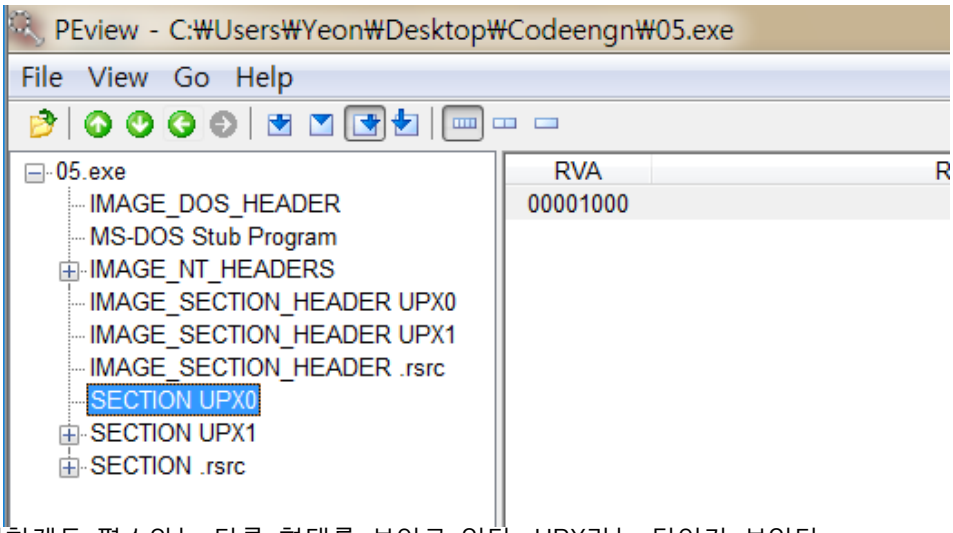
코드 엔진 Challenges: Basic 05

Author: Acid Bytes [CFF]
Korean: 이 프로그램의 등록키는 무엇인가 ?

문제를 확인했으니 파일을 다운로드 받아서 실행해보자.

파일을 실행해보니 이름과 시리얼을 받아서 인증하는 프로그램인 것 같다. 파일을 실행해 보았으니 분석을 해보자 .

먼저 PEView를 사용해 이 파일이 무슨 함수를 참조하는지 보자



이상하게도 평소와는 다른 형태를 보이고 있다. UPX라는 단어가 보인다.
이 파일은 UPX패킹이 되어있는 것 같다. 일단 함수를 살펴보도록하자.

RVA	Data	Description	Value
000570FC	0005718E	Hint/Name RVA	0000 LoadLibraryA
00057100	0005719C	Hint/Name RVA	0000 GetProcAddress
00057104	000571AC	Hint/Name RVA	0000 ExitProcess
00057108	00000000	End of Imports	KERNEL32.DLL
0005710C	000571BA	Hint/Name RVA	0000 RegCloseKey
00057110	00000000	End of Imports	advapi32.dll
00057114	000571C8	Hint/Name RVA	0000 ImageList_Add
00057118	00000000	End of Imports	comctl32.dll
0005711C	000571D8	Hint/Name RVA	0000 SaveDC
00057120	00000000	End of Imports	gdi32.dll
00057124	000571E0	Hint/Name RVA	0000 IsEqualGUID
00057128	00000000	End of Imports	ole32.dll
0005712C	000571EE	Hint/Name RVA	0000 VariantClear
00057130	00000000	End of Imports	oleaut32.dll
00057134	000571FC	Hint/Name RVA	0000 GetDC
00057138	00000000	End of Imports	user32.dll

다양한 함수를 사용한다. IsEqualGUID,RegCloseKey가 조금 수상한 것 빼고는 잘 모르겠다.

이 문제는 패킹에 대한 개념을 알려주는 문제인 것 같다. 패킹에 대해서 먼저 알아보자.

