

Forum: [Hack Forums](#)

Thread: [Getting the contents of the .text section of a PE file](#)

Ozedath - March 5, 2016, 12:17 pm

I'm trying to read the contents of the .text section of notepad.exe and am using the code mentioned here:

<https://genesisdatabase.wordpress.com/2011/02/22/the-official-way-of-writing-a-crypter-in-c-source-code/>

```
Code // open it and get the size
hFile = CreateFileA(FileName, GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ, 0,
OPEN_EXISTING, 0, 0);
dwFileSize = GetFileSize(hFile, 0);

// load in memory
FileBuffer = (LPBYTE) malloc(dwFileSize);
ReadFile(hFile, FileBuffer, dwFileSize, &bytes, 0);

pDosH = (PIMAGE_DOS_HEADER) FileBuffer;

// check if it is valid PE, i would say that this is merely a proper check, for a proper one you would
need to calculate all the RVA's and see if they are valid
if(pDosH->e_magic != IMAGE_DOS_SIGNATURE)
    return -1;

pNtH = (PIMAGE_NT_HEADERS) (FileBuffer + pDosH->e_lfanew);

if(pNtH->Signature != IMAGE_NT_SIGNATURE)
    return -2;

pSecH = (PIMAGE_SECTION_HEADER) SECHDROFFSET(FileBuffer);

while(memcmp(pSecH->Name, ".text", 5)) // get the ".text" section header
    pSecH++;
```

When I get to the while loop I cannot seem to find any printable section names which indicates that the structure offsets are wrong.

How can the contents of .text be read from a PE file?

Only3 - March 5, 2016, 12:26 pm

DWORD seg = pSecH->VirtualAddress + pNtH->OptionalHeader.ImageBase; // the section address

Ozedath - March 5, 2016, 12:35 pm

(Mar 5, 2016 01:26 PM)Only3 Wrote: [CITE](#) DWORD seg = pSecH->VirtualAddress + pNtH->OptionalHeader.ImageBase; // the section address

The problem is that I do not know in which section I'm in. Whenever comparing the section names, I never get anything readable to be compared with the string ".text" in the while loop and eventually

get a memory page fault.

Is this macro correct:

```
Code #define SECHDROFFSET(a) ((LPVOID) ( (LPBYTE) a + \
      ((PIMAGE_DOS_HEADER)a->e_lfanew + \
      sizeof(IMAGE_NT_HEADERS)))
```

It has to jump over the DOS header from the beginning of the file.

excr4sh - March 5, 2016, 12:48 pm

Code // open it and get the size

```
hFile = CreateFileA(FileName, GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ, 0,
OPEN_EXISTING, 0, 0);
dwFileSize = GetFileSize(hFile, 0);
```

```
// load in memory
```

```
FileBuffer = (LPBYTE) malloc(dwFileSize);
ReadFile(hFile, FileBuffer, dwFileSize, &bytes, 0);
```

```
pDosH = (PIMAGE_DOS_HEADER) FileBuffer;
```

// check if it is valid PE, i would say that this is merely a proper check, for a proper one you would need to calculate all the RVA's and see if they are valid

```
if(pDosH->e_magic != IMAGE_DOS_SIGNATURE)
return -1;
```

```
pNtH = (PIMAGE_NT_HEADERS) ((ULONG_PTR)FileBuffer + pDosH->e_lfanew);
```

```
if(pNtH->Signature != IMAGE_NT_SIGNATURE)
return -2;
```

```
PIMAGE_SECTION_HEADER FirstSection;
FirstSection = IMAGE_FIRST_SECTION(pNtH);
for(DWORD n = 0; n < pNtH->FileHeader.NumberOfSections; n++)
{
    if(memcmp(FirstSection [n].Name, ".text", 5) == 0)
    {
        // found
    }
}
```

Ozedath - March 5, 2016, 2:44 pm

I've fixed it by calling

Code IMAGE_FIRST_SECTION()over the nt header.

0xDEAD10CC - March 5, 2016, 3:31 pm

The section headers immediately follow the optional header. Once you have the address of the optional header the file header has a member that will tell you the size of the optional header, and you can add that to get to the section header.

Ticoma - March 5, 2016, 4:31 pm

(Mar 5, 2016 04:31 PM)0xDEAD10CC Wrote: [\[CITE\]](#) The section headers immediately follow the optional header. Once you have the address of the optional header the file header has a member

that will tell you the size of the optional header, and you can add that to get to the section header. Could you demonstrate using standard functions>?

0xDEAD10CC - March 5, 2016, 5:46 pm

(Mar 5, 2016 05:31 PM)Ticoma Wrote: [\[CITE\]](#) Could you demonstrate using standard functions>? Standard functions? would you clarify what standard is? There's lots of Win32 functions you can use to grab the image headers. For the notepad PE i'm looking at, the size of the optional header is 240 bytes, which means that once you get the optional header offset, you can add that number of bytes to the offset, and you'll be at the section headers. Each entry will give you the raw address, and size of the section.

0x400 - 0x19200

excr4sh - March 5, 2016, 9:08 pm

(Mar 5, 2016 06:46 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Standard functions? would you clarify what standard is? There's lots of Win32 functions you can use to grab the image headers. For the notepad PE i'm looking at, the size of the optional header is 240 bytes, which means that once you get the optional header offset, you can add that number of bytes to the offset, and you'll be at the section headers. Each entry will give you the raw address, and size of the section.

0x400 - 0x19200There are no Win32 functions that you can use to working with the PE image, can you clarify what are you talking about?

0xDEAD10CC - March 5, 2016, 10:22 pm

(Mar 5, 2016 10:08 PM)excr4sh Wrote: [\[CITE\]](#) There are no Win32 functions that you can use to working with the PE image, can you clarify what are you talking about? Oh yeah? These must not exist then:

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms680225\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680225(v=vs.85).aspx)

And in addition to functions like FindResource and UpdateResource, there are others. Did you even try to look?