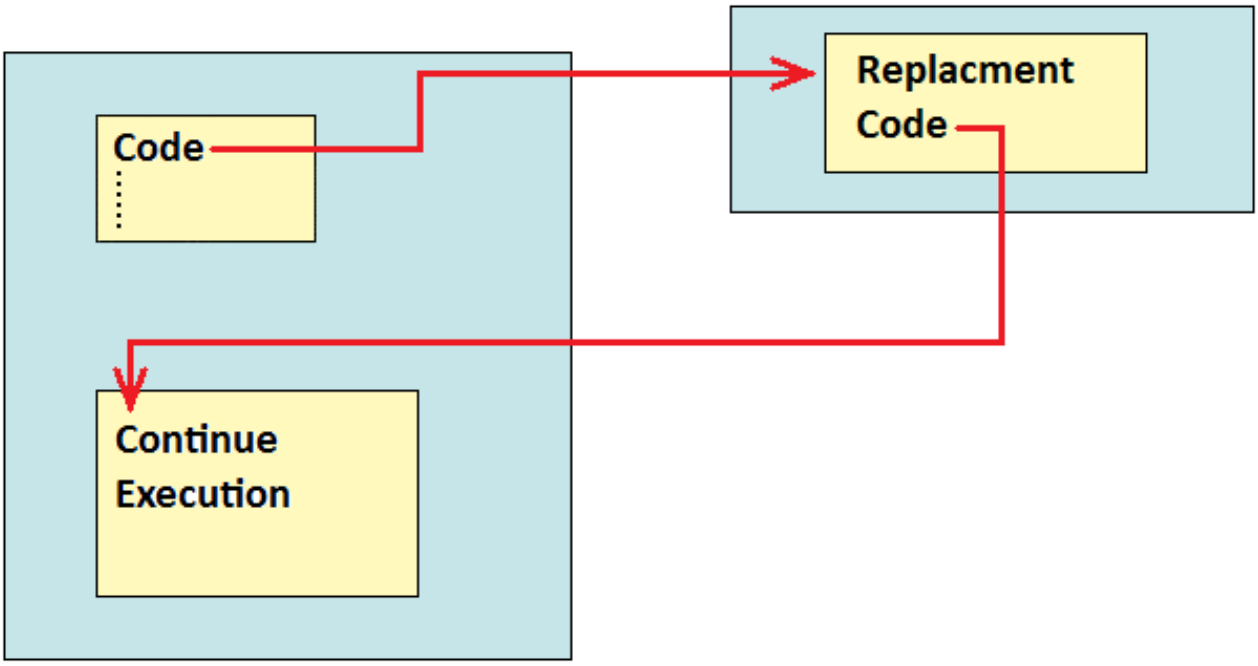
לפי וויקיפדיה:

> "**Hooking** היא שיטה המשמשת לשינוי ההתנהגות של מערכת ההפעלה, תוכנה או קוד מכונה באמצעות יירוט קריאה לשגרה או הודעה המועברת בין תוכניות. קטע הקוד האחראי ליירוט נקרא **Hook**."

הכוונה היא, ש-Hook הוא בעצם יירוט של הקוד שאמור היה לרוץ- לקוד שלנו. משמעות הדבר היא, שאנחנו בשליטה על מהלך הריצה- אם נרצה נוכל לקרוא לקוד המקורי ואם נרצה נוכל לשנות לגמרי את מסלול הריצה, הכל בידיים שלנו.

(כשמתכוונים ל-Hook, הכוונה היא לשינוי של קוד **התהליך** בזמן ריצה, ולא שינוי קוד הקובץ).

שימו לב לתרשים הבא:



ה-Hook הוא שינוי קטע קטן ב-code, אשר יקפוץ ל-Replacement code שיבצע מה שהוא רוצה. ברוב המקרים הוא גם חוזר לקוד המקורי (Continue execution) – אבל זה לא הכרחי.

