

Challenges : Basic 06

Author : Raz0r

Korean :

Unpack를 한 후 Serial를 찾으시오. 정답인증은 OEP + Serial

Ex) 00400000PASSWORD

English :

Unpack, and find the serial. The solution should be in this format: OEP + Serial

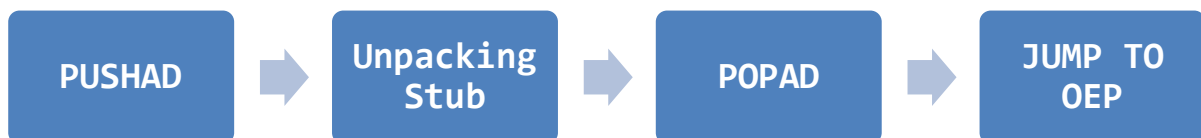
Ex) 00400000PASSWORD

[Download](#)

<문제>

언패킹을 한 후 OEP와 Serial 을 찾아 인증을 하라는 문제이다.

이번에는 ESP Trick을 통해 언패킹을 해 볼텐데 언패킹의 순서는 다음과 같다.



PUSHAD로 현재의 레지스터들을 스택에 저장시킨다.

그리고 Unpacking Stub을 통해 암호화된 코드들을 복구시키고, 정상적으로 복구가 되었을 경우 POPAD를 통해 아까 스택에 저장된 레지스터들의 상태를 다시 불러온다. 그리고 마지막으로 프로그램의 시작점인 OEP로 점프하게 된다.

잘 생각해보면 마지막에 POPAD를 통해 스택을 다시 복구시키는데 이 복구되는 부분에 bp를 걸어 실행하게 되면 곧바로 OEP로 점프하는 부분을 찾을 수 있게 될 것이다.

CPU - main thread, module 06			
004298F0	\$ 60	PUSHAD	
004298F1	. BE 00404200	MOV ESI,06.00424000	
004298F6	. 8DBE 00D0FDF	LEA EDI,DWORD PTR DS:[ESI+FFFD0000]	
004298FC	. 57	PUSH EDI	06.<ModuleEntryPoint>
004298FD	. 83CD FF	OR EBP,FFFFFFFF	
00429900	EB 10	JMP SHORT 06.00429912	
00429902	90	NOP	
00429903	90	NOP	
00429904	90	NOP	
00429905	90	NOP	
00429906	90	NOP	
00429907	90	NOP	
00429908	> 8A06	MOV AL, BYTE PTR DS:[ESI]	
0042990A	. 46	INC ESI	06.<ModuleEntryPoint>
0042990B	. 8807	MOV BYTE PTR DS:[EDI],AL	
0042990D	. 47	INC EDI	06.<ModuleEntryPoint>
0042990E	> 01DB	ADD EBX,EBX	
00429910	~ 75 07	JNZ SHORT 06.00429919	
00429912	> 8B1F	MOV EBX,DWORD PTR DS:[ECX]	

위 사진을 보면 프로그램을 로드시킨 후 F8로 명령어를 실행시킨 모습이다.

004298F0의 주소값에 PUSHAD가 들어가 있는 모습을 볼 수 있다.

또한 오른쪽 FPU 창을 보면 ESP의 값이 빨간색으로 표시된 것을 볼 수 있는데

이것은 값이 바뀌었다는 것을 시각적으로 표현해주고 있는 것이다.

(ESP는 최근에 스택에 넣은 데이터를 가리키고 있음)

저 ESP값으로 덤프창을 이동시켜보면,

Address	Hex dump	ASCII
0019FF64	F0 98 42 00 F0 98 42 00 94 FF 19 00 84 FF 19 00	?B.?B.?-.?-.?
0019FF74	00 D0 39 00 F0 98 42 00 F0 98 42 00 F6 99 18 40	.??.?B.?B.?@
0019FF84	F4 38 BB 75 00 D0 39 00 D0 38 BB 75 F6 99 18 40	??.??.??.?@
0019FF94	DC FF 19 00 E3 5D 68 77 00 D0 39 00 48 2D AE 42	?-.?hw.?H-?
0019FFA4	00 00 00 00 00 00 00 00 00 D0 39 00 00 00 00 00?.....
0019FFB4	00 00 00 00 00 00 00 00 00 00 00 00 48 2D AE 42H-?
0019FFC4	A0 FF 19 00 00 00 00 00 E4 FF 19 00 50 ED 69 77	?-.....?-.P?w
0019FFD4	6C E1 C6 35 00 00 00 00 EC FF 19 00 AE 5D 68 77	?hw

0019FF64의 주소에 PUSHAD로 넣은 값들을 볼 수 있다.

