

Details:

Author: **Yossef Zidan (@yossefzidann)**

Challenge Overview: The challenge goal is find the correct input which is the flag.

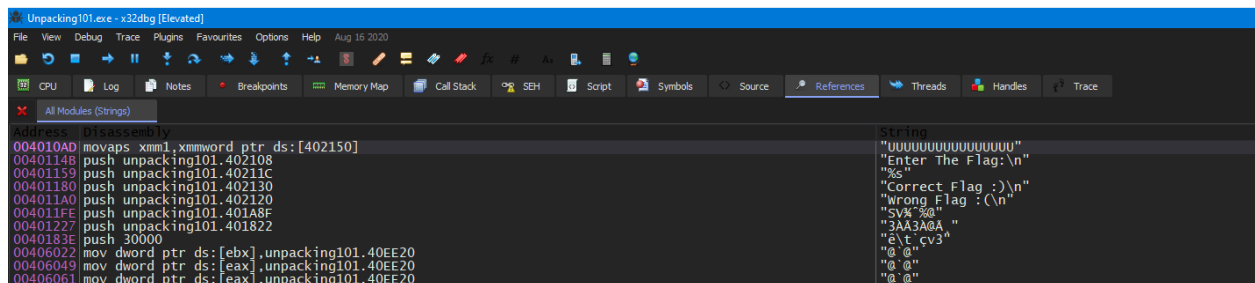
Step 1: Discovery

```
C:\Users\joezid\Desktop\Unpacking 101-Easy\Challenge
λ file Unpacking101.exe
Unpacking101.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

In this challenge, we are given one file which is an x86 PE file.

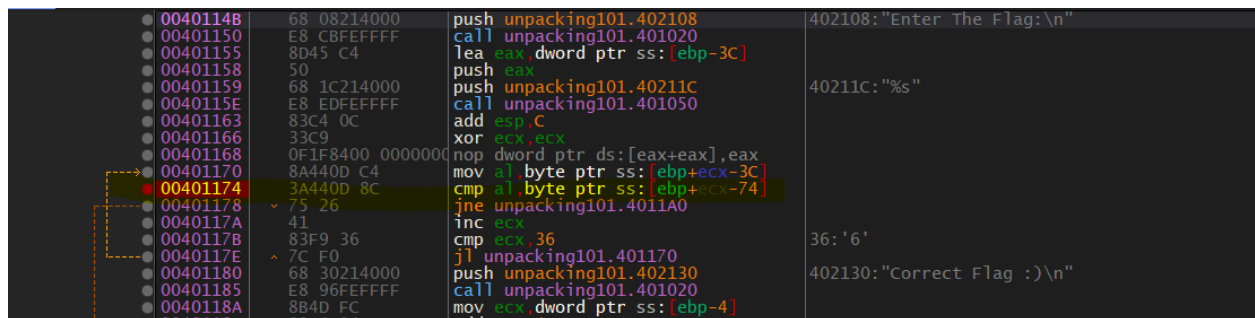
Step 2: Binary Analysis

If we checked the PE file in IDA we can notice that the pe file is packed so the easiest way to bypass the unpacking process and go directly to the flag check part is to run the PE and attach it to the process using any debugger.



```
Unpacking101.exe - x32dbg [Elevated]
File View Debug Trace Plugins Favourites Options Help Aug 16 2020
CPU Log Notes Breakpoints Memory Map Call Stack SBH Script Symbols Source References Threads Handles Trace
All Modules (Strings)
Address Disassembly String
004010AD movaps xmm1,xmmword ptr ds:[402150] "UUUUUUUUUUUUUUUUUU"
0040114B push unpacking101.402108 "Enter The Flag:\n"
00401159 push unpacking101.40211C "%s"
00401180 push unpacking101.402130 "Correct Flag :)\n"
004011A0 push unpacking101.402120 "Wrong Flag :(\n"
004011FE push unpacking101.401A8F "svk"
00401227 push unpacking101.401822 "3AA3A8A"
0040183E push 30000 "et cv3"
00406022 mov dword ptr ds:[ebx],unpacking101.40EE20 "0 0"
00406049 mov dword ptr ds:[eax],unpacking101.40EE20 "0 0"
00406061 mov dword ptr ds:[eax],unpacking101.40EE20 "0 0"
```

We can see some references to the strings we saw earlier enter the flag ...etc.



