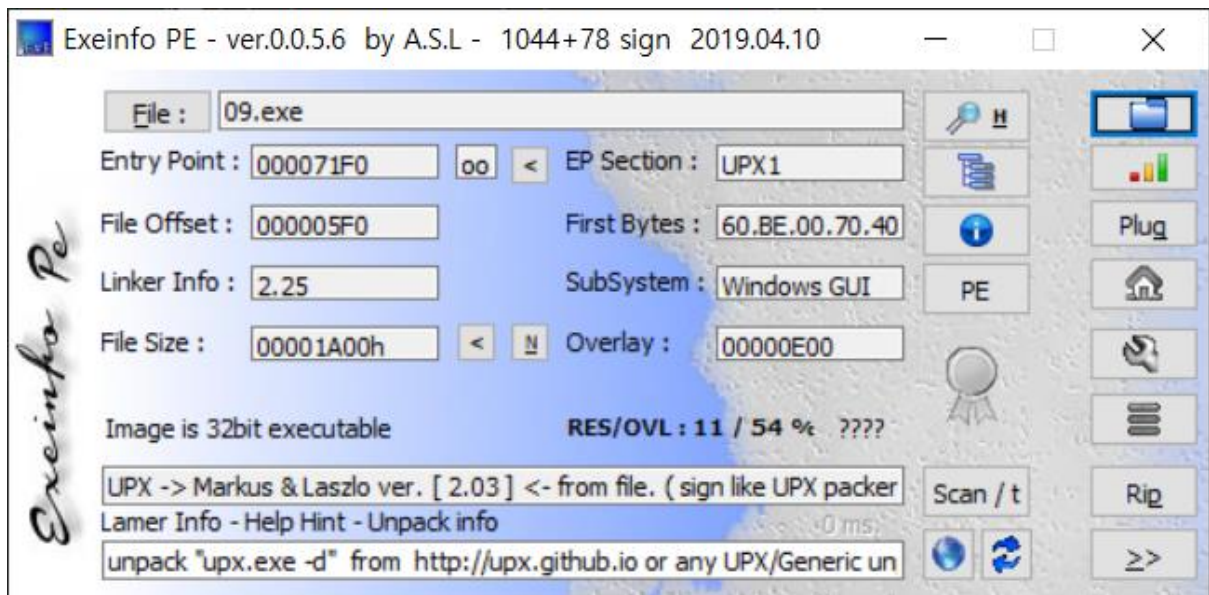
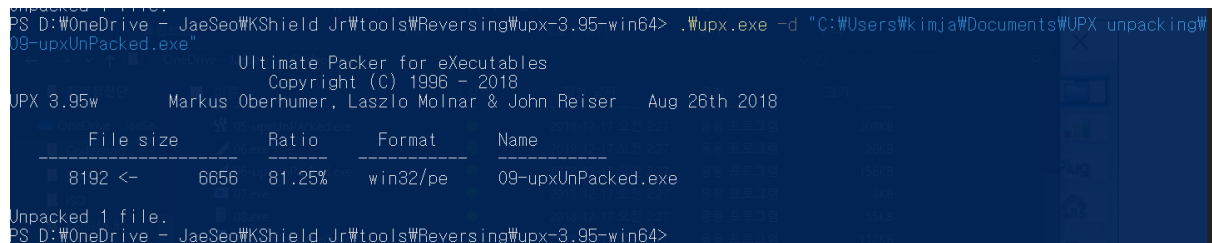


09.exe - StolenByte를 구하시오 Ex) 75156A0068352040

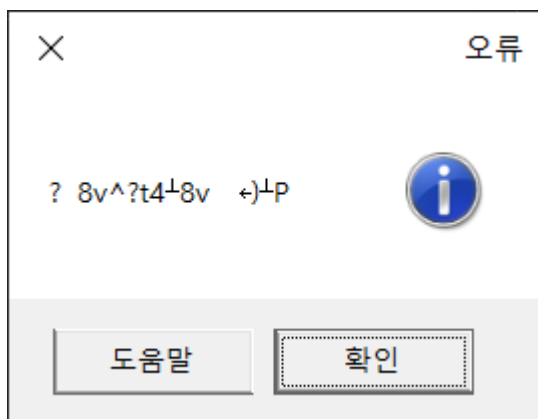
일단 PE 분석을 해본다.



그 결과 UPX로 Packing 되어 있는 것을 볼 수 있다. UPX로 UnPacking 한다.



이제 프로그램을 실행하여 분석을 한다.



Upx로 Unpacking를 하고 나니 문자열이 다 깨지고 오류 메시지를 출력하고 있다.

Upx에서 StolenByte를 일부 가지고 있어서 오류가 나는 것으로 예상이 된다.