위에 회색으로 표시한 곳을 마우스로 드래그한 후

마우스 오른쪽 -> BreakPoint -> Hardware, On Access -> Word 또는 DWORD 를 눌러 하드웨어

bp를 건 후, 프로그램을 F9로 실행시킨다.

CPU - main thread, module 06			
00429A39	. 58	POP EAX	KERNEL32.75BB38F4
00429A3A	. 61	POPAD	
00429A3B	. 8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]	
00429A3F	> 6A 00	PUSH 0	
00429A41	. 39C4	CMP ESP,EAX	
00429A43	. ^ 75 FA	JNZ SHORT 06.00429A3F	
00429A45	. 83EC 80	SUB ESP,-80	
00429A48	. - E9 1379FDFF	JMP 06.00401360	
00429A4D	00	DB 00	
00429A4E	00	DB 00	
00429A4F	00	DB 00	
00429A50	00	DB 00	
00429A51	00	DB 00	
00429A52	00	DB 00	
00429A53	00	DB 00	
00429A54	00	DB 00	
00429A55	00	DB 00	
00429A56	00	DB 00	
00429A57	00	DB 00	
00429A58	00	DB 00	
00429A59	00	DB 00	

사진에서 볼 수 있듯이 POPAD 바로 밑 부분에서 프로그램이 멈춰진 것을 볼 수 있다.

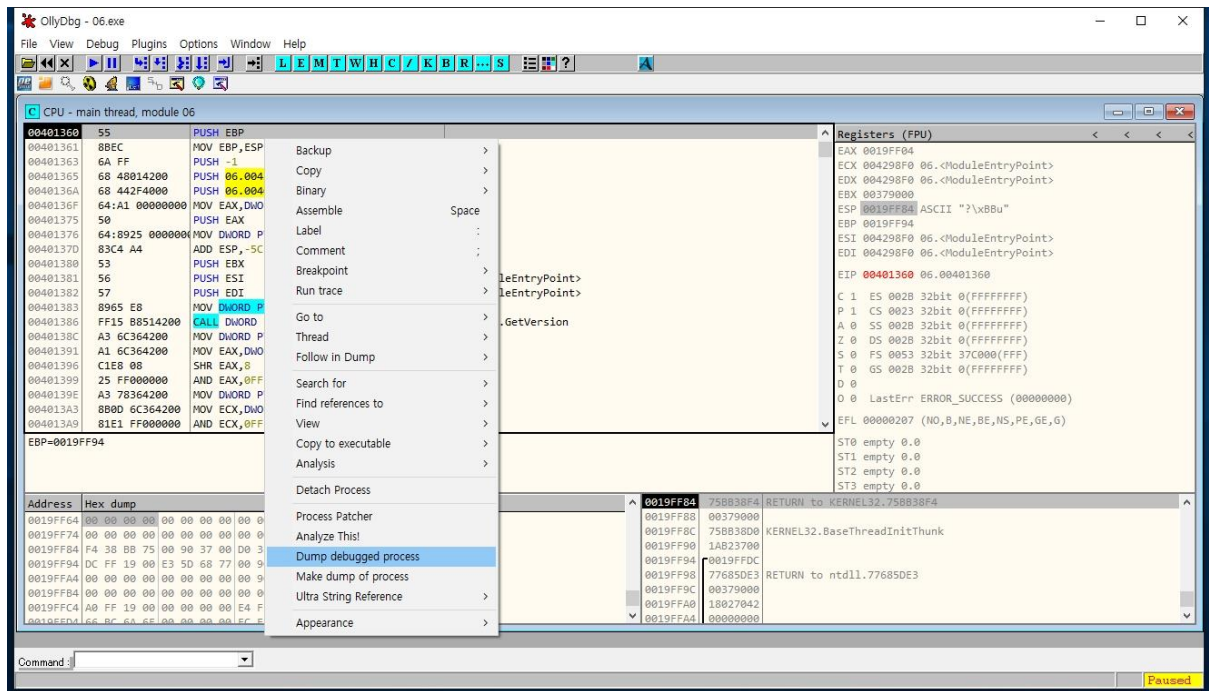
그리고 00429A48을 보게되면 어딘가로 점프를 하는 코드가 있는 것을 볼 수 있다.