```
0040114B 68 08214000 push unpacking101.402108 402108:"Enter The Flag:\n"
00401150 E8 CBFEFFFF call unpacking101.401020
00401155 8D45 C4 lea eax,dword ptr ss:[ebp-3C]
00401158 50 push eax
00401159 68 1C214000 push unpacking101.40211C 40211C:"%s"
0040115E E8 EDFEFFFF call unpacking101.401050
00401163 83C4 0C add esp,C
00401166 33C9 xor ecx,ecx
00401168 0F1F8400 00000000 nop dword ptr ds:[eax+eax],eax
00401170 8A440D C4 mov al,byte ptr ss:[ebp+ecx-3C]
00401174 3A440D 8C cmp al,byte ptr ss:[ebp+ecx-74]
00401178 75 26 jne unpacking101.4011A0
0040117A 41 inc ecx
0040117B 83F9 36 cmp ecx,36 36:'6'
0040117E 7C F0 jl unpacking101.401170
00401180 68 30214000 push unpacking101.402130 402130:"Correct Flag :)\n"
00401185 E8 96FEFFFF call unpacking101.401020
0040118A 8B4D FC mov ecx,dword ptr ss:[ebp-4]
```

If we moved to the reference of the strings we can see that the input is being compared to the flag at 0x401174 we can put a breakpoint there and continue the execution.

Address	Disassembly	Comment	Register/Value
00401148	68 08214000	push unpacking101.402108	
00401150	E8 CBFFFFFF	call unpacking101.401020	
00401155	8D45 C4	lea eax, dword ptr ss:[ebp-3C]	
00401158	50	push eax	
00401159	68 1C214000	push unpacking101.40211C	
00401159	E8 EDFEFFFF	call unpacking101.401050	
00401161	83C4 0C	add eax, C	
00401166	33C9	xor eax, ecx	
00401168	0F1F8400 00000000	nop dword ptr ds:[eax+eax],eax	
00401170	844400 C4	mov al, byte ptr ss:[ebp+ecx-3C]	
00401170	3A4400 5C	cmp al, byte ptr ss:[ebp+ecx-74]	
00401178	75 26	jne unpacking101.4011A0	
0040117A	41	inc ecx	
0040117B	83F9 36	cmp ecx, 36	
0040117E	7C F0	j1 unpacking101.401170	
00401180	68 30214000	push unpacking101.402130	
00401185	E8 96FEFFFF	call unpacking101.401020	
0040118A	8B40 FC	mov ecx, dword ptr ss:[ebp-4]	
0040118D	83C4 04	add eax, 4	
00401190	33C0	xor eax, ebx	
00401192	B8 01000000	mov eax, 1	
00401197	E8 1A000000	call unpacking101.401186	
0040119C	8BE5	mov esi, ecx	
0040119E	5D	pop ebp	
004011A0	C3	ret	

Register	Value
EAX	00000061 'a'
EBX	0020E000
ECX	00000000
EDX	00000001
EBP	0063FE28
ESP	0063FEB4
ESI	009957A0
EDI	00996158
EIP	00401174 unpacking101.00401174
EFLAGS	00000344
ZF	1 PF 1 AF 0
OF	0 SF 0 DF 0
CF	0 TF 1 IF 1
LastError	00000000 (ERROR_SUCCESS)
LastStatus	C00700B8
CS	002B FS 0053
ES	002B DS 002B

Flag: **ASCWG{M4y_1_cL34r_y0ur_sL4t3_4nd_w1p3_y0ur_c0nsc13nc3}**