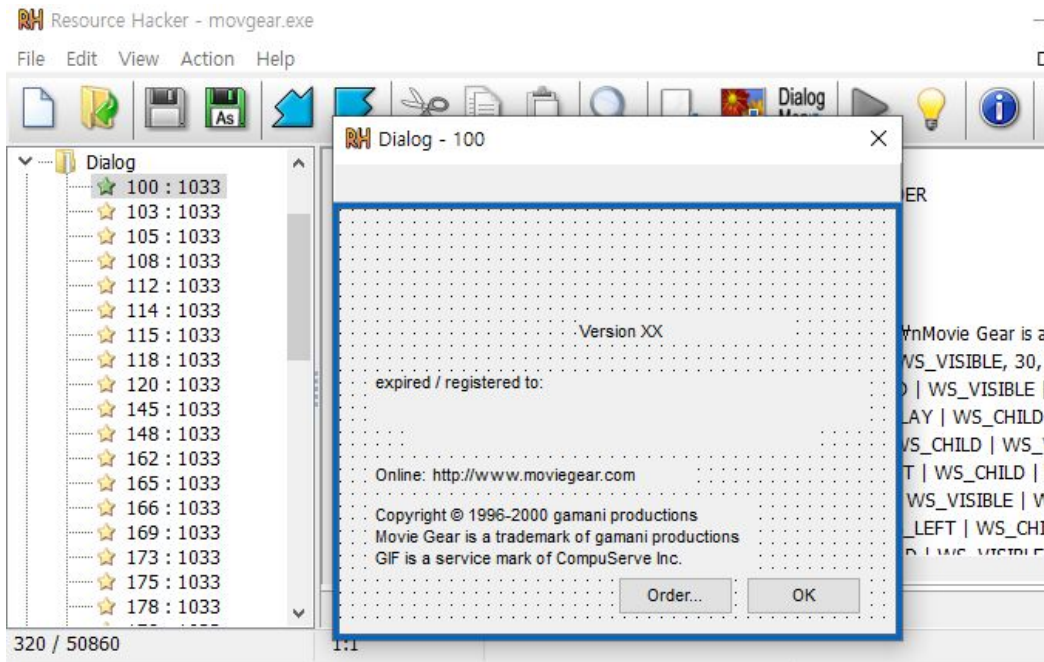




프로그램을 실행하고, 종료할 시 해당 Nag창이 뜬다. 이 Nag창을 Resource Hacker 툴을 써서 없애는게 이번 리버싱의 목적이다.



Resource hacker로 살펴보면
100번대에 아까 봤던 Nag창을 볼 수
있다. 100을 헥스 값으로 표현하면
64다.

push 64로 해당 명령을 사용하는
부분을 검색해본다.

R Found commands		
Address	Disassembly	Comment
0040395B	PUSH 64	
00406725	PUSH 64	
00406728	CALL DWORD PTR DS:[<USER32.DialogBoxPa	(Initial CPU selection)
00407062	PUSH 64	
0040D577	PUSH 64	
0043195E	PUSH 64	
00431979	PUSH 64	
00438F5D	PUSH 64	
00439E52	PUSH 64	
00439A2D	PUSH 64	
0043B53E	PUSH 64	
0043B5EA	PUSH 64	
0043D5E2	PUSH 64	
0043D686	PUSH 64	
0043EE39	PUSH 64	
00441092	PUSH 64	
004425DB	PUSH 64	Case 69 ('i') of switch 0044258F

push 64를 검색해보면 다양한
부분에서 사용하는걸 알 수 있다.
모든 부분에 bp를 걸고 하나씩
트레이싱 해본다.

```

00406707 > 6A 00      PUSH 0
00406709 . 6A 00      PUSH 0
0040670B . E8 40AF0200 CALL movgear.00431650
00406710 . 83C4 08     ADD ESP,8
00406713 . 83F8 01     CMP EAX,1
00406716 . 74 16       JE SHORT movgear.0040672E
00406718 . A1 68854600 MOV EAX,DWORD PTR DS:[468568]
0040671D . 6A 01       PUSH 1
0040671F . 68 D0E84000 PUSH movgear.0040E8D0
00406724 . 56          PUSH ESI
00406726 . 6A 64       PUSH 64
00406727 . 50          PUSH EAX
00406728 . FF15 E4824400 CALL DWORD PTR DS:[<&USER32.DialogBoxPa
0040672E > 8B8C24 B80000 MOV ECX,DWORD PTR SS:[ESP+B8]
00406735 . 8B9424 B40000 MOV EDX,DWORD PTR SS:[ESP+B4]
0040673C . 51          PUSH ECX
0040673D . 52          PUSH EDX
0040673E . 6A 10       PUSH 10
00406740 . 56          PUSH ESI
00406741 . FF15 0C844400 CALL DWORD PTR DS:[<&USER32.DefWindowPr
00406747 . 5F          POP EDI
00406748 . 5E          POP ESI
00406749 . 5D          POP EBP
0040674A . 5B          POP EBX