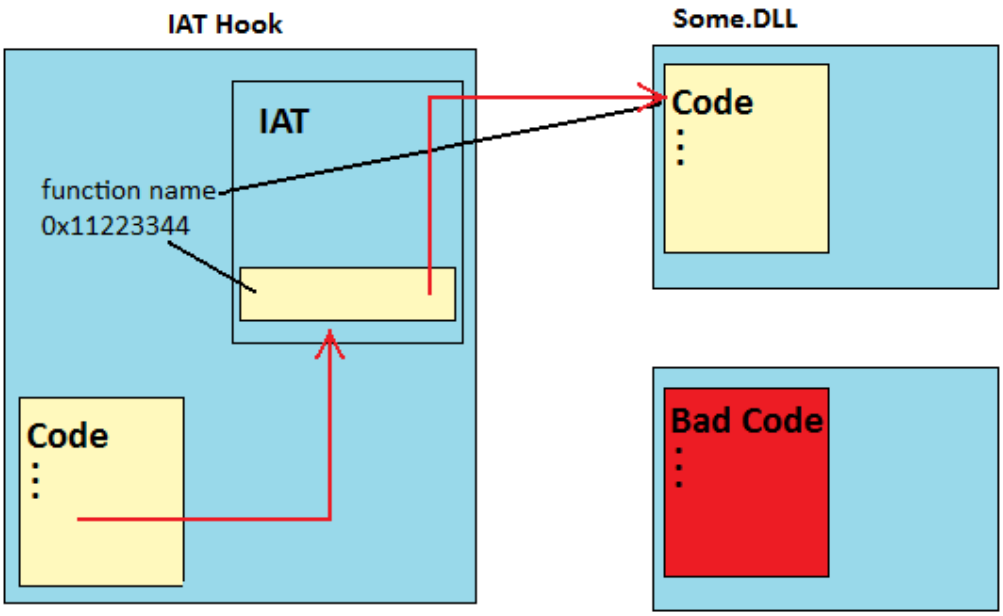
הרבה תוכנות משתמשות ב-Hooking, ולא רק לצרכים זדוניים, למשל, ישנן תוכנות אשר משתמשות ב-Hooking על מנת להרחיב את הפונקציונליות של קוד או לשנותו לשם תאימות המערכת.
וכמובן – גם בצד הזדוני, כמו שהדגמתי במאמרים על Rootkits.

IAT Hooking

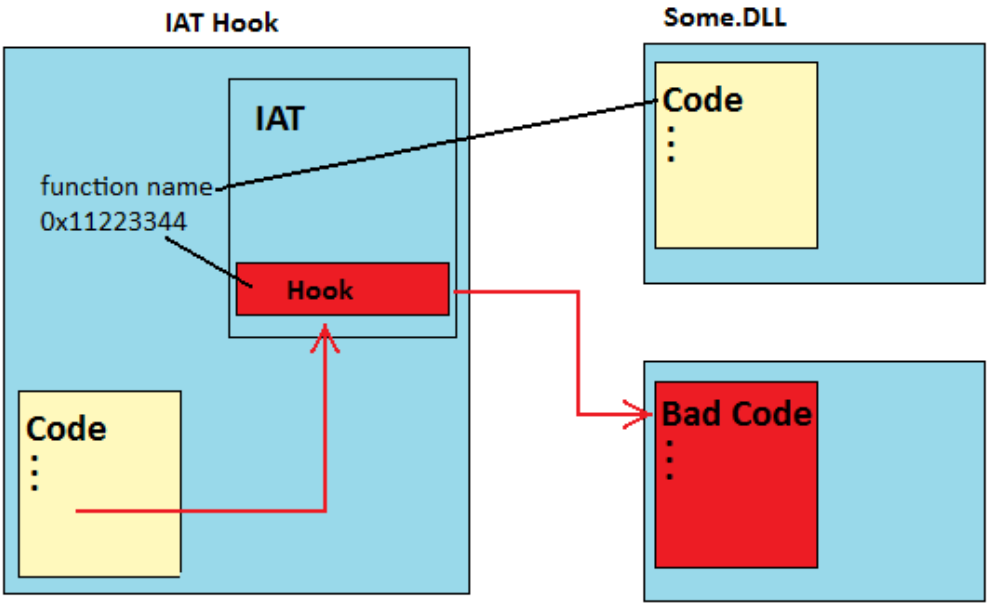
(אם אינך יודע מהי ה-IAT , תוכל לקרוא עליה במאמר שלי על Manual Unpacking שפורסם בגליון השני של Digital Whisper)