#패킹과 언패킹의 개념
Packing: 프로그램 코드 크기를 줄이려고 압축하거나 프로그램 분석을 어렵게 만들려고 암호화하는 것을 패킹이라고 한다.
(*단순 압축하는 것을 Compressing,암호화하는 것을 Protecting이라고 한다.)
Compressing: 실행코드를 압축해서 PE파일의 특정 섹션에 저장하고 프로그램이 실행될 때 공간에 압축을 풀어 실행하는 구조
Protecting: 실행 파일을 암호화해서 분석을 어렵게 만드는 기술로 실행코드는 암호화된 상태로 배포되고, 실행 시점에 복호화되어 동작을 수행한다. 약간의 속도 저하가 발생하지만 , 비밀보호가 중요한 프로그램이나 악성코드 같은 경우는 다양한 방식의 프로텍팅 기술이 적용된다

대표적인 오픈소스 패킹 도구인 UPX(Ultimatie Packer for eXecutables)기반으로 실행파일 살펴보자. UPX로 패킹된 파일은 언패킹에 필요한 코드영역(Unpacking Code) 과 패킹된 데이터가 들어가 있는 영역(Packed Data)그리고 패킹된 데이터가 저장된 빈 공간(Empty Space)로 구성된다.
-운영체제의 로더가 실행파일을 메모리에 로딩하고 나면
-바로 엔트리포인트로부터 프로그램이 실행된다. 패킹된 파일의 진입점은 언패킹 코드 영역에 들어있다.
-언패킹 코드는 패킹된 데이터를 하나씩 읽어 압축을 풀고 공간에 원본 데이터를 저장한다. 모든 코드가 언패킹되면 원 진입점에서부터 프로그램이 다시 시작된다(원 진입점이란 언패킹되기 이전 실행 파일의 진입점이다.)

#언패킹과 OEP

엔트리포인트는 프로그램에 대한 제어권이 운영체제에서 사용자 코드로 넘어가는 지점이다. 다시 말하면 프로세서가 메모리에 있는 코드섹션으로 처음으로 들어가는 지점이라 할 수 있다. 패킹된 프로그램을 디버거로 실행하면 실행이 완료된 시점에 언패킹된 코드가 메모리 어디엔가 저장된다. 물론 패커에 따라 실행이 완료되면 언패킹된 코드를 지우는 프로그램도 있다. 어차피 실행이 완료되면 모든 프로그램은 메모리상에 언패킹된 상태로 저장되어 있는데 메모리 덤프(메모리에 저장된 상태를 그대로 파일로 저장)를 떠서 실행 파일로 저장하면 그게 바로 언패킹된게 아닌가 생각하지만 여기서 중요한 것은 패킹되기 이전 프로그램의 실제 엔트리 포인트를 찾아야한다. 메모리 덤프를 파일로 저장했을 때 PE 헤더에 있는 엔트리 포인트 값을 실제 엔트리 포인트 주소로 수정해줘야 프로그램이 정상적으로 동작한다.

리버싱할 때 자주 언급되는 OEP(Original Entry Point)가 바로 패킹되기 이전 프로그램의 실제 엔트리 포인트다.

*언패킹 과정

-UPX로 패킹된 파일의 엔트리 포인트는 PUSHAD 명령어로 시작된다. PUSHAD 모든 레지스터의 값을 스택으로 백업하는 동작을 수행한다.

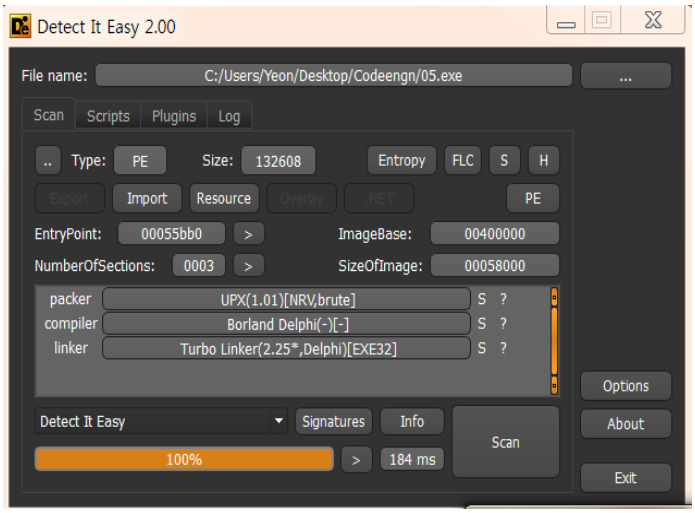
-다음으로 언패킹 코드가 뜨라오고 압축된 데이터를 풀어 메모리 특정 영역에 저장한다.

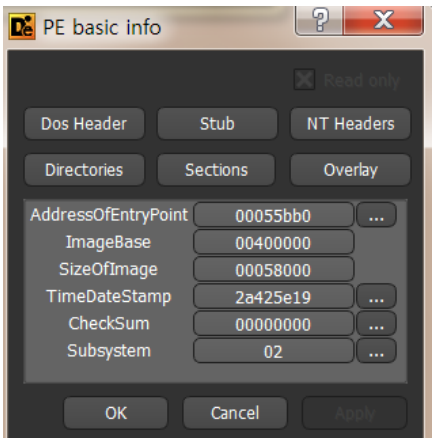
-언패킹이 끝나면 POPAD명령어를 수행해서 스택에 저장된 레지스터 값을 다시 복구 한다.

