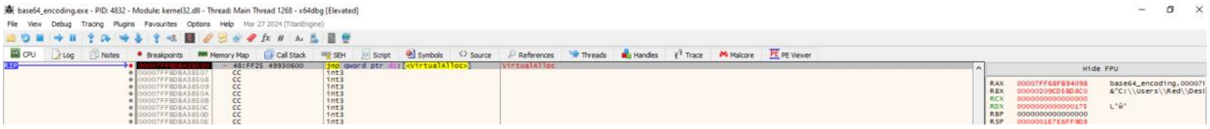
Reverse Engineering base64 Encoded Payload

So, here we will reverse engineer the base64 payload, and will extract the shellcode in .bin/.dump form.

So, open the file in the xdbg, and put the breakpoints at following points:

1. CryptStringToBinaryA
2. VirtualAlloc
3. VirtualProtect

So now run the program, and now you get to the first break point: “VirtualAlloc”.

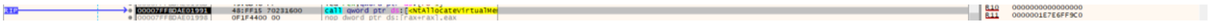


base64_encoding.exe - PID: 4832 - Module: kernel32.dll - Thread: Main Thread 1268 - x64dbg [Elevated]

```
00007FFBDBA38500  - 48:FF25 49930600   jmp qword ptr ds:[<VirtualAlloc>]   VirtualAlloc
00007FFBDBA38507    CC                 int3
00007FFBDBA38508    CC                 int3
00007FFBDBA38509    CC                 int3
00007FFBDBA3850A    CC                 int3
00007FFBDBA3850B    CC                 int3
00007FFBDBA3850C    CC                 int3
00007FFBDBA3850D    CC                 int3
00007FFBDBA3850E    CC                 int3
```

```
RAX  00007FF68FB94098   base64_encoding.00007F
RBX  00000209CD58D8C0   &"C:\\Users\\Red\\Des
RCX  0000000000000000
RDX  0000000000000175   L'ŵ'
RBP  0000000000000000
RSP  0000001E7E6FF9D8
```

Then form there step over and come to:



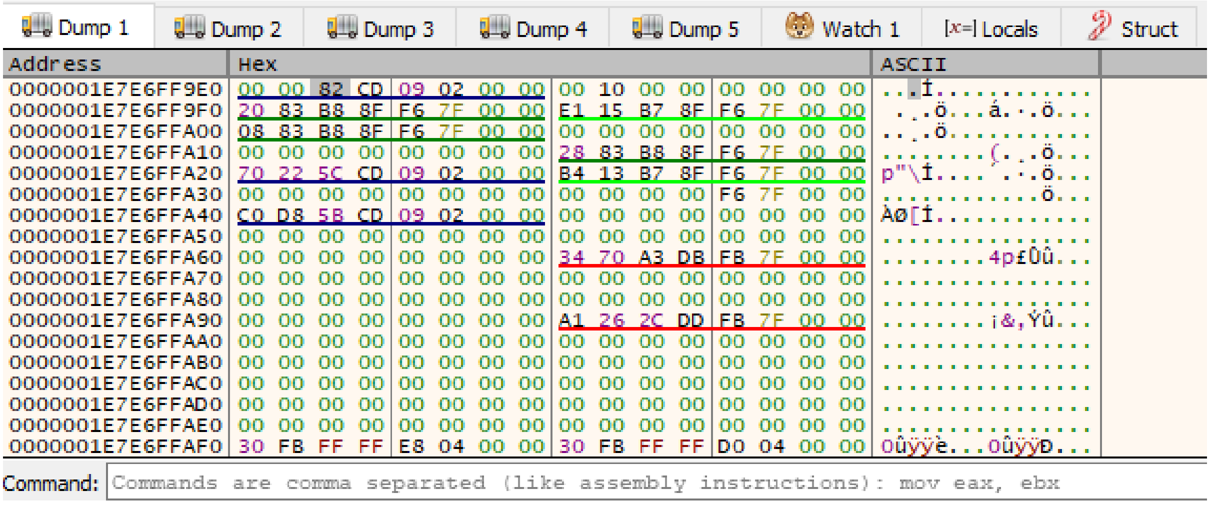
```
00007FFBDAE01991   48:FF15 70231600   call qword ptr ds:[<NtAllocateVirtualMe
00007FFBDAE01998   0F1F4400 00        nop dword ptr ds:[rax+rax],eax
```

Here we can it AllocateVirtualMemory, so if we see in the parameters.