IAT Hooking היא הפעולה של שינוי רישום של פונקציה ב-IAT . (ניתן למצוא את ה-IAT על ידי שימוש בהיסט המתאים לתוך ה-PE header).

הנה תרשים של פעולת קריאה לפונקציה דרך ה-IAT לפני ביצוע ה-Hook:



לאחר ביצוע ה-Hook, הכתובת (0x11223344 במקרה הזה) של הפונקציה ב-IAT תשונה לכתובת של BAD CODE:



לכן זרימת הקוד תעבור ל-Bad Code במקום לפונקציה ב-DLL.

זיהוי IAT Hooking הוא פשוט למדי, ברב המקרים מספיק לעשות בדיקת תקינות לכתובת הרשומה ב-IAT, אם היא נמצאת מחוץ לטווח הכתובות של ה-DLL המתאים, כנראה שהתבצע IAT Hook.

ניתן לראות דוגמא של IAT Hooking בכתובת הבאה:
http://jpassing.com/2008/01/06/using-import-address-table-hooking-for-testing

Inline Hooking

Inline Hooking הוא הסוג הנפוץ ביותר של Hooks בסביבת ה-User-Land כך שבדרך כלל, כאשר אומרים Hook, מתכוונים לסוג הזה.

Function Prologue

ברוב הפונקציות ה-API של Windows ישנו פרולוג (פרולוג- מיוונית, "פתיחה"), שהוא בעצם חמשת הבתים הראשונית של הפונקציה, שאחראים על יצירת ה- Stack frame של הפונקציה.

(http://en.wikibooks.org/wiki/X86_Disassembly/Functions_and_Stack_Frames)

כפי שניתן לראות מהפונקציה CreateFileA:

```
7C801A24  $  8BFF          MOV EDI,EDI
7C801A26  .  55            PUSH EBP
7C801A27  .  8BEC          MOV EBP,ESP
7C801A29  .  FF75 08       PUSH DWORD PTR SS:[EBP+8]
7C801A2C  .  E8 53C60000   CALL kernel32.7C80E084
7C801A31  .  85C0          TEST EAX,EAX
7C801A33  .v 74 1E         JE SHORT kernel32.7C801A53
7C801A35  .  FF75 20       PUSH DWORD PTR SS:[EBP+20]      hTemplateFile
7C801A38  .  FF75 1C       PUSH DWORD PTR SS:[EBP+1C]      Attributes
7C801A3B  .  FF75 18       PUSH DWORD PTR SS:[EBP+18]      Mode
7C801A3E  .  FF75 14       PUSH DWORD PTR SS:[EBP+14]      pSecurity
7C801A41  .  FF75 10       PUSH DWORD PTR SS:[EBP+10]      ShareMode
7C801A44  .  FF75 0C       PUSH DWORD PTR SS:[EBP+C]       Access
7C801A47  .  FF70 04       PUSH DWORD PTR DS:[EAX+4]       FileName
7C801A4A  .  E8 21ED0000   CALL kernel32.CreateFileW       CreateFileW
7C801A4F  >  5D            POP EBP
7C801A50  .  C2 1C00       RETN 1C
7C801A53  >  83C8 FF       OR EAX,FFFFFFFF
7C801A56  .^ EB F7         JMP SHORT kernel32.7C801A4F
```