-이제 OEP로 점프해서 프로그램의 본래의 기능을 수행한다.

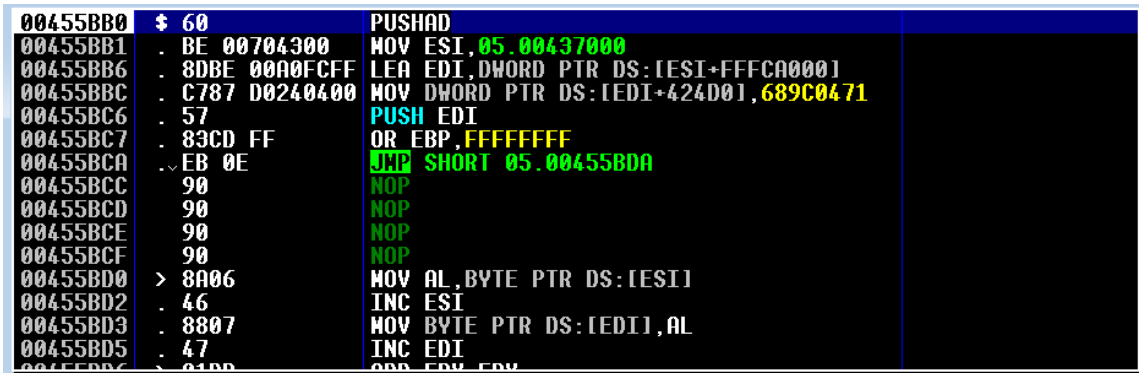
-OEP아래에서 패킹되기 전의 원본코드를 확인 가능하다.

~~패킹과 언패킹에 대한 개념을 대략적으로 알았으니 DE 프로그램으로 다시 파일을 살펴보도록~~ 하자.



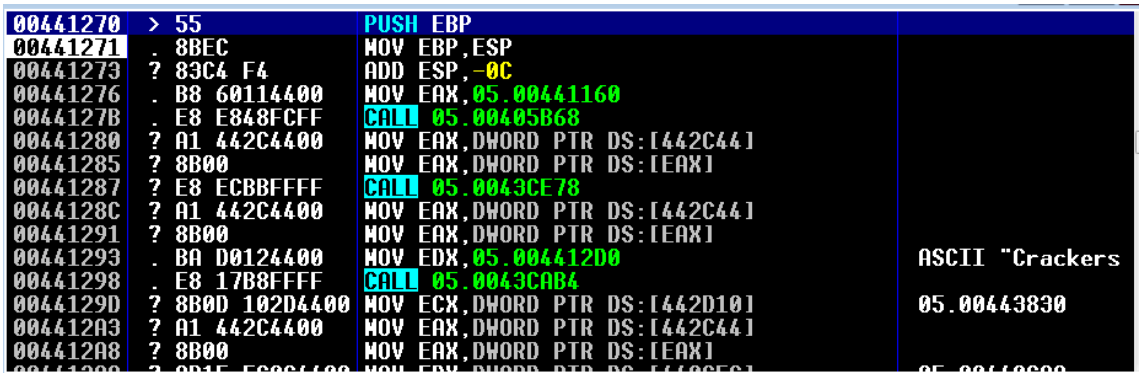


앞서 확인한 바와 같이 UPX로 패킹되어 있음을 확인할 수 있고 패킹된 프로그램의 엔트리포인트는 00455bb0이다. 그럼 올리디버거로 파일을 분석해보자.