(DEBUG 메뉴 -> Hardware Breakpoints 를 통해 하드웨어 bp를 삭제시킬 수 있음)

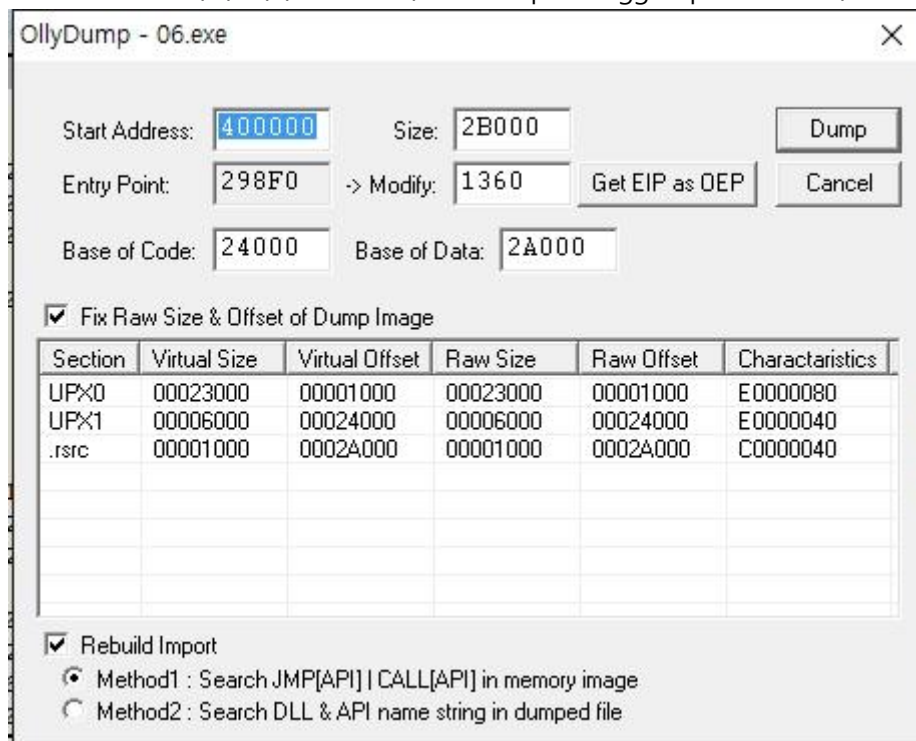
CPU - main thread, module 06			
00401360	55	PUSH EBP	
00401361	8BEC	MOV EBP,ESP	
00401363	6A FF	PUSH -1	
00401365	68 48014200	PUSH 06.00420148	
0040136A	68 442F4000	PUSH 06.00402F44	
0040136F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00401375	50	PUSH EAX	
00401376	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0040137D	83C4 A4	ADD ESP,-5C	
00401380	53	PUSH EBX	
00401381	56	PUSH ESI	06.<ModuleEntryPoint>
00401382	57	PUSH EDI	06.<ModuleEntryPoint>
00401383	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00401386	FF15 B8514200	CALL DWORD PTR DS:[425188]	KERNEL32.GetVersion
0040138C	A3 6C364200	MOV DWORD PTR DS:[42366C],EAX	
00401391	A1 6C364200	MOV EAX,DWORD PTR DS:[42366C]	
00401396	C1E8 08	SHR EAX,8	
00401399	25 FF000000	AND EAX,0FF	
0040139E	A3 78364200	MOV DWORD PTR DS:[423678],EAX	
004013A3	8B0D 6C364200	MOV ECX,DWORD PTR DS:[42366C]	
004013A9	81E1 FF000000	AND ECX,0FF	

따라서 401360이 OEP가 되는 것이다. (Procedure Prolog 또는 Function Prolog에 대해 찾아보면 더 자세히 알 수 있음)

이제 언패킹을 했으니 덤프를 해 보도록 하자. (OllyDump.dll이 있어야 함)



CPU 창에서 마우스 오른쪽 -> Dump debugged process 클릭



이런 창이 뜨게 되는데, 여기서 조심해야 할 점이 있다.