(בטח כבר שמתם לב שאני אובסיבי לגבי הפונקציה CreateFileA ;))

הדבר המצחיק קיים בהוראה הבאה:

```
MOV EDI, EDI
```

אין בה צורך, אך היא בכל זאת משלימה ל-5 בתים, כך שהיא מושלמת לצורך ה-Inline Hooking שלנו (הוראת JMP היא בדיוק 5 בתים), ממש כאילו מיקרוסופט ידעו על זה מראש.

Inline Hooking הם בדיוק זה – שינוי הפרולוג של הפונקציה לקפיצה לקוד שלנו. לאחר ביצוע ה-Hook, השליטה ברשותינו לשנות את הפרמטרים, לשנות את הערך החוזר, וכן הלאה.

איך ניתן לעשות זאת?

DLL Injection

DLL Injection הינה טכניקה בה אדם "מזריק" קובץ DLL לתוך תהליך קיים וגורם להרצת קוד מתוכו. תהליך הזרקת ה-DLL הוא כזה:

1. יצירת DLL אשר יבצע את ה-Hook כשהוא בתוך מרחב הכתובות של התהליך, ה-DLL הזה הוא המוזרק.
2. יצירת תכנית (EXE) אשר תזריק את ה-DLL, על מנת להזריק אותו עליו לעשות את השלבים הבאים.

התכנית שמזריקה את ה-DLL מבצעת זאת על ידי יצירת Thread חיצוני, שיורץ בתהליך הרצוי על ידי CreateRemoteThread. ה-Thread שנוצר יריץ קריאה ל-LoadLibraryA אשר תטען את ה-DLL שברצוננו להזריק.

התכנית מבצעת זאת בשלבים הבאים:

1. הקצאת זיכרון חיצונית שתספיק להכיל את הפרמטרים לפונקציה LoadLibraryA על ידי קריאה לפונקציה VirtualAllocEx().
2. שינוי הגנות הדף על הזיכרון שהוקצה על מנת שיהיה ניתן לכתוב אליו.
3. השגת ה-Path ל-DLL שברצוננו להזריק.
4. כתיבת ה-Path לזיכרון שהוקצאה על ידי WriteProcessMemory().
5. השגת הכתובת של LoadLibraryA במערכת ההפעלה הספציפית על ידי קריאה ל-GetProcAddress().

6. יצירת Thread חיצוני על ידי CreateRemoteThread, אליו מועבר הפרמטר Path (שכתבנו לתהליך קודם לכן) , שיריץ קריאה לLoadLibraryA.

תפקיד ה-DLL בתהליך הוא כתיבת ההוראה JMP + ההיסט המתאים לקפיצה לפונקציה (שלכם) שתחליף את הפרולוג (אציג דוגמא בהמשך).

זיהוי Inline Hooking הוא גם לא מסובך מדי, פשוט בודקים אם הפרולוג של הפונקציה לא שונה.

SetWindowsHookEx

SetWindowsHookEx הוא אמצעי הניתן על ידי Windows, לביצוע Hook גלובאלי על המערכת. זאת אומרת שה-Hook יבוצע על כל התהליכים הרצים במערכת, והוא לא לוקאלי לתהליך ספציפי, לכן לא ניתן להשתמש בו לפיתרון בעיית ה-Nag ב-mIRC.

SetWindowsHookEx משומש כדי לזהות ולקבל כל מיני אירועים במערכת, כגון שליחת הודעת חלון, מקלדת ועכבר, לפני שהתהליכים האחרים מקבלים אותם. כמו כן, למשתמש ניתנת שליטה על המידע שיועבר להתהליכים ואפשרות לשנות את תוכן האירוע. עם זאת, המשתמש מוגבל לאירועים שמיקרוסופט הגדירו.

