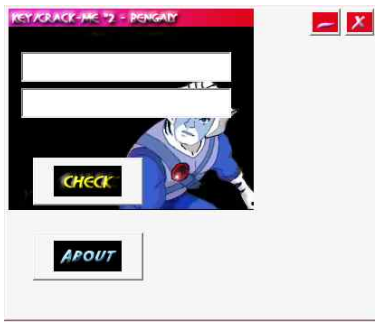
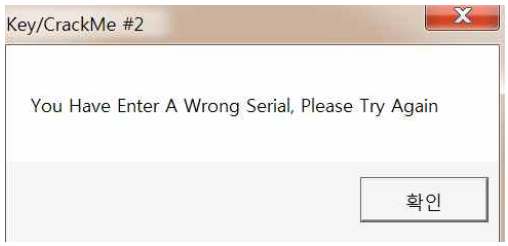
코드 엔진 Challenges: Basic 14

Author: BENGALY
Korean: Name이 CodeEngn일 때 Serial을 구하시오
(이 문제는 정답이 여러개 나올 수 있는 문제이며 5개의 숫자로 되어있는 정답을 찾아야 함,BruteForce필요)
Ex)11111

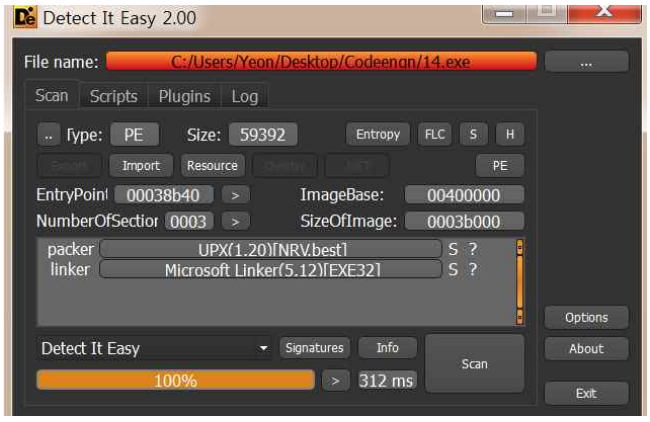
시리얼값을 생성하는 알고리즘을 찾아서 분석하는게 문제의 키이다. 문제를 통해서 얻을 수 있는 약간의 정보는 이름에 따라서 시리얼이 바뀐다는 것이다.

문제를 확인했으니 파일을 다운로드 받아서 실행해보자.





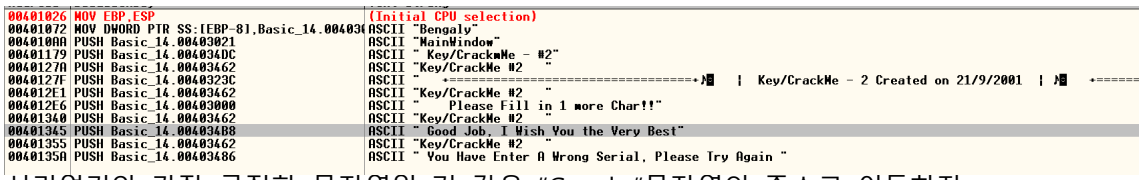
위와 같은 화면이 나와서 아무값이나 넣어보니 "You~"이러한 문자가 나왔다.
아직은 어떠한 힌트도 얻지못하였다. DE를 통해서 패킹여부를 확인해보자.



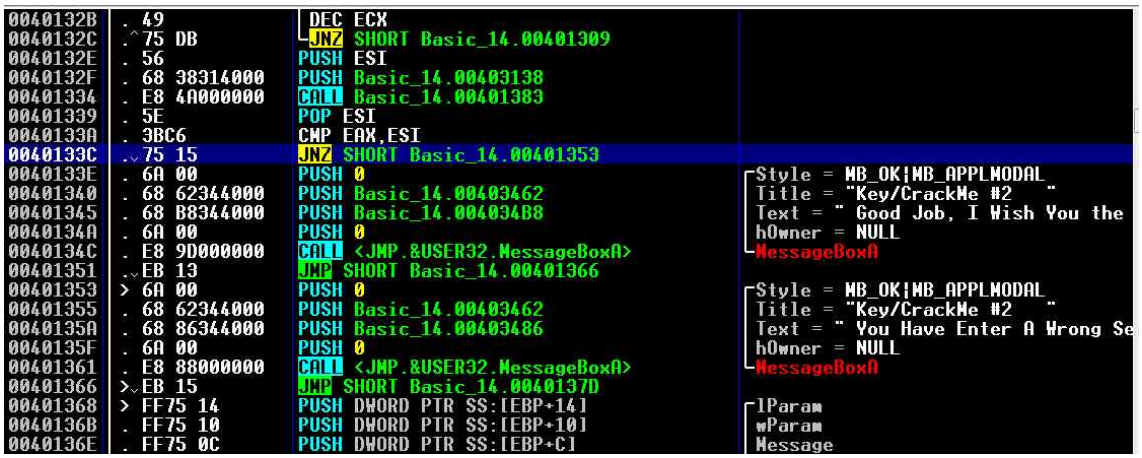
DE로 파일을 열어보니 이 파일이 UPX로 패킹되어있다는 것을 알 수 있다.
그리고 올리디버거로 열어보니 따로 StolenByte가 된건 아닌거 같다.

