;x = dword ptr -14h = ebp + EC (int)

;y = dword ptr -10h = ebp + F0 (int)

;entry = dword ptr -0Ch = ebp + F4 (IMAGE\_RESOURCE\_DATA\_ENTRY\*)

;id = dword ptr -8 = ebp + F8 (int)

;data = dword ptr -4 = ebp + FC (void\*)

mov edi, edi ; 8B FF ?

; Simulate the command ENTER 14,0 [C8 14 00] (14 is the size of the local variables)

push ebp ; 55

mov ebp, esp ; 8B EC

sub esp, 14h ; 83 EC 14

; Set size and handle for \_ResFindDataEntryFromImage@28 (0 / 0 = winload)

xor eax, eax ; 33 C0

xor ecx, ecx ; 33 C9

; Set the data pointer to NULL and push a pointer to the pointer

and [ebp+FC], 0 ; 83 65 [FC] [00]

lea edx, [ebp+FC] ; 8D 55 [FC]

push edx ; 52

; Push a pointer to the blank entry (won’t use it but need to provide it)

lea edx, [ebp+F4] ; 8D 55 [F4]

push edx ; 52

; Push a pointer to the name and id (NULL = no name)

push 0 ; 6A [00]

lea edx, [ebp+F8] ; 8D 55 [F8]

push edx ; 52

; Push the type (0xA = RC\_DATA)

push 0Ah ; 6A [0A]

; Set the id to 1

mov [ebp+F8], 1 ; C7 45 [F8] [01 00 00 00]

; Call ResFindDataEntryFromImage

call \_ResFindDataEntryFromImage ; E8 [XX XX XX XX] (relative)

test eax, eax ; 85 C0

jl short [to DONE] ; 7C [15] (relative)

; Set the position to draw as 0, 0 and get a pointer to the “POINT” structure

and [ebp+EC], 0 ; 83 65 [EC] [00]

and [ebp+F0], 0 ; 83 65 [F0] [00]

lea eax, [ebp+EC] ; 8D 45 [EC]

; Get the data pointer and call BgpGxDrawBitmapImage

mov ecx, [ebp+FC] ; 8B 4D [FC]

push eax ; 50

push ecx ; 51

call \_BgpGxDrawBitmapImage@12 ; E8 [XX XX XX XX] (relative)

; Return (and had 0 parameters)

DONE:

leave ; C9

retn ; C3

Total bytes: 71, return >=0 if the image was drawn, <0 otherwise

Next we need to call it:

Original code (total 25 bytes):

.text:004013E4 mov eax, graphics\_pack ; A1 ?? ?? 49 00

.text:004013E9 mov ecx, [eax] ; 8B 08

.text:004013EB test ecx, ecx ; 85 C9

.text:004013ED jz short loc\_4013FD ; 74 0E

.text:004013EF add eax, 8 ; 83 C0 08

.text:004013F2 push eax ; 50

.text:004013F3 push ecx ; 51

.text:004013F4 call \_BgpGxDrawBitmapImage@12 ; E8 ?? ?? 04 00

.text:004013F9 test eax, eax ; 85 C0

.text:004013FB jl short loc\_401416 ; 7C 19

New Code (total 9 bytes):

.text:004013E4 call \_TheNewFunction@0 ; E8 [XX XX XX XX] (relative)

.text:004013E9 test eax, eax ; 85 C0 (if eax>=0, jump)

.text:004013EB jge/jnl short loc\_401416 ; 7D [29] (relative)

(jumps over BgpDrawCopyright, BlResourceFindMessage, and BgDisplayString)

.text:004013ED through .text:004013FD are 16 nops (90)

Relocation removal:

Zero out:

.reloc / IMAGE\_DIRECTORY\_ENTRY\_BASERELOC : RVA 1000 : 33E5

The image is stored in “winload.exe : RC\_DATA : 1”

Must be a full BMP with “BM” bytes (not a DIB which is what BITMAP resources are)

No compression (the standard)

BPP must be 24 or 32

The DIB header must be size 40 (the standard, a Windows V3 or BITMAPINFOHEADER)

When decompiled this code looks like:

signed int \_\_cdecl sub\_457370()

{

signed int result; // eax@1

int x; // [sp+0h] [bp-14h]@2

int y; // [sp+4h] [bp-10h]@2

IMAGE\_RESOURCE\_DATA\_ENTRY \*entry; // [sp+8h] [bp-Ch]@1

int id; // [sp+Ch] [bp-8h]@1

void \*data; // [sp+10h] [bp-4h]@1

data = 0;

id = 1;

result = ResFindDataEntryFromImage(0, 0, 10, &id, 0, &entry, &data);

if ( result >= 0 )

{

x = 0;

y = 0;

result = BgpGxDrawBitmapImage(data, &x);

}

return result;

}