그래서 이번에는 upx Unpacking을 진행하지 않고 디버깅을 진행하도록 한다.

이때 앞에 정상적으로 작동을 하였다면 보이는 메시지가 보인다.

StolenByte로 추정이 된다 앞에 BreakPoint를 걸고 분석을 해본다.

00407360	8060 28 7F	and byte ptr ds:[eax+28],7F	
00407364	58	pop eax	eax:"`UPX1"
00407365	50	push eax	eax:"`UPX1"
00407366	54	push esp	
00407367	50	push eax	eax:"`UPX1"
00407368	53	push ebx	
00407369	57	push edi	edi:"MZP"
0040736A	FFD5	call ebp	
0040736C	58	pop eax	eax:"`UPX1"
0040736D	61	popad	
0040736E	6A 00	push 0	
00407370	68 00204000	push 09.402000	402000:"abex' 3rd crackme"
00407375	68 12204000	push 09.402012	402012:"Click OK to check for the keyfile."
0040737A	8D4424 80	lea eax,dword ptr ss:[esp-80]	
0040737E	6A 00	push 0	
00407380	39C4	cmp esp,eax	
00407382	75 FA	jne 09.40737E	
00407384	83EC 80	sub esp,FFFFFFF80	
00407387	E9 809CFFFF	jmp 09.40100C	
0040738C	0000	add byte ptr ds:[eax],al	eax:"`UPX1"
0040738E	0000	add byte ptr ds:[eax],al	eax:"`UPX1"
00407390	0000	add byte ptr ds:[eax],al	eax:"`UPX1"
00407392	0000	add byte ptr ds:[eax],al	eax:"`UPX1"

Popad을 통해 oep에서 작동할 레지스터를 복구를 한 다음 StolenByte로 추정이 되는 명령들을 실행한다.

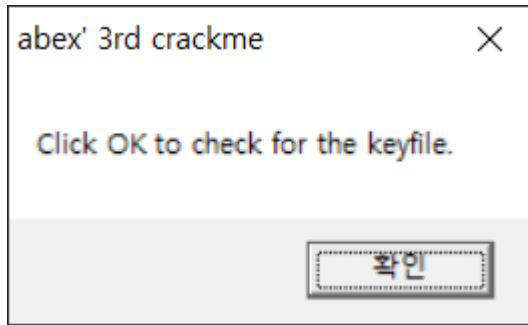
이때 push로 스택에 미리 명령어를 삽입 후 반복적으로 0을 집어넣어 esp와 eax를 같은 값으로 바꾸고 JMP를 하는 것을 볼 수 있다.

이제 실제로 이 코드를 UnPacking 된 파일에 집어넣어서 확인을 해본다.






00401008	90	nop	
00401009	90	nop	
0040100A	90	nop	
0040100B	90	nop	
0040100C	6A 00	push 0	
0040100E	E8 8C000000	call <JMP.&MessageBoxA>	
00401013	6A 00	push 0	
00401015	68 80000000	push 80	
0040101A	6A 03	push 3	
0040101C	6A 00	push 0	
0040101E	6A 00	push 0	
00401020	68 00000080	push 80000000	
00401025	68 B9204000	push 09-upxunpacked.4020B9	4020B9:"abex.12c"
0040102A	E8 5E000000	call <JMP.&CreateFileA>	
0040102F	A3 CA204000	mov dword ptr ds:[4020CA],eax	
00401034	83F8 FF	cmp eax,FFFFFFF	
00401037	74 3C	jle 09-upxunpacked.401075	
00401039	6A 00	push 0	
0040103B	FF35 CA204000	push dword ptr ds:[4020CA]	
00401041	E8 4D000000	call <JMP.&GetFileSize>	
00401046	83F8 12	cmp eax,12	
00401049	75 15	jne 09-upxunpacked.401060	
0040104B	6A 00	push 0	
0040104D	68 35204000	push 09-upxunpacked.402035	402035:"we'll done!"
00401052	68 40204000	push 09-upxunpacked.402040	402040:"Yep, keyfile found!"
00401057	6A 00	push 0	
00401059	E8 41000000	call <JMP.&MessageBoxA>	
0040105E	EB 28	jmp 09-upxunpacked.401088	
00401060	6A 00	push 0	