כאשר משתמש מבצע קריאה ל-SetWindowsHookEx, מערכת ההפעלה Windows מכניסה את ה-Hook שלו לשרשרת Hook–ים, שנקראת בכל פעם שקורה האירוע הספציפי אשר אליו בוצע ה-Hook, בסוף ה-Hook, על המשתמש לקרוא ל-Hook הבא בשרשרת על ידי קריאה ל-()CallNextHookEx.

ניתן לראות דוגמא של השימוש בפונקציה בכתובת הבאה:

http://www.codeproject.com/kb/dll/keyboardhook.aspx

דוגמה מצחיקה נוספת שביצעתי בעבר ע"י שימוש ב-Hook הגלובאלי, היא ביצוע Hook למקלדת ושינוי כל דבר שהמשתמש כותב למחרוזת "אני טמבל". התכנית שביצעה את ה-Hook הגלובאלי כתבה ב-Console: "Hooked Successfully" – כך כששלחתי את התכנית לחברים, הם הריצו את התכנית וכל מה שהם ראו זה את הפלט. מיותר לציין שכשהם רצו לרשום "התכנית לא עושה כלום", הם כתבו "אני טמבל". ☺

מימוש Hooking

לאחר שהצלחנו להסביר מה זה Hooking, הבה נראה איך מבצעים את זה. נחזור לתרחיש הקודם: תקרית ה-Nag של mIRC . הפונקציה המשמשת ליצירת חלון ה-Nag היא DialogBoxParamA. לאחר חקירה קצרה, הקריאה ל-DialogBoxMessageA מתבצעת בשגרה הבאה:

```
004762CA  .  56              PUSH ESI
004762CB  .  8B7424 10       MOV ESI,DWORD PTR SS:[ESP+10]
004762CF  .  57              PUSH EDI
004762D0  .  8B7C24 18       MOV EDI,DWORD PTR SS:[ESP+18]
004762D4  .  83FF 0A         CMP EDI,0A
004762D7  .  8935 A8D9630(   MOV DWORD PTR DS:[63D9A8],ESI
004762DD  .v 74 1C           JE SHORT mirc.004762FB
004762DF  .  A1 A86E6700     MOV EAX,DWORD PTR DS:[676EA8]
004762E4  .  85C0            TEST EAX,EAX
004762E6  .v 74 13           JE SHORT mirc.004762FB
004762E8  .  53              PUSH EBX                          lParam
004762E9  .  55              PUSH EBP                          DlgProc
004762EA  .  0FB7CF          MOVZX ECX,DI
004762ED  .  56              PUSH ESI                          hOwner
004762EE  .  51              PUSH ECX                          pTemplate
004762EF  .  50              PUSH EAX                          hInst => NULL
004762F0  .  FF15 28475F0(   CALL DWORD PTR DS:[<&USER32.DialogBoxPa:  DialogBoxParamA
004762F6  .  83F8 FF         CMP EAX,-1
004762F9  .v 75 26           JNZ SHORT mirc.00476321
004762FB  > A1 3C0F6A00      MOV EAX,DWORD PTR DS:[6A0F3C]
00476300  .  53              PUSH EBX                          lParam
00476301  .  55              PUSH EBP                          DlgProc
00476302  .  0FB7D7          MOVZX EDX,DI
00476305  .  56              PUSH ESI                          hOwner
00476306  .  52              PUSH EDX                          pTemplate
00476307  .  50              PUSH EAX                          hInst => NULL
00476308  .  FF15 28475F0(   CALL DWORD PTR DS:[<&USER32.DialogBoxPa:  DialogBoxParamA
0047630E  .  8BF0            MOV ESI,EAX
00476310  .  83FE FF         CMP ESI,-1
00476313  .v 75 0A           JNZ SHORT mirc.0047631F
00476315  .  6A 00           PUSH 0
00476317  .  E8 A4BAFAFF     CALL mirc.00421DC0
0047631C  .  83C4 04         ADD ESP,4
0047631F  > 8BC6             MOV EAX,ESI
00476321  > 5F               POP EDI
00476322  .  5E              POP ESI
00476323  .  5D              POP EBP
00476324  .  C705 A8D9630(   MOV DWORD PTR DS:[63D9A8],0
0047632E  .  5B              POP EBX
0047632F  .  C3              RETN
```