UPX를 통해 언패킹한후 올리디버거로 파일을 열어분석해보자.
(언패킹은 많이 해봤으니 생략)

이 프로그램은 이름값을 입력받고 시리얼값을 생성하기 때문에 어느 한 주소에 시리얼값이 저장되어있진 않을 것이다. 사용자 입력값을 가져와서 사용하는 방식이라는 것을 예상할 수 있을 것이다. 시리얼 키를 찾기위해 문자열들을 확인해보자.



시리얼키와 가장 근접한 문자열일 거 같은 "Good~"문자열의 주소로 이동하자.



위 코드를 잘 보면 JMZ라는 분기문이 보인다. 00401353으로 가보니 시리얼값이 틀렸을때 나오는 문자열을 띄우는 함수가 보인다. 분기문의 전을 살펴보면 EAX와 ESI의 값을 비교함을 알 수 있다. 비교하는 EAX와 ESI값은 어떤 값이 들어간지 알기위해 분기문 근처의 함수에 BP를 걸고 진행해보자.

분기문의 근처함수는 lstrlenA 함수이고 이 함수는 문자열의 길이를 반환해준다. 함수에 대한 반환값은 EAX에 저장되는 것을 보면 시리얼의 길이를 비교하는 것을 예상 할 수 있다. 확인해보기위해 실행을 해보자.

#lstrlenA 함수
//지정된 문자열의 길이를 결정합니다 (종료 널 문자 제외).

int lstrlenA(
  LPCSTR lpString
);
매개 변수
-lpString
=유형 : LPCTSTR
=검사 할 Null 종료 문자열입니다.
반환 값
=유형 : int
=이 함수는 문자열의 길이를 문자로 반환합니다. lpStringNULL일 경우 , 함수는 0을 반환합니다.

예측했던대로 lstrlenA의 값은 EAX에 저장되는 것을 알 수 있다.
그 밑의 코드를 따라가면
004012F6 PUSH 403038 name 값 (codeengn)
004012F8 CALL 00401430 lstrlen 함수 호출 길이 값 EAX에 저장
00401300 XOR ESI,ESI ESI 0 으로 초기화
00401302 MOV ECX,EAX ECX 에 EAX 값 저장
00401304 MOV EAX,1 EAX 값에 1 저장
00401309 MOV EDX,DWORD PTR DS:[403038]
EDX 에 첫번째 부터 4번쨔 문자열이 저장
0040130F MOV DL,BYTE PTR DS:[EAX+403037]
DL 에 첫번째 문자를 저장
00401315 AND EDX,0FF EDX 에 0FF 를 AND 시켜서 EDX 에 DL 많이 남도록 함
0040131B MOV EBX,EDX EDX 값을 EBX에 저장
0040131D IMUL EBX,EDX EBX 값과 EDX 값을 곱한다. 즉 제곱 값을 EBX 에 저장
00401320 ADD ESI,EBX ESI 값에 EBX (제곱 값) 를 더한다.
00401322 MOV EBX,EDX EDX 값을 EBX에 저장한다. 첫글자를 EBX에 저장한다.
00401324 SAR EBX,1 쉬프트 연산을 진행한다. (S : 쉬프트 A : 산술 R : 우측)
00401326 ADD ESI,EBX ESI 값에 EBX 값을 더한다. (제곱값 + 첫글자 쉬프트 연산값)

00401328 SUB ESI,EDX ESI 값에서 다시 EDX 값 (첫글자) 을 빼준다.
0040132A INC EAX EAX 값에 1을 증가시킨다. 반복문 카운팅 용도
0040132B DEC ECX ECX 값에 1을 감소시킨다. 반복문 카운팅 용도
0040132C JNZ SHORT 00401309
ECX 값이 0이면 (문자의 길이 만큼 반복하면) 점프
0040132E PUSH ESI ESI 값을 PUSH
0040132F PUSH 403138 입력한 Serial 값을 PUSH
00401334 CALL 00401383 401383 (16진수로 변환하는 코드) 호출
00401339 POP ESI ESI 값을 POP
0040133A CMP EAX,ESI EAX 와 ESI 비교
0040133C JNZ SHORT 00401353 같으면 분기

*입력한 시리얼값이 입력되는 곳은 00403138이다.

이런식인데 우리가 찾고자하는 것은 성공 메시지가 나왔을때의 시리얼값이다.
결국 레지스터값을 확인해보게 되면 EAX는 사용자 입력값이며 그와 비교되는 값인 ESI의 10진수 변환한 값이 우리가 찾고자 한 값이다.

이때 ESI값은 00129A1이다. 십진수로 변환해보면 76193이다