So, we know that its 2nd parameter starts the allocation of the memory.

Then just press on step over

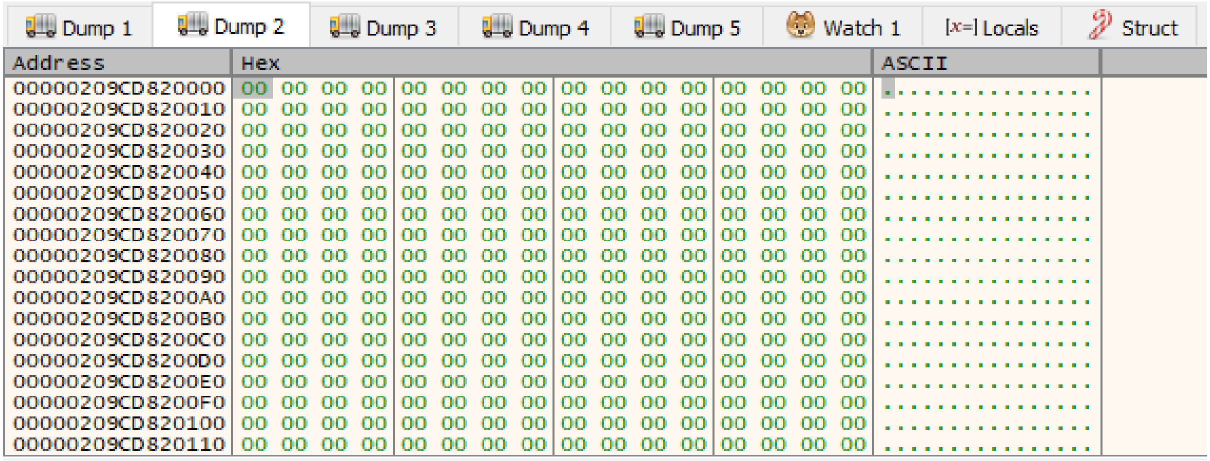
So, in the dump we can see it:



```
Address            Hex                                              ASCII
0000001E7E6FF9E0  00 00 82 CD 09 02 00 00 00 10 00 00 00 00 00 00  ..Í.............
0000001E7E6FF9F0  20 83 B8 8F F6 7F 00 00 E1 15 B7 8F F6 7F 00 00   .¸.ö...á.·.ö...
0000001E7E6FFA00  08 83 B8 8F F6 7F 00 00 00 00 00 00 00 00 00 00  ..¸.ö...........
0000001E7E6FFA10  00 00 00 00 00 00 00 00 28 83 B8 8F F6 7F 00 00  ........(.¸.ö...
0000001E7E6FFA20  70 22 5C CD 09 02 00 00 B4 13 B7 8F F6 7F 00 00  p"\Í....´.·.ö...
0000001E7E6FFA30  00 00 00 00 00 00 00 00 00 00 00 00 F6 7F 00 00  ............ö...
0000001E7E6FFA40  C0 D8 58 CD 09 02 00 00 00 00 00 00 00 00 00 00  ÀØXÍ............
0000001E7E6FFA50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFA60  00 00 00 00 00 00 00 00 34 70 A3 DB FB 7F 00 00  ........4p£Ûû...
0000001E7E6FFA70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFA80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFA90  00 00 00 00 00 00 00 00 A1 26 2C DD FB 7F 00 00  ........¡&,Ýû...
0000001E7E6FFAA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFAB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFAC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFAD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFAE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001E7E6FFAF0  30 FB FF FF E8 04 00 00 30 FB FF FF D0 04 00 00  0ûÿÿè...0ûÿÿÐ...
```

