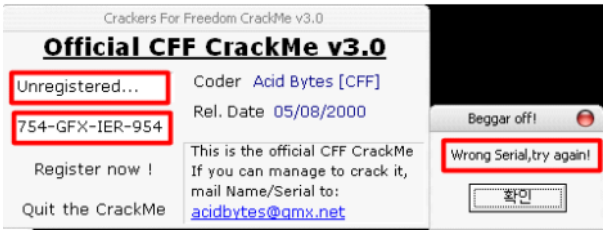
Reverse L05

2009년 12월 23일 수요일
오전 10:57

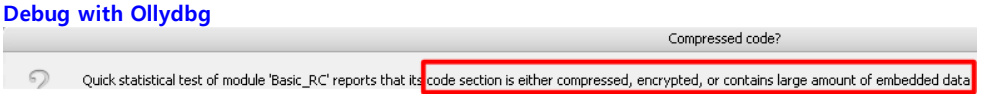
파일 확인

프로그램 실행



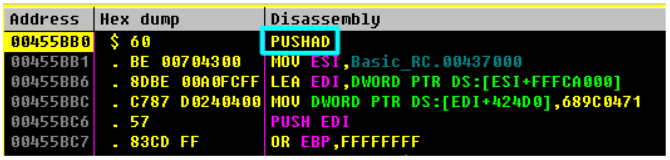
등록키를 입력하면 **MessageBox** 를 출력하는 프로그램이다.

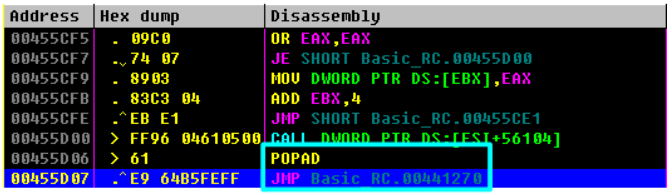
Debug with Ollydbg



Code Section이 Compressed 즉, **Packing 된 코드**란 것을 알 수 있다.

무시하고 Code 확인





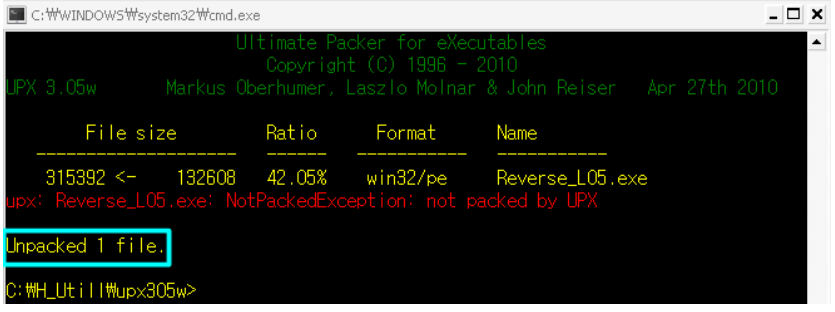
PUSHAD 로 시작해서 **POPAD** 로 끝난 후, **00441270 으로 무조건 JMP** 하는 Code 임을 확인할 수 있다.

- UPX로 Packing된 파일의 경우 프로그램 시작시 **PUSHAD를 실행**한다.
  - PUSHAD 명령은 모든 **32비트 범용 레지스터를 Stack에 Push**한다.
    - 프로그램 실행 준비가 완료된 시점의 레지스터를 **Stack에 백업**
  - **Unpack Routine이 실행되면서 레지스터를 사용하기 때문에, 복원하기 위해서 백업하는 것**

- 원본프로그램이 시작되는 위치 **OEP(Original Entry Point)**
  - **POPAD명령**을 이용해 OEP를 찾을 수 있다.
    - code안에 popad 명령어가 많다면 일일이 위치를 찾아서 확인해야 되기 때문에 어렵다.
  - POPAD 명령은 **백업한 레지스터를 다시 POP 시킨다.**

Unpacking with UPX Tool

간단히 **UPX 라는 Tool 을 사용해서 Unpacking** 해 보겠다.