כאן מתבצעת הקריאה לשגרה המדוברת, ולאחר מכן השוואה של הערך החוזר ל-0x7B0:

```
00501E53  .  56              PUSH ESI
00501E54  .  68 10184000     PUSH mirc.00401810
00501E59  .  6A 0A           PUSH 0A
00501E5B  .  51              PUSH ECX
00501E5C  .  E8 5F44F7FF     CALL mirc.004762C0   <---
00501E61  .  83C4 10         ADD ESP,10
00501E64  .  3D B0070000     CMP EAX,7B0
00501E69  .v 74 20           JE SHORT mirc.00501E8B
00501E6B  .  891D 7C6E670(   MOV DWORD PTR DS:[676E7C],EBX
00501E71  .  E8 EACAFFFF     CALL mirc.004FE960
00501E76  .  5E              POP ESI
00501E77  .  8BC3            MOV EAX,EBX
00501E79  .  C705 C88F670(   MOV DWORD PTR DS:[678FC8],0
00501E83  .  5B              POP EBX
00501E84  .  81C4 6810000(   ADD ESP,1068
00501E8A  .  C3              RETN
00501E8B  > 8B15 440F6A0(    MOV EDX,DWORD PTR DS:[6A0F44]
00501E91  .  57              PUSH EDI
00501E92  .  8B3D 5C465F0(   MOV EDI,DWORD PTR DS:[<&USER32.UpdateWi  USER32.UpdateWindow
00501E98  .  52              PUSH EDX                          hWnd => 00050990 (class='MDIClient',parent=
00501E99  .  FFD7            CALL EDI                          UpdateWindow
00501E9B  .  A1 400F6A00     MOV EAX,DWORD PTR DS:[6A0F40]
00501EA0  .  50              PUSH EAX                          hWnd => 00060994 ('mIRC',class='mIRC')
00501EA1  .  FFD7            CALL EDI                          UpdateWindow
00501EA3  .  85F6            TEST ESI,ESI
00501EA5  .  5F              POP EDI
00501EA6  .v 74 05           JE SHORT mirc.00501EAD
```

לאחר כמה נסיונות הבנתי שהפונקציה DialogBoxParamA חייבת להחזיר את הערך 0x7B0 על מנת שתמשך ריצת התכנית (ולא תצא משום שהתכנית נמצאת ב-Evaluation Period). לכן בחרתי לממש Inline Hook לפונקציה DialogBoxParamA. כאן פשוט אנווט את הפונקציה הבאה:

DialogBoxParamA

לפונקציה שלי, אשר תשנה את ערך האוגר EAX (המהווה את הערך החוזר מהפונקציה) לערך- 0x7B0 ותחזור מהפונקציה (זאת אומרת, מהפונקציה DialogBoxParamA).

ה-EXE:

```
void InjectToProcess(char* Path, HANDLE hProcess)
{
      DWORD Old;
    size_t PathLen = strlen(path);
    DWORD Address = VirtualAllocEx(hProcess, NULL, PathLen, NULL,
NULL);
    VirtualProtectEx(hProcess, Address ,PathLen, PAGE_READWRITE ,&Old);
    WriteProcessMemory(hProcess, Address, &Path, PathLen,
&BytesWritten);

   DWORD LLA = GetProcAddress("LoadLibraryA");
   CreateRemoteThread(hProcess,NULL,NULL, LLA, Address, NULL, NULL);
}
```

ה-EXE הוא רק קריאה לפונקציה InjectToProcess שמקבלת את ה-Path ל-DLL שברצוננו להזריק, ואת התהליך שאליו אנחנו מזריקים. הוא מקצה זיכרון חיצוני בתהליך, כותב את הפרמטרים ל-LoadLibraryA, ויוצר Thread חיצוני שיטען את ה-DLL.