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

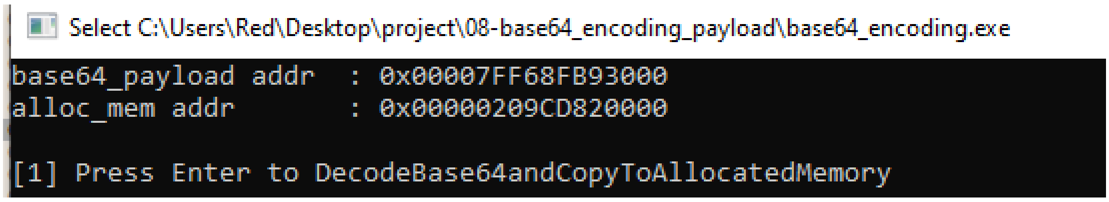
And in the dump, we can see the allocated memory:



```
Address           Hex                                                ASCII
00000209CD820000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD8200F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

As we can the address of the allocated memory is same as we had seen in the dump earlier.

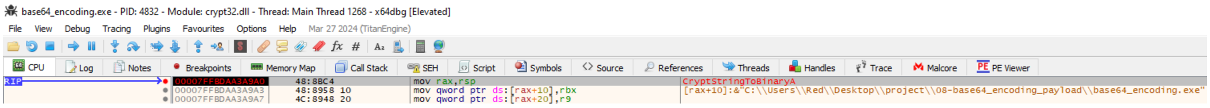
So, when we continue the program, in cmd it asks to enter, so just press it.


Select C:\Users\Red\Desktop\project\08-base64_encoding_payload\base64_encoding.exe

```
base64_payload addr  : 0x00007FF68FB93000
alloc_mem addr       : 0x00000209CD820000

[1] Press Enter to DecodeBase64andCopyToAllocatedMemory
```

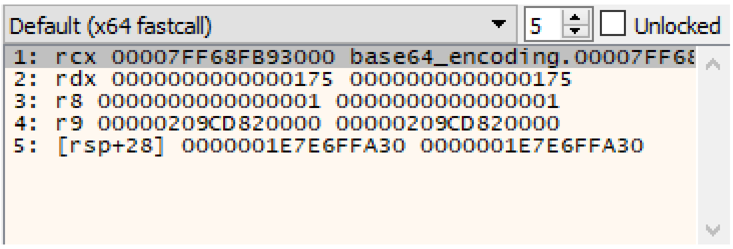
And now once again we hot another break point: “CryptStringToBinaryA”.



