코드 엔진 Challenges: Basic 09

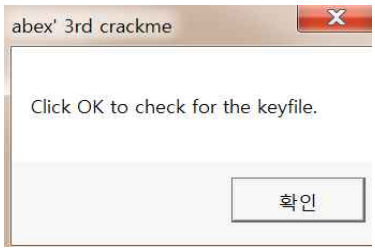
Author: abex
Korean: StolenByte를 구하라.
Ex)75156A0068352040

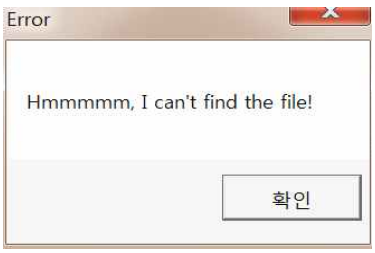
문제를 보니 프로그램내에 StolenByte가 존재한다는 것을 알 수 있다. StolenByte에 대해 간단하게 알아보자.

※StolenByte는 훔친 바이트란 의미로 프로그램의 한 부분의 코드를 훔쳐내어 다른 부분으로 옮겨진 코드를 말합니다. 주로 옮겨지는 코드는 엔트리 포인트 위의 몇 개의 옮겨진 코드들이며 OEP주소로 점프하기 전에 위치에서 PUSH 된다. 이러한 StolenByte는 주로 패커가 프로그램을 패킹할 때 볼 수 있는데 이렇게 옮겨진 코드들은 할당된 메모리 공간에서 실행된다. 이 때문에 패킹된 프로세스가 덤프될 때 StolenByte를 복구하지 못하면 프로그램은 정상적으로 작동하지 못하게 된다. 즉 , StolenByte란 패커가 위치를 이동시킨 코드로써 보호된 프로그램의 코드의 윗부분이다.

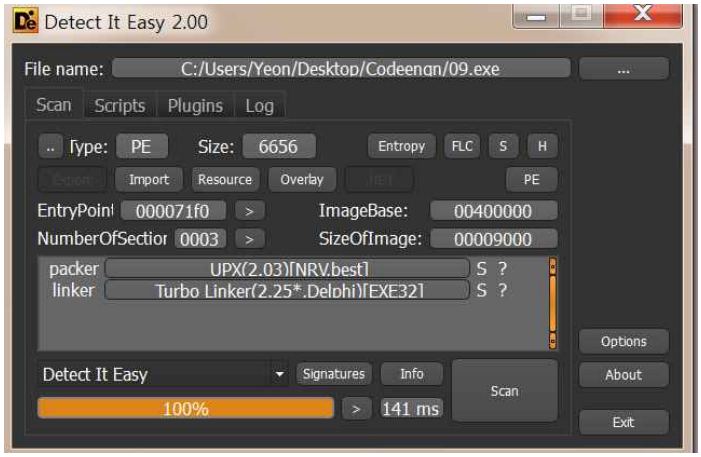
패킹 진행 과정에서 원본 코드 중 일부를 별도의 영역에서 실행하게 하여 OEP의 위치를 다른 위치로 가장하는 기법인 StolenByte를 이해했으니 올바른 언패킹을 진행하여 보자 .

문제를 확인했으니 파일을 다운로드 받아서 실행해보자.





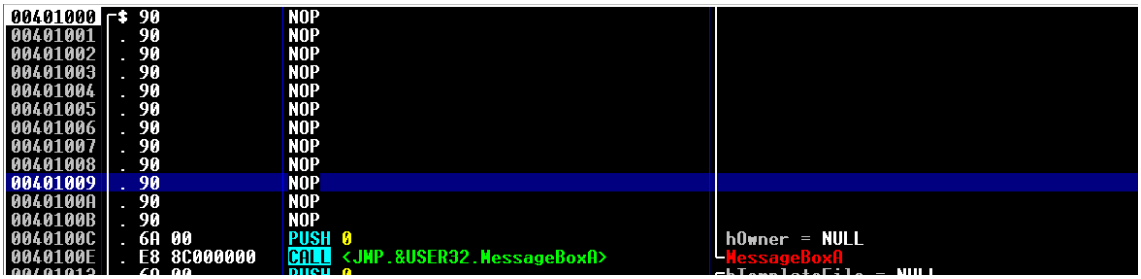
실행해보니 위와 같은 메시지를 출력한다. 프로그램의 형태를 확인했으니 DE를 통해 어떤 방식의 패킹이 사용되었는지 확인해보자.



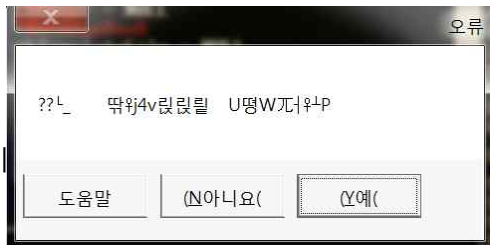
UPX로 패킹되어있다는 것을 확인할 수 있다. UPX.exe를 통해 언패킹을 해보자.

```
C:₩upx-3.95-win32>upx -d -o Basic_09_unpack.exe 09.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95w       Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
      8192 <-      6656   81.25%    win32/pe     Basic_09_unpack.exe

Unpacked 1 file.
```

언패킹이 된 것을 확인해보자 .그러면 올리디버거를 실행해서 제대로 언패킹되었는지를 확인해보자.