ה-DLL:

```
#define JMPTO(From, To) ((TO) - (FROM) - 5)

void __declspec(naked) Hook()
{
    __asm {
                                MOV EAX, 0x7B0
                                RETN 14
                    };
                }
}

HookDialog(DWORD Function)
{
            DWORD Old;
            DWORD n;
```

```
            DWORD Function =
GetProcAddress(GetModuleHandle("User32.dll"), "DialogBoxParamA");
            VirtualProtect(Function, 5, PAGE_EXECUTE_READWRITE, &Old);
      *(BYTE *)Function = 0xE9; //JMP Opcode
      *(DWORD *)(Function+1) = JMPTO(Function, &Hook)//Calculate amount
of bytes to jmp
            VirtualProtect(Function, 5, Old, &n);

     //That's it...hooked.
}

DllMain(__in  HINSTANCE hinstDLL,__in  DWORD fdwReason,__in  LPVOID
lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH)
   {
         HookDialog();
   }
}
```

הדבר הראשון שאני עושה בכניסת ה-DLL היא השוואת הפרמטר fdw ל-dll_proces_attach. הפרמטר שווה לערך הזה כאשר ה-DLL נטען על ידי הפונקציה LoadLibraryA – עוד לפני שהתכנית עושה ב-DLL שימוש. לאחר מכן אני מבצע קריאה לפונקציה HookDialog, המשיגה את הכתובת של DialogBoxParamA, שזאת כמובן הפונקציה שברצוננו לעשות לה Hook. לאחר מכן, אני מבצע קריאה לפונקציה VirtualProtect על מנת לשנות את הגנות הדף סביב הפרולוג של DialogBoxParamA כדי שאוכל לכתוב אליו. פה מתבצעת הכתיבה של ה-JMP במקום הפרולוג – הבית 0xE9 הוא המזהה של האופקוד JMP. לאחר מכן מתבצע החישוב של האופסט המתאים לפונקציה שלנו, על ידי שימוש בנוסחא מפורסמת:

```
(TO-FROM-5)
```

הפונקציה Hook היא בעצם הפונקציה שאליה אנחנו קופצים, היא מוגדרת כ:

```
__declspec(naked)
```

מתאר זה מציין לקומפיילר שלא ישים לב לפונקציה Prologue ו-Epilogue ולכן גם עלינו להשתמש ב-RETN 14 בעצמנו (עלינו להשתמש ב-RETN 14 משום שיש להוריד את הפרמטרים שנדחפו לפונקציה מהמחסנית: 5 פרמטרים כפול 4 בתים לכל פרמטר == 0x14). כמו כן, יש לזכור כי כאשר אתם מבצעים Inline Hook אשר חוזר לפונקציה הקודמת, עליכם לממש בעצמכם את הפרולוג, כי אתם מוחקים אותו והוא דרוש להמשך פעולה תקינה של הפונקציה המקורית. מכאן אנחנו ממשיכים בשיחזור הגנות הדף הקודמות וזהו, עקפנו את ה-Nag ל-mIRC.

סיכום

Hooking היא דרך מצוינת להשיג הרבה מטרות, גם מנקודת המבט של תוקף זדוני וגם מנקודת המבט של האנטי-וירוס. לטכניקה זו יש יתרונות רבים, למשל, היא ממש פשוטה לממש ביחס לטכניקות אחרות. חיסרון גדול של הטכניקה הוא שהיא קלה מאוד לזיהוי, אך הדבר מהווה בעיה רק כאשר אתה מבצע פעולה זדונית ☺