```

```

[
lParam = 00000001
DlgProc = movgear.0040E8D0
hOwner
pTemplate = 64
hInst => 00400000
DialogBoxParamA

[
lParam
wParam
Message = WM_CLOSE
hWnd
DefWindowProcA

```

프로그램이 실행되고, 프로그램을 종료시키면 해당 부분에서 bp가 걸린다. 아마 이 부분이 Nag 창을 호출하는 부분인것 같다. push 64중에 64가 무엇인지 찾아보니까 식별자라고 한다. 즉, DialogBoxParam을 사용하는 함수를 하나의 정수값으로 표현해 구별한다는 의미이다.

위쪽에 보면 분기문이 있다 조건에 만족하지 못하면 분기한다.



Nag창은 사라졌지만, 등록은 아직 안된 상태이다. 이걸 우회에 등록해보도록 하겠다.

00406707	> 6A 00	PUSH 0	
00406709	. 6A 00	PUSH 0	
0040670B	. E8 40AF0200	CALL movgear.00431650	
00406710	. 83C4 08	ADD ESP,8	
00406713	. 83F8 01	CMF EAX,1	
00406716	✓74 16	JE SHORT movgear.0040672E	
00406718	. A1 68854600	MOV EAX,DWORD PTR DS:[468568]	
0040671D	. 6A 01	PUSH 1	
0040671F	. 68 D0E84000	PUSH movgear.0040E8D0	
00406724	. 56	PUSH ESI	
00406725	. 6A 64	PUSH 64	
00406727	. 50	PUSH EAX	
00406728	. FF15 E4824400	CALL DWORD PTR DS:[<&USER32.DialogBoxPa	DialogBoxParamA
0040672E	> 8B8C24 B80000	MOV ECX,DWORD PTR SS:[ESP+B8]	
00406735	. 8B9424 B40000	MOV EDX,DWORD PTR SS:[ESP+B4]	
0040673C	. 51	PUSH ECX	
0040673D	. 52	PUSH EDX	
0040673E	. 6A 10	PUSH 10	
00406740	. 56	PUSH ESI	
00406741	. FF15 0C844400	CALL DWORD PTR DS:[<&USER32.DefWindowPr	DefWindowProcA
00406747	. 5F	POP EDI	
00406748	. 5E	POP ESI	
00406749	. 5D	POP EBP	
0040674A	. 5B	POP EBX	

위 분기문 위에 보면 함수가 있다. 이 함수에 의해 EAX가 결정되고 같은 경우 분기된다. 해당 함수를 분석해본다.

함수를 보니 RegOpenKeyExA를 통해 프로그램을 레지스터에 등록하고 있다. 이 함수에서 프로그램 등록 여부를 체크하는 것 같다.

0043164F	. 90	NOP	
00431650	✓81EC D0000000	SUB ESP,0D0	
00431656	. 8D4424 00	LEA EAX,DWORD PTR SS:[ESP]	
0043165A	. 53	PUSH EBX	
0043165B	. 56	PUSH ESI	
0043165C	. 57	PUSH EDI	
0043165D	. 50	PUSH EAX	
0043165E	. 68 19000200	PUSH 20019	
00431663	. 6A 00	PUSH 0	
00431665	. 68 F8B34400	PUSH movgear.0044B3F8	
0043166A	. 68 01000000	PUSH 00000001	
0043166F	. 83CB FF	OR EBX,FFFFFFFF	
00431672	. FF15 04804400	CALL DWORD PTR DS:[<&ADVAPI32.RegOpenKe	RegOpenKeyExA
00431678	. 85C0	TEST EAX,EAX	
0043167A	✓0F85 C2000000	JNZ movgear.00431742	
00431680	. 8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]	
00431684	. 8B35 08804400	MOV ESI,DWORD PTR DS:[<&ADVAPI32.RegQue	apphelp.6E871370
0043168A	. 8D5424 14	LEA EDX,DWORD PTR SS:[ESP+14]	
0043168F	. 51	PUSH ECX	

3431738	. 8BC8	MOV ECX, EAX	
343173D	. 83E1 03	AND ECX, 3	
3431740	. F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS:	
3431742	> 8B4C24 0C	MOV ECX, DWORD PTR SS:[ESP+C]	
3431746	. 51	PUSH ECX	
3431747	. FF15 00804400	CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey	hKey RegC
343174D	. 5F	POP EDI	
343174E	. 8BC3	MOV EAX, EBX	
3431750	. 5E	POP ESI	
3431751	. 5B	POP EBX	
3431752	. 81C4 00000000	ADD ESP, 000	
3431758	. C3	RETN	
3431759	. 90	NOP	
343175A	. 90	NOP	

아까 EAX에 따라 분기 여부를 결정한다고 했다. 프로그램 등록 관련 함수 RETN이 실행되기 전 MOV EAX, EBX를 통해 EAX 값을 결정하고 있다.

