Crack me 문제 보고서

|  |  |
| --- | --- |
| 문제명 | Unpackme 07 |
| 파일명 | Unpackme 07.exe |
| 풀이자 | 서치원(2011003155) |
| 풀이 기간 | 2016/01/10 ~ 2017/01/16 |
| 문제 내용 | \* Unpacking 하여 OEP를 찾아내라.  \* IAT의 RVA 값을 찾아내라. |

문제 풀이:



실행 파일을 실행할 시의 모습입니다. 딱히 외적으로 푸는 요소는 없고 특정 주소만 찾아내는 문제입니다.



ollydbg로 확인한 결과 보통 packing 프로그램에서 나오는 PUSHAD가 없습니다. 그래서 UPX packing 방식이 아닌 다른 방식으로 되어있는지를 확인 해봅니다.



PEiD 프로그램으로 확인해본 결과 PECompact 형식이라는 생소한 방식으로 packing 되어 있습니다. 일단 이 형식이 어떻게 되는지를 알 수 없으니 진행해 봅니다.





실행을 해보면 401016에서 갑자기 멈추게 되는데 밑의 설명칸에 access violation이 일어났음을 알려주고 있습니다. 여기서 일어난 예외를 추적하는 기능을 사용해 코드를 추적합니다.



View 의 SEH Chain 기능을 사용한 화면입니다. 위의 주소들이 예외 처리를 해주는 곳이므로 break를 걸고 실행합니다.



예외 처리를 하는 곳으로 이동하고 진행하면 주소 41004F에서 스택에 레지스터 값을 넣는 구간이 보입니다. 즉, PUSHAD와 같은 역할을 하고 있습니다. 그렇다면 모든 레지스터를 POP하는 구간도 있을 것입니다.



진행한 결과 POP 구간을 찾고 그 후, JMP 하는 구간도 있음을 확인하였습니다. 아마도 저 주소가 시작지점으로 가는 부분일것입니다.



이동 후 Analysis code로 분석한 후의 상태입니다. 정상적인 코드가 보이게 됩니다. 이를 덤프해 저장하고 IAT 복구까지 합니다.



ImpRec 프로그램에서 복구한 후의 IAT 정보 화면입니다. 이 툴에서 RVA 값까지 자동으로 구할 수 있는데 RVA란 RAW라는 파일 상의 Import table의 위치 주소에서 rawoffset 을 빼고 VirtualOffset를 더한 값입니다.

다시 설명하자면 RVA는 PE 파일이 메모리에 로딩되었을 때의 메모리 주소이고 RAW는 PE 파일이 로딩 되기 전의 fileoffset으로 기존 offset(rawoffset)을 빼고 메모리에 올라간 offset(virtualOffset)을 더해서 구하는 것입니다.