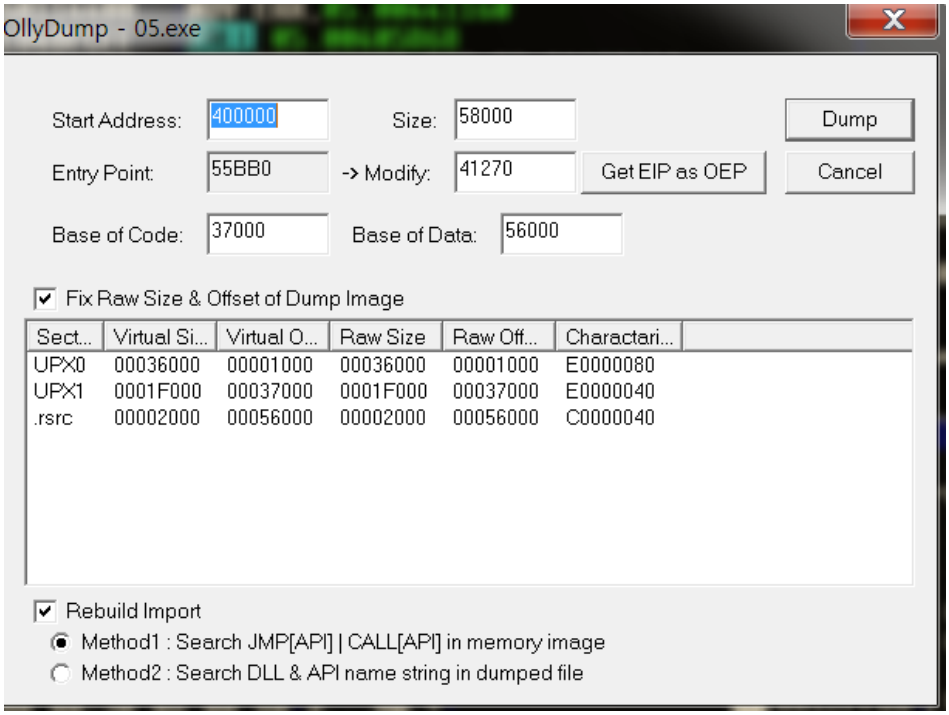
위에서 확인했던대로 00455bb0이 엔트리 포인트이며 UPX로 패킹된 파일의 엔트리 포인트는 PUSHAD 명령어로 시작되는 특징에 부합하게 PUSHAD 명령어로 시작된다.
언패킹이 끝나면 POPAD명령어를 수행해서 스택에 저장된 레지스터 값을 다시 복구하고 OEP로 점프해서 프로그램의 본래의 기능을 수행하며 OEP아래에서 패킹되기 전의 원본코드를 확인가능하므로 popad로 이동해보도록 하자.

POPAD가 위치한 곳으로 이동해보면 아래에 OEP로 가는 코드를 발견할 수 있다. 이 코드에 BP를 설정하고 프로그램을 실행한다음(F9)에 한단계 더 실행(F8)하도록 해서 올바른 코드가 있는곳으로 이동해보자.

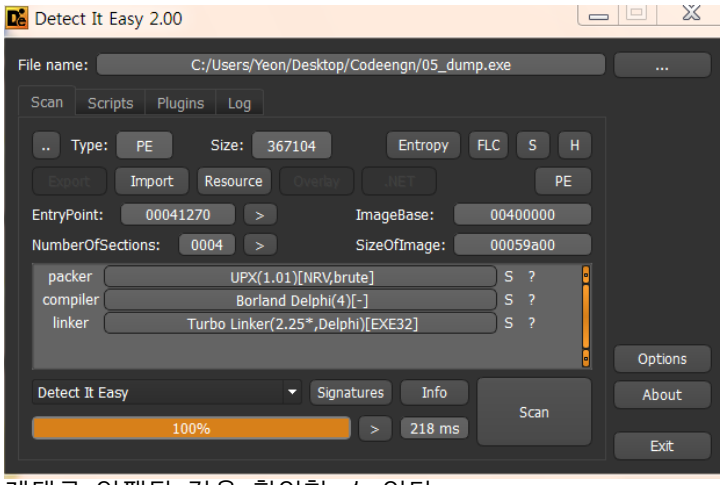


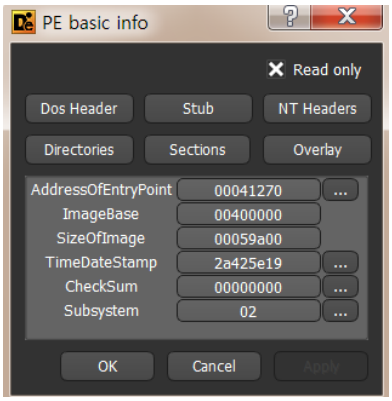
원래의 코드로 돌아왔다 이 코드를 다시 덤프해서 언패킹된 바이너리로 만들어보자.