CPU	Graph	Log	Notes	Breakpoints	Memory Map	Call Stack	SEH	Script	Symbols	References	Threads
EIP ECX EDX ESI EDI											
00401000	6A 00	push 0									
00401002	68 00204000	push 09-upxunpacked.402000									
00401007	68 12204000	push 09-upxunpacked.402012									
0040100C	6A 00	push 0									
0040100E	E8 8C000000	call <JMP.&MessageBoxA>									
00401013	6A 00	push 0									
00401015	68 80000000	push 80									
0040101A	6A 03	push 3									
0040101C	6A 00	push 0									
0040101E	6A 00	push 0									
00401020	68 00000080	push 80000000									
00401025	68 B9204000	push 09-upxunpacked.4020B9									
0040102A	E8 5E000000	call <JMP.&CreateFileA>									
0040102F	A3 CA204000	mov dword ptr ds:[4020CA],eax									
00401034	83F8 FF	cmp eax,FFFFFFF									
00401037	74 3C	jle 09-upxunpacked.401075									
00401039	6A 00	push 0									
0040103B	FF35 CA204000	push dword ptr ds:[4020CA]									
00401041	E8 4D000000	call <JMP.&GetFileSize>									
00401046	83F8 12	cmp eax,12									
00401049	75 15	jne 09-upxunpacked.401060									
0040104B	6A 00	push 0									
0040104D	68 35204000	push 09-upxunpacked.402035									
00401052	68 40204000	push 09-upxunpacked.402040									
00401057	6A 00	push 0									
00401059	E8 41000000	call <JMP.&MessageBoxA>									
0040105E	EB 28	jmp 09-upxunpacked.401088									
00401060	6A 00	push 0									
00401062	68 79204000	push 09-upxunpacked.402079									
00401067	68 7E204000	push 09-upxunpacked.40207E									

NOP으로 채워져 있던 공간과 정확하게 일치하는 것을 볼 수가 있다. 실제로 실행을 시켜서 정상적인지 확인해본다.



위와 같은 모습으로 정상적으로 작동하는 모습을 볼 수 있다.

 09.exe		2018-12-17 오전 2:31	응용 프로그램	7KB
 09-upxUnPacked.exe		2018-12-17 오전 2:31	응용 프로그램	8KB
 09-upxUnPacked-Patched.exe		2019-10-06 오전 12:37	응용 프로그램	8KB

정답: 6A0068002040006812204000

0040100C	6A 00	push 0	
0040100E	E8 8C000000	call <JMP.&MessageBoxA>	
00401013	6A 00	push 0	
00401015	68 80000000	push 80	
0040101A	6A 03	push 3	
0040101C	6A 00	push 0	
0040101E	6A 00	push 0	
00401020	68 00000080	push 80000000	
00401025	68 B9204000	push 09.4020B9	4020B9:"abex.12c"
0040102A	E8 5E000000	call <JMP.&CreateFileA>	
0040102F	A3 CA204000	mov dword ptr ds:[4020CA],eax	
00401034	83F8 FF	cmp eax,FFFFFFFF	
00401037	74 3C	jle 09.401075	
00401039	6A 00	push 0	
0040103B	FF35 CA204000	push dword ptr ds:[4020CA]	
00401041	E8 4D000000	call <JMP.&GetFileSize>	
00401046	83F8 12	cmp eax,12	
00401049	75 15	jne 09.401060	
0040104B	6A 00	push 0	
0040104D	68 35204000	push 09.402035	402035:"we'll done!"
00401052	68 40204000	push 09.402040	402040:"Yep, keyfile found!"
00401057	6A 00	push 0	
00401059	E8 41000000	call <JMP.&MessageBoxA>	
0040105E	EB 28	jmp 09.401088	
00401060	6A 00	push 0	
00401062	68 79204000	push 09.402079	402079:"Error"
00401067	68 7F204000	push 09.40207F	40207F:"The found file is not a valid keyfile!"
0040106C	6A 00	push 0	
0040106E	E8 2C000000	call <JMP.&MessageBoxA>	
00401073	EB 13	jmp 09.401088	
00401075	6A 00	push 0	
00401077	68 54204000	push 09.402054	402054:"Error"
0040107C	68 5A204000	push 09.40205A	40205A:"HMMMMM, I can't find the file!"
00401081	6A 00	push 0	
00401083	E8 17000000	call <JMP.&MessageBoxA>	
00401088	E8 0C000000	call <JMP.&ExitProcess>	
0040108D	FF25 54304000	jmp dword ptr ds:[<&CreateFileA>]	JMP.&CreateFileA

Main 함수 내부로 추정이 되는 부분이다.

일단 처음에는 안내 메시지를 출력한 다음에 "CreateFileA"라는 함수를 호출 하여 파일이 있는지 없는지 체크를 한 다음 만약 있다면 "GetFileSize" 함수를 통해 키파일이 맞는지 에 대해 확인을 하고 메시지를 출력 하는 것으로 예상이 된다.

이제 직접 실행을 하여 어떻게 진행이 되는지 분석해본다.