Unpacking 후 Ollydbg 로 확인

```
Address   Hex dump      Disassembly                          Comment
00441270  $ 55          PUSH EBP
00441271  . 8BEC        MOV EBP,ESP
00441273  . 83C4 F4     ADD ESP,-0C
00441276  . B8 60114400 MOV EAX,Unpack.00441160
0044127B  . E8 E848FCFF CALL Unpack.00405B68
00441280  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
00441285  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
00441287  . E8 ECBBFFFF CALL Unpack.0043CE78
0044128C  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
00441291  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
00441293  . BA D0124400 MOV EDX,Unpack.004412D0      ASCII "Crackers For Freedom CrackMe v3.0"
00441298  . E8 17B8FFFF CALL Unpack.0043CAB4
0044129D  . 8B0D 102D4400 MOV ECX,DWORD PTR DS:[442D10]  Unpack.00443830
004412A3  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
004412A8  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
004412AA  . 8B15 5C0C4400 MOV EDX,DWORD PTR DS:[440C5C]  Unpack.00440CA8
```

올바르게 Unpacking 이 된 것을 확인할 수 있다.

문제 해결

마우스 우클릭 -> Search for -> All referenced text strings 를 통해 **"Wrong Serial, try again"** 을 찾아 이동

```
00440F2F  . BA 14104400 MOV EDX,Unpack.00441014      ASCII "Registered User"
00440F34  . E8 F32BFCFF CALL Unpack.00403B2C
00440F39  . 75 51       JNZ SHORT Unpack.00440F8C
00440F3B  . 8D55 FC     LEA EDX,DWORD PTR SS:[EBP-4]
00440F3E  . 8B83 C8020000 MOV EAX,DWORD PTR DS:[EBX+2C8]
00440F44  . E8 D7FEFDFF CALL Unpack.00420E20
00440F49  . 8B45 FC     MOV EAX,DWORD PTR SS:[EBP-4]
00440F4C  . BA 2C104400 MOV EDX,Unpack.0044102C      ASCII "GFX-754-IER-954"
00440F51  . E8 D62BFCFF CALL Unpack.00403B2C
00440F56  . 75 1A       JNZ SHORT Unpack.00440F72
00440F58  . 6A 00       PUSH 0
00440F5A  . B9 3C104400 MOV ECX,Unpack.0044103C      ASCII "CrackMe cracked successfully"
00440F5F  . BA 5C104400 MOV EDX,Unpack.0044105C      ASCII "Congrats! You cracked this CrackMe!"
00440F64  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
00440F69  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
00440F6B  . E8 F8C0FFFF CALL Unpack.0043D068
00440F70  . EB 32       JMP SHORT Unpack.00440FA4
00440F72  > 6A 00       PUSH 0
00440F74  . B9 80104400 MOV ECX,Unpack.00441080      ASCII "Beggar off!"
00440F79  . BA 8C104400 MOV EDX,Unpack.0044108C      ASCII "Wrong Serial,try again!"
00440F7E  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
00440F83  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
00440F85  . E8 DEC0FFFF CALL Unpack.0043D068
00440F8A  . EB 18       JMP SHORT Unpack.00440FA4
00440F8C  > 6A 00       PUSH 0
00440F8E  . B9 80104400 MOV ECX,Unpack.00441080      ASCII "Beggar off!"
00440F93  . BA 8C104400 MOV EDX,Unpack.0044108C      ASCII "Wrong Serial,try again!"
00440F98  . A1 442C4400 MOV EAX,DWORD PTR DS:[442C44]
00440F9D  . 8B00        MOV EAX,DWORD PTR DS:[EAX]
00440F9F  . E8 C4C0FFFF CALL Unpack.0043D068
```

비교문을 찾기 위해 스크롤을 위로 올린 결과, **Crack 성공에 관련된 문구와 User 와 Serial Key 로 보이는 문구**를 찾을 수 있었다.