#메모리 덤프
언패킹된 파일을 별로 저장하는 것으로 OllyDumpEx 플러그인을 이용할 수 있다.
OllyDumpEx플러그인은 메모리에 로딩된 PE 파일의 상태를 그대로 PE 파일로 다시 저장하는 기능을 지원하다. OllyDbg에서 자동으로 UPX를 언패킹 그 상태 그대로 PE 파일로 저장하면 패킹하기 전의 파일보다 크기는 좀 커지지만 원본 파일을 복구 할 수 있다.



Plugin -> ollydump -> Dump debugged Process 를 눌러서 덤프를 진행한 후 DE를 이용해 확인해보자.

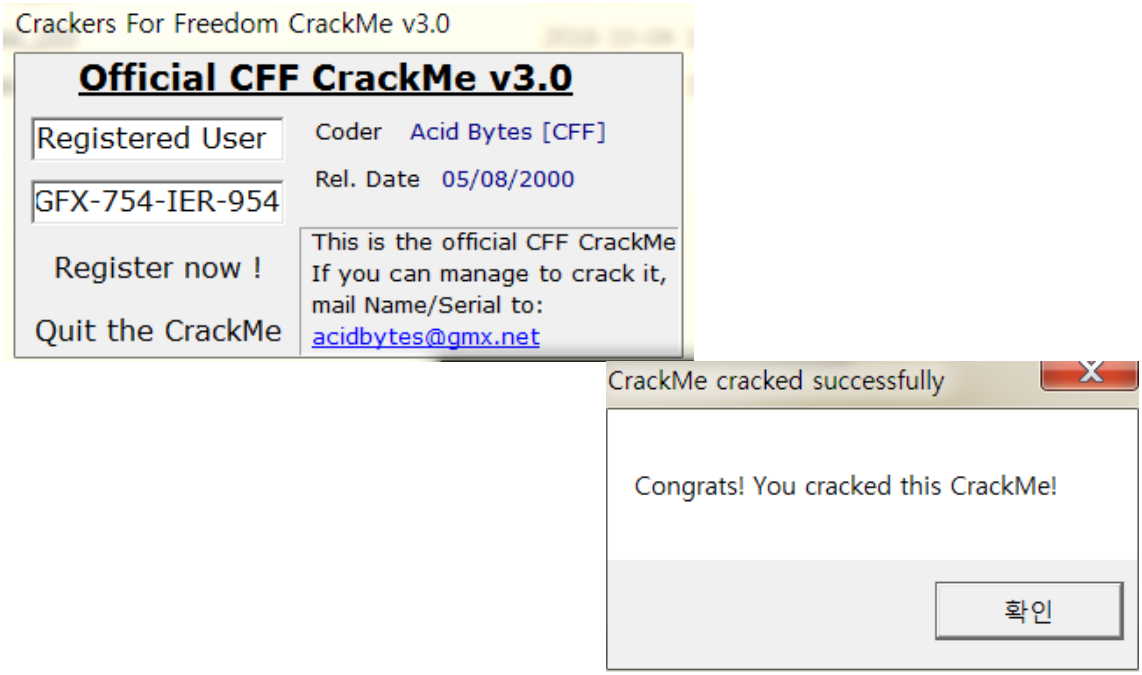




제대로 언팩된 것을 확인할 수 있다.

패스워드를 찾기위해 덤프한 파일을 올리디버거로 다시열어보자. 열어서 "Congrats!~"부분의 text strings 을 확인하여 주변의 아스키 코드를 확인해보면 "GFX-754-IER-954"라는 시리얼키가 나오는데 그 키를 입력해보면 인증이 되는 것을 확인할 수 있다.

```
00440F3E   . 8B83 C8020000  MOV EAX,DWORD PTR DS:[E
00440F44   . E8 D7FEFDFF    CALL 05_dump.00420E20
00440F49   . 8B45 FC        MOV EAX,DWORD PTR SS:[E
00440F4C   . BA 2C104400    MOV EDX,05_dump.0044102  ASCII "GFX-754-IER-954"
00440F51   . E8 D62BFCFF    CALL 05_dump.00403B2C
00440F56   .v75 1A          JNZ SHORT 05_dump.00440
00440F58   . 6A 00          PUSH 0
00440F5A   . B9 3C104400    MOV ECX,05_dump.0044103  ASCII "CrackMe cracked successfu
00440F5F   . BA 5C104400    MOV EDX,05_dump.0044105  ASCII "Congrats! You cracked thi
00440F64   . A1 442C4400    MOV EAX,DWORD PTR DS:[4
```



문제의 답은 GFX-754-IER-954이다 .