```
RIP ──────► 00007FFBDAA3A9A0   48:8BC4      mov rax,rsp                      CryptStringToBinaryA
            00007FFBDAA3A9A3   48:8958 10   mov qword ptr ds:[rax+10],rbx    [rax+10]:&"C:\\Users\\Red\\Desktop\\project\\08-base64_encoding_payload\\base64_encoding.exe"
            00007FFBDAA3A9A7   4C:8948 20   mov qword ptr ds:[rax+20],r9
```

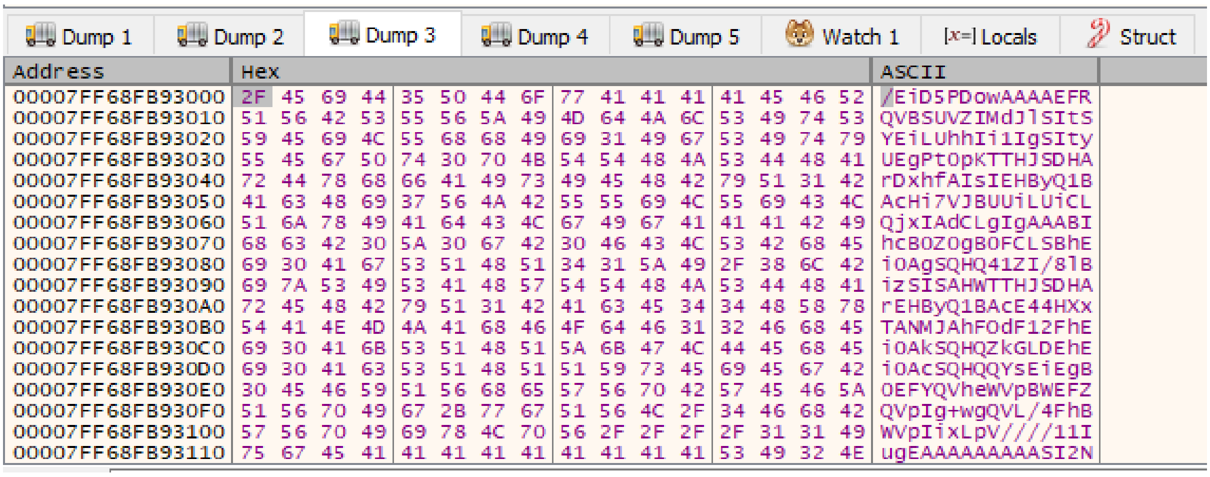
So, for the first parameter of CryptStringToBinary, if we follow in the dump, we can the encoded payload.

Here’s the first parameter:



```
1: rcx 00007FF68FB93000 base64_encoding.00007FF6
2: rdx 0000000000000175 0000000000000175
3: r8 0000000000000001 0000000000000001
4: r9 00000209CD820000 00000209CD820000
5: [rsp+28] 0000001E7E6FFA30 0000001E7E6FFA30
```

Here’s the dump:



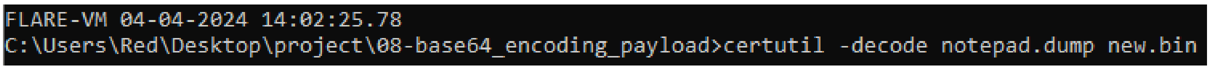
```
Address          Hex                                              ASCII
00007FF68FB93000 2F 45 69 44 35 50 44 6F 77 41 41 41 41 45 46 52 /EiD5PDowAAAAEFR
00007FF68FB93010 51 56 42 53 55 56 5A 49 4D 64 4A 6C 53 49 74 53 QVBSUVZIMdJlSItS
00007FF68FB93020 59 45 69 4C 55 68 68 49 69 31 49 67 53 49 74 79 YEiLUhhIi1IgSIty
00007FF68FB93030 55 45 67 50 74 30 70 4B 54 54 48 4A 53 44 48 41 UEgPt0pKTTHJSDHA
00007FF68FB93040 72 44 78 68 66 41 49 73 49 45 48 42 79 51 31 42 rDxhfAIsIEHByQ1B
00007FF68FB93050 41 63 48 69 37 56 4A 42 55 55 69 4C 55 69 43 4C AcHi7VJBUUiLUiCL
00007FF68FB93060 51 6A 78 49 41 64 43 4C 67 49 67 41 41 41 42 49 QjxIAdCLgIgAAABI
00007FF68FB93070 68 63 42 30 5A 30 67 42 30 46 43 4C 53 42 68 45 hcB0Z0gB0FCLSBhE
00007FF68FB93080 69 30 41 67 53 51 48 51 34 31 5A 49 2F 38 6C 42 i0AgSQHQ41ZI/8lB
00007FF68FB93090 69 7A 53 49 53 41 48 57 54 54 48 4A 53 44 48 41 izSISAHWTTHJSDHA
00007FF68FB930A0 72 45 48 42 79 51 31 42 41 63 45 34 34 48 58 78 rEHByQ1BAcE44HXx
00007FF68FB930B0 54 41 4E 4D 4A 41 68 46 4F 64 46 31 32 46 68 45 TANMJAhFOdF12FhE
00007FF68FB930C0 69 30 41 6B 53 51 48 51 5A 6B 47 4C 44 45 68 45 i0AkSQHQZkGLDEhE
00007FF68FB930D0 69 30 41 63 53 51 48 51 51 59 73 45 69 45 67 42 i0AcSQHQQYsEiEgB
00007FF68FB930E0 30 45 46 59 51 56 68 65 57 56 70 42 57 45 46 5A 0EFYQVheWVpBWEFZ
00007FF68FB930F0 51 56 70 49 67 2B 77 67 51 56 4C 2F 34 46 68 42 QVpIg+wgQVL/4FhB
00007FF68FB93100 57 56 70 49 69 78 4C 70 56 2F 2F 2F 2F 31 31 49 WVpIixLpV////11I
00007FF68FB93110 75 67 45 41 41 41 41 41 41 41 41 41 53 49 32 4E ugEAAAAAAAAASI2N
```

As we can it is the encoded payload.

So, for the 2nd parameter, it is number of strings that need to be converted.

And as we saw in the parameter it was 175, so there would be 175 encoded characters, so it starts from 3000, so follow it till 3174(address), that’s our encoded payload.

So, copy it then extract it, then using the cmd, we can just decode it:

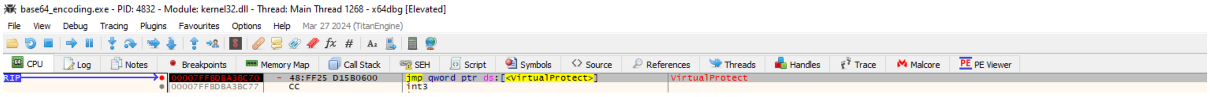