함수의 인자가 제대로 전달되지 않아 이상한 메시지가 출력 된 것을 볼 수 있다. StolenByte를 복구하지 않아 생긴 문제인 것을 알 수 있다. StolenByte는 주로 Entrypoint 위의 몇 줄의 코드를 이동 시키는데 이를 복구해보도록하자. 먼저 프로그램 복원을 위해 StolenByte를 찾아보자.



" OEP주소로 점프하기 전에 위치에서 PUSH 된다" 이러한 특성을 살펴보고 찾아보도록 하자. 다시 패킹전의 파일을 올리디버거로 열어보자 . 시작전에 간단하게 알아본 StolenByte의 특성 중에 훔쳐진 코드는 OPE 주소로 점프하기 전에 PUSH 된다는 것을 알 수 있다. UPX로 패킹되었을 때 OEP 주소로 점프하는 주소의 위치를 잘 알 고 있기에 그 위치로 이동해보자.

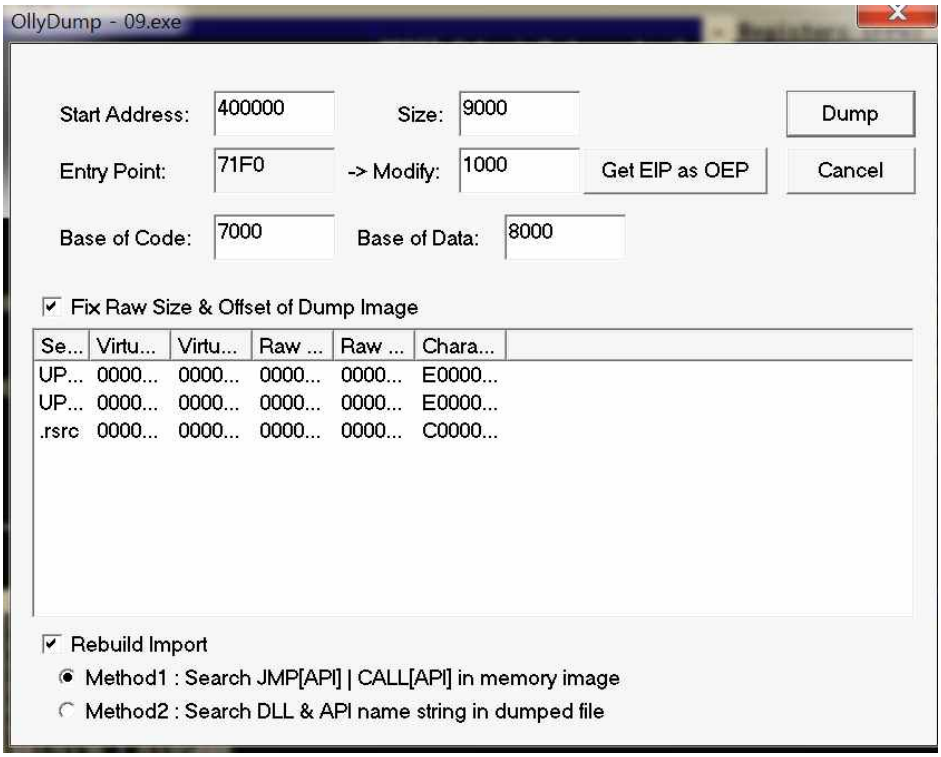
위치로 이동해보니 위의 그림과 같이 3개의 값이 PUSH되어 있는 것을 확인할 nt 있다. 이 부분이 훔친 코드임을 확인할 수 있다. 더 정확한 확인을 위해 OEP주소로 점프해보자. 만약 위의 코드가 훔친 코드가 맞다면 OEP 주소 위의 3개 값이 비어 있을 것이다.

그 전에 OEP로 점프하는 구간에 BP를 걸고 실행히켰더니 MessageBox의 함수 호출에 필요한 값이 스택에 PUSH됨을 확인할 수 있다.

```
#MessageBoxA함수

    int messagebox{
      HWND hwnd,        //생성될 메시지 상자의 소유자 윈도우에 대한 핸들
      LPCSTR lptext,    //표시할 메시지
      LPTSTR lpCaption, //대화상자제목
      UINT uType;       //대화 상자의 내용과 동작
}
```

OEP 주소로 이동해보니 위와 같이 여러개의 NOP이 있고 PUSH 0 과 messageboxA를 출력하고 있는 모습을 볼 수 있다. MessageBox함수의 기본구조에 파라미터값이 4개인 것을 봤을 때 1개만 입력된 상태이고 3개가 비어있는 것을 보니 훔쳐진 모드 3개와 비어있는 코드 3개가 일치하고 있는 것을 확인할 수 있다. 그럼 이제 훔쳐진 코드를 빈자리에 입력을 해보도록 하자. MessageBoxA함수 호출에 필요한 인자가 모두 스택에 적재되었다.

이제 StolenByte를 덮어쓴 프로그램을 올리덤프를 이용해서 덤프해보도록 하자. 아래와 같이 OEP 주소를 1000로 변경해야한다.



확인결과 정상적으로 실행되는 것을 알 수 있다.
코드엔진 인증값은 6A006800204000681220400O이다.