그게 Base of Code 와 Base of Data 인데 위 사진 상태로 그냥 Dump를 뜯 경우 프로그램을 올리디버거에 로드한 후 스트링을 찾아보면 아무 값도 나오지 않는 것을 볼 수 있다.

R Text strings referenced in 06:UPX1		
Address	Disassembly	Text string
00401360	PUSH EBP	<Initial CPU selection>
00424003	ASCII "	
0042400C	ASCII "	
0042402C	ASCII "	
0042404F	ASCII "	
0042405A	ASCII "	
00424124	ASCII "	
00424195	ASCII "	
004243F3	ASCII "	
0042444A	ASCII "	
004244B1	ASCII "	
004244F0	ASCII "	
004245B8	ASCII "	
004246D4	ASCII "	

따라서 위 사진처럼 나오지 않게 하기 위해서는 Base of Code 값을 바꿔주어야 한다.

올리디버거 창에서 ALT+M을 누르면 메모리맵을 볼 수 있는데

00382000	00007000			data block d	Priv	RW	RW	
00400000	00001000	06		PE header	Imag	R	RWE	
00401000	00023000	06	UPX0		Imag	R	RWE	
00424000	00006000	06	UPX1	code	Imag	R	RWE	
0042A000	00001000	06	.rsrc	data,import	Imag	R	RWE	
00465000	00000000				Priv	RW	Guar	RW

UPX0과 UPX1을 더블클릭해서 보면 UPX0에는 16진수로 이루어진 코드들이 있고 UPX1에는 어셈블리들을 볼 수 있다.

따라서 Base of Code와 Base of Data 값을 00401000의 RVA값인 1000으로 바꾸어주고 Dump를 뜨게되면 정상적으로 스트링이 나오는 것을 볼 수 있다.

OllyDump - 06.exe

Start Address: 400000

Size: 2B000

Dump

Entry Point: 298F0

-> Modify: 1360

Get EIP as OEP

Cancel

Base of Code: 1000

Base of Data: 1000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
UPX0	00023000	00001000	00023000	00001000	E0000080
UPX1	00006000	00024000	00006000	00024000	E0000040
.rsrc	00001000	0002A000	00001000	0002A000	C0000040

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image
☐ Method2 : Search DLL & API name string in dumped file

Address	Disassembly	Text string
0040106E	PUSH 1234.00422A30	ASCII "AD46DFS547"
00401083	PUSH 1234.00420048	ASCII "Good Job!"
00401088	PUSH 1234.00420038	ASCII "You got it ;>"
004010A7	PUSH 1234.00420030	ASCII "ERROR"
004010AC	PUSH 1234.0042001C	ASCII "Wrong serial!!!"
0040132E	PUSH 1234.00420068	ASCII "The value of ESP was not properly saved across
0040133A	PUSH 1234.00420054	ASCII "i386\Winchkesp.c"
00401360	PUSH EBP	<Initial CPU selection>
004016CC	PUSH 1234.00420220	ASCII "user32.dll"
004016E6	PUSH 1234.00420214	ASCII "wsprintfA"
00401716	PUSH 1234.004201E0	ASCII "Second Chance Assertion Failed: File %s, Line
00401779	PUSH 1234.004201B4	ASCII "_CrtDbgReport: String too long or IO Error"
00401799	MOV [LOCAL.3082],1234.004201A0	ASCII "Assertion failed: "
004017A5	MOV [LOCAL.3082],1234.0042018C	ASCII "Assertion failed!"
0040182F	PUSH 1234.00420178	ASCII "%s(%d) : %s"
0040184C	PUSH 1234.004201B4	ASCII "_CrtDbgReport: String too long or IO Error"

덤뜨른 파일을 올리디버거로 열어보면 한 눈에봐도 아 저게 Serial 이구나 하는 문자열을 볼 수 있다.

이제 OEP와 저 Serial을 이어 붙여서 인증하면 끝!