- User 가 맞지 않은 경우, **00440F8C 로 JMP** 하게되어 **"Wrong Serial,try again!"** 을 출력
- KEY 가 맞지 않은 경우, **00440F72 로 JMP** 하게되어 **"Wrong Serial,try again!"** 을 출력

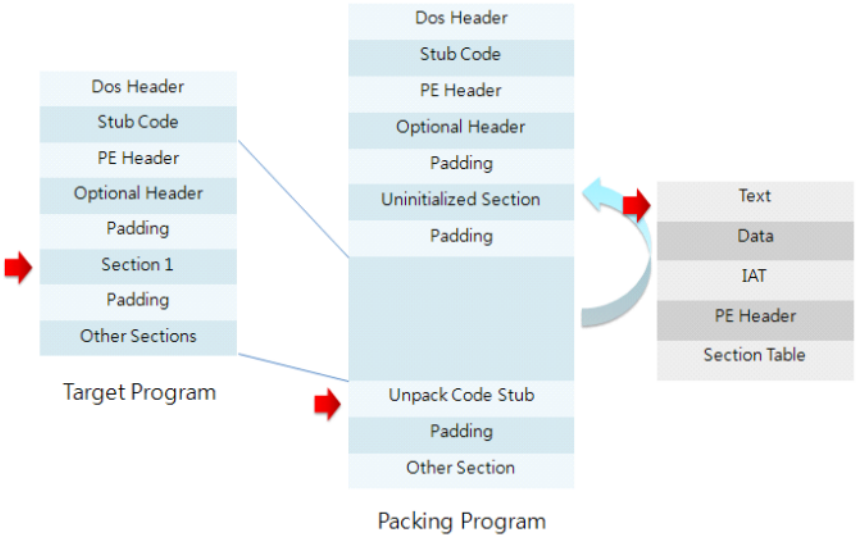
※ 여기서 0043D068 은 ASCII 문구 후에 다 있으므로 MessageBox 로 추측, 00440FA4는 빠져나가는 구문이라고 추측할 수 있다.

User : **Registered User** , KEY : **GFX-754-IER-954** 를 입력하여 실행 해보았다.

성공 문구가 적혀진 MessageBox 가 출력 되었다.

보충 설명

Packing



실행 파일의 **크기 감소, 자사의 기술 보호, 리버싱을 통한 분석 방지**를 위해 수행하는 작업

- 압축, 암호화를 통해 코드를 가공하여 **새로운 실행 파일을 하나 만든다.**
  - 수행하는 **Programmer에 따라서 각기 다른 방법으로 Packing**이 이루어진다.
- 새로 만드는 실행 파일의 **내용을 데이터 형태로** 집어 넣게 된다.

Packing이 된 프로그램의 **원본 기능은 그대로 유지**된다.

- 가공을 하여 Data형태로 집어 넣었지만 **결과적으로 실행이 되어야 한다**.
- **원상태로 돌리기 위한 Code가 필요하다.(Unpack Code)**

※ **보통 Code부분을 비워두고 원본이 들어갈 위치를 만들어 Code와 Data를 넣어 packing 된다.**

Unpacking 방법

Code 내용을 그대로 분석	Packing된 파일은 Code에 **압축을 해제하는 Algorithm**이 들어 있을 것이다. ○ **암호화, 압축 Algorithm을 분석하는 것은 오랜시간이 걸리고 비효율 적이다.** ○ 분석 한다고 하더라도 **해당 Packer의 Algorithm만 알게 되기 때문에 정적인 분석방법은 쓰지 않는다.**
실행시 압축 해제 시점을 찾아 분석	**CPU는 압축된 데이터는 해석 할 수 없기 때문에 프로그램이 수행되면 메모리상에 압축이 풀려야 한다.** ○ 압축이 풀렸기 때문에 **Code가 실행 될 것이다.** ○ Packing된 프로그램이 **실행이 될때 압축을 풀어 놓은 시점을 찾으면 Unpacking때의 시작 지점을 찾을 수 있다.** ○ Packing 프로그램은 **원본 프로그램을 최대한 유지 시켜줌**으로 정상적으로 동작할 수 있는 상황을 만들어준다.

답

GFX-754-IER-954