```
FLARE-VM 04-04-2024 14:02:25.78
C:\Users\Red\Desktop\project\08-base64_encoding_payload>certutil -decode notepad.dump new.bin
```

And then just upload in Hexeditor , you can compare it with the original payload, you can see that both are same.

Another way is to continue the program, because we know that it has already allocated the memory, so we can just take the payload from there.

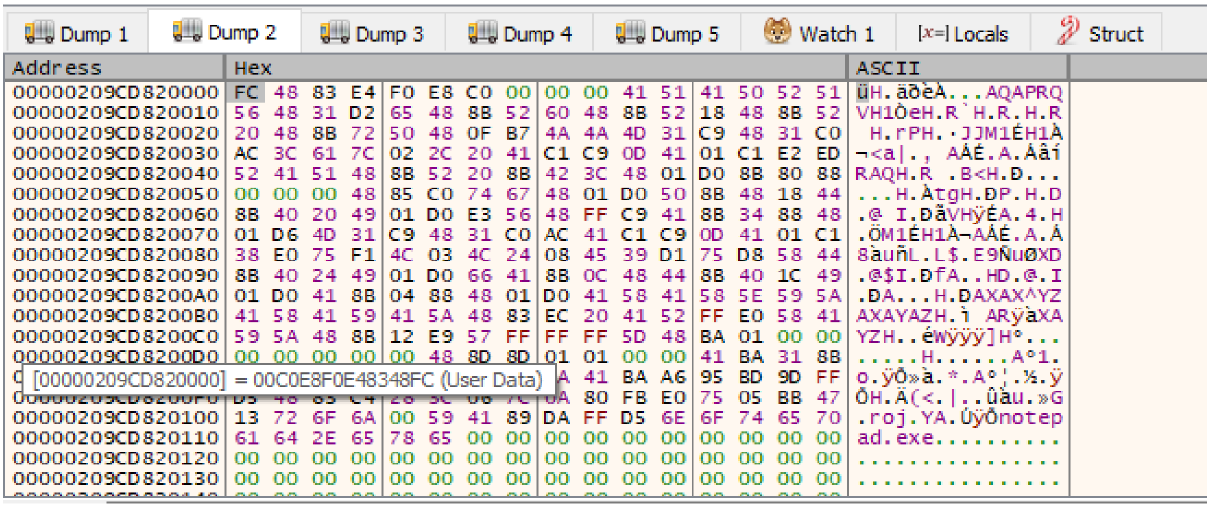
And as we run the program, we hit another breakpoint: “VirtualProtect”



base64_encoding.exe - PID: 4832 - Module: kernel32.dll - Thread: Main Thread 1268 - x64dbg [Elevated]