00431692	. 8B4424 1C	MOV EAX, DWORD PTR SS:[ESP+1C]	
00431696	. BF 64000000	MOV EDI, 64	
0043169B	. 68 98D44400	PUSH movgear.0044D498	ValueName = "RegName3"
004316A0	. 50	PUSH EAX	hKey
004316A1	. 897C24 28	MOV DWORD PTR SS:[ESP+28], EDI	
004316A5	. FFD6	CALL ESI	RegQueryValueExA
004316A7	. 85C0	TEST EAX, EAX	
004316A9	> 0F85 93000000	JNZ movgear.00431742	
004316AF	. 8D4C24 10	LEA ECX, DWORD PTR SS:[ESP+10]	
004316B3	. 8D5424 78	LEA EDX, DWORD PTR SS:[ESP+78]	
004316B7	. 51	PUSH ECX	pBufSize
004316B8	. 52	PUSH EDX	Buffer
004316B9	. 50	PUSH EAX	pValueType
004316BA	. 50	PUSH EAX	Reserved
004316BB	. 8B4424 1C	MOV EAX, DWORD PTR SS:[ESP+1C]	
004316BF	. 68 A4D44400	PUSH movgear.0044D4A4	ValueName = "RegCode3"
004316C4	. 50	PUSH EAX	hKey
004316C5	. 897C24 28	MOV DWORD PTR SS:[ESP+28], EDI	
004316C9	. FFD6	CALL ESI	RegQueryValueExA
004316CB	. 85C0	TEST EAX, EAX	
004316CD	> 75 73	JNZ SHORT movgear.00431742	
004316CE	. 8B4C24 78	LEA ECX, DWORD PTR SS:[ESP+78]	

함수를 처음부터 실행시켜 보면 RegQueryValueExA를 통해 레지스터 키를 비교하고 있다. 두 번의 API가 호출되는데, 해당 부분은 Name과 시리얼 번호가 있는지 없는지 확인하는 것이다. 등록된 시리얼 키가 없으니 조건문을 만족하지 못하고, 431742함수로 분기될 것이다.

00431730	. 83E1 03	AND ECX,3	
00431740	. F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:	
00431742	> 8B4C24 0C	MOV ECX,DWORD PTR SS:[ESP+C]	
00431746	. 51	PUSH ECX	
00431747	. FF15 00804400	CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey	hKey RegCloseKey
0043174D	. 5F	POP EDI	
0043174E	. 8BC3	MOV EAX,EBX	
00431750	. 5E	POP ESI	
00431751	. 5B	POP EBX	
00431752	. 81C4 00000000	ADD ESP,0D0	
00431758	. C3	RETN	

Registers (FPU)	
EAX	00000002
ECX	0056F473
EDX	00790000
EBX	FFFFFFFF
ESP	0019F038
EBP	0019F1FC
ESI	6E871370 apphelp.6E871370
EDI	00000064
EIP	00431742 movgear.00431742
EAX	FS:002B:32bit 0(FFFFFFFF)

분기하니 함수의 끝부분으로 이동하고, 이대로 **MOV EAX, EBX**를 통해 진행하면 아까와 같은 오류를 만날것이다. 그렇기에 해당 명령을 패치해줘야한다. 지금 **EBX**값은 **FFFFFFFF**이고, 이 값을 옮기면 안되기 때문에 **MOV EAX, EBX -> MOV AL, 1**로 바꿔줄 것이다. 이유는 원래 명령이 2바이트로 되어있기 때문에 똑같이 2바이트 크기로 패치 해줘야하기 때문이다. 그렇지 않으면 뒤에 명령을 덮어씌울것이다.

. F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:	
> 8B4C24 0C	MOV ECX,DWORD PTR SS:[ESP+C]	
. 51	PUSH ECX	
. FF15 00804400	CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey	hKey RegCloseKey
. 5F	POP EDI	
B0 01	MOV AL,1	
. 5E	POP ESI	
. 5B	POP EBX	
. 81C4 D0000000	ADD ESP,0D0	
. C3	RETN	
0A	NOP	

이렇게 패치를 해주면 된다. 이러면
아까 Nag창을 없애는 패치를 굳이
하지 않아도 Nag창이 뜨지 않는다.