```
00007FFBDBA3BC70  - 48:FF25 D15B0600   jmp qword ptr ds:[<VirtualProtect>]   VirtualProtect
00007FFBDBA3BC77    CC                 int3
```

But in the dump, where we had seen the address if VirtualAlloc, we can now see the decoded payload:



```
Address           Hex                                              ASCII
00000209CD820000  FC 48 83 E4 F0 E8 C0 00 00 00 41 51 41 50 52 51  üH.äðèÀ...AQAPRQ
00000209CD820010  56 48 31 D2 65 48 8B 52 60 48 8B 52 18 48 8B 52  VH1ÒeH.R`H.R.H.R
00000209CD820020  20 48 8B 72 50 48 0F B7 4A 4A 4D 31 C9 48 31 C0   H.rPH.·JJM1ÉH1À
00000209CD820030  AC 3C 61 7C 02 2C 20 41 C1 C9 0D 41 01 C1 E2 ED  ¬<a|., AÁÉ.A.Áâí
00000209CD820040  52 41 51 48 8B 52 20 8B 42 3C 48 01 D0 8B 80 88  RAQH.R .B<H.Ð...
00000209CD820050  00 00 00 48 85 C0 74 67 48 01 D0 50 8B 48 18 44  ...H.ÀtgH.ÐP.H.D
00000209CD820060  8B 40 20 49 01 D0 E3 56 48 FF C9 41 8B 34 88 48  .@ I.ÐãVHÿÉA.4.H
00000209CD820070  01 D6 4D 31 C9 48 31 C0 AC 41 C1 C9 0D 41 01 C1  .ÖM1ÉH1À¬AÁÉ.A.Á
00000209CD820080  38 E0 75 F1 4C 03 4C 24 08 45 39 D1 75 D8 58 44  8àuñL.L$.E9ÑuØXD
00000209CD820090  8B 40 24 49 01 D0 66 41 8B 0C 48 44 8B 40 1C 49  .@$I.ÐfA..HD.@.I
00000209CD8200A0  01 D0 41 8B 04 88 48 01 D0 41 58 41 58 5E 59 5A  .ÐA...H.ÐAXAX^YZ
00000209CD8200B0  41 58 41 59 41 5A 48 83 EC 20 41 52 FF E0 58 41  AXAYAZH.ì ARÿàXA
00000209CD8200C0  59 5A 48 8B 12 E9 57 FF FF FF 5D 48 BA 01 00 00  YZH..éWÿÿÿ]Hº...
00000209CD8200D0  00 00 00 00 00 48 8D 8D 01 01 00 00 41 BA 31 8B  .....H......Aº1.
0 [00000209CD820000] = 00C0E8F0E48348FC (User Data)A 41 BA A6 95 BD 9D FF  o.ÿÕ»à.*.Aº¦.½.ÿ
00000209CD8200F0  D5 48 83 C4 28 3C 06 7C 0A 80 FB E0 75 05 BB 47  ÕH.Ä(<.|..ûàu.»G
00000209CD820100  13 72 6F 6A 00 59 41 89 DA FF D5 6E 6F 74 65 70  .roj.YA.ÚÿÕnotep
00000209CD820110  61 64 2E 65 78 65 00 00 00 00 00 00 00 00 00 00  ad.exe..........
00000209CD820120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000209CD820130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

So, copy it, and compare this payload with the original payload, we can see that both are same.