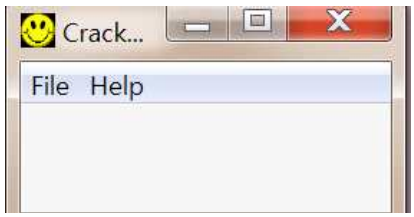
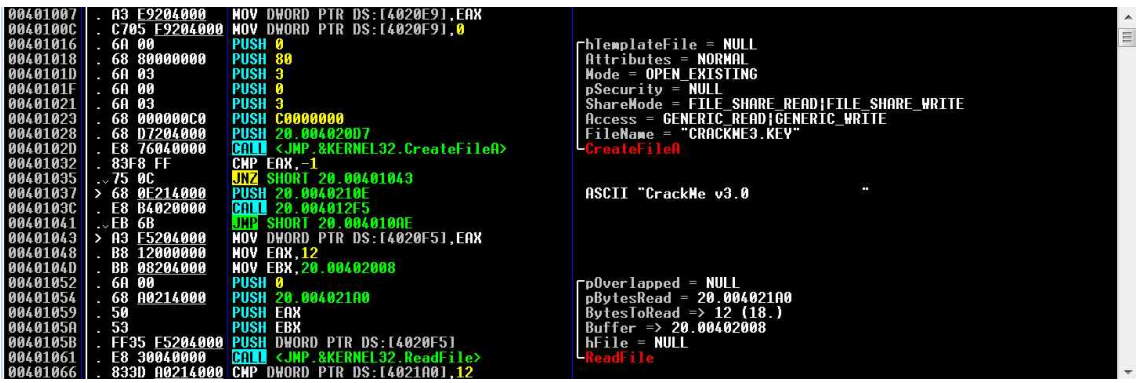
코드 엔진 Challenges: Basic 20

Author:Cruhead/MiB
Korean: 이 프로그램은 key 파일을 필요로 하는 프로그램이다.
위 문구가 출력되도록 하려면 crackme3.key 파일 안의 데이터는 무엇이되어야 하는가
EX)41424344454647



파일을 실행해보지 위와 같은 화면이 나오며 별 다른 기능은 없는 것처럼 보인다.
이 파일을 분석하기위해 DE로 패킹여부 분석 결과 패킹은 되어있지않고 어셈블리어로 코딩되어 있었다 .바로 올리디버거를 통해 분석해보자.



00401066까지는 CreateFileA함수를 통해 CrackMe.KEY파일을 만들고 18바이트를 채워 만들어 놓은 과정을 진행해야 한다. 그래서 18바이트 크기의 CRACKME3.KEY파일을 만들어보자.

파일을 작성하고 ReadFile 함수를 거치게되면 아래와 같이 00402008~004020019 부분에 CRACKME3.KEY 파일을 읽어들인다.

#CreateFile 함수
파일 혹은 오브젝트를 생성하거나 열 수 있는 함수입니다.
핸들을 받아와 다른 함수들을 이용해서 열기나 쓰기 등을 할 수 있습니다.
예를 들어 ReadFile 함수를 사용하여 파일을 읽을 수 있고 WriteFile 함수를 사용하여 파일을 쓸 수 있습니다.

-함수 원형

```
HANDLE CreateFile(
        LPCTSTR lpFileName,
        DWORD dwDesiredAccess,
        DWORD dwShareMode,
        LPSECURITY_ATTRIBUTES lpSecurityAttributes,
        DWORD dwCreationDisposition,
        DWORD dwFlagsAndAttributes,
        HANDLE hTemplateFile);
```

-인수
lpFileName
파일을 생성하거나 열 경로입니다.
dwDesiredAccess
파일을 열거나 쓰기 등을 할 때의 엑세스 권한을 지정합니다.
어떠한 목적으로 사용하냐에 따라 권한을 지정해야 합니다.
플래그를 중복해서 지정할 수 있습니다.

값	플래그	설명
0x10000000	GENERIC_ALL	모든 엑세스 권한을 가집니다.
0x20000000	GENERIC_EXECUTE	실행 권한을 가집니다.
0x40000000	GENERIC_READ	읽기 권한을 가집니다.
0x80000000	GENERIC_WRITE	쓰기 권한을 가집니다.
0x00010000	DELETE	삭제 권한을 가집니다.
0x00020000	READ_CONTROL	사용자, 그룹, 임의 액세스 제어 목록(DACL)이 읽기 권한을 가집니다.
0x00040000	WRITE_DAC	임의 액세스 제어 목록(DACL)이 읽기 권한을 가집니다.
0x00080000	WRITE_OWNER	사용자가 쓰기 권한을 가집니다.

0x00100000 SYNCHRONIZE 동기화 권한을 가집니다.

-dwShareMode

파일의 공유 모드를 지정합니다.

공유 모드를 지정할 경우 다른 프로세스에서 현재 파일을 엑세스할 수 있습니다.

하나의 플래그만 지정할 수 있습니다.

값	플래그	설명
0x00000000	0	모든 프로세스의 접근을 차단합니다.

다른 프로세스가 접근을 시도할 경우 엑세스가 거부됩니다.

0x00000001	FILE_SHARE_READ	다른 프로세스의 열기 권한을 허가합니다.
0x00000002	FILE_SHARE_WRITE	다른 프로세스의 쓰기 권한을 허가합니다.
0x00000004	FILE_SHARE_DELETE	다른 프로세스의 삭제 권한을 허가합니다.

-lpSecurityAttributes

SECURITY_ATTRIBUTES 구조체의 포인터입니다.

사용하지 않을 경우 NULL 값을 사용합니다.

-dwCreationDisposition

파일의 해당 위치에 존재하는지에 따른 행동입니다.

하나의 플래그만 지정할 수 있습니다.

값	플래그	설명
1	CREATE_NEW	파일이 존재하지 않을 경우 새로운 파일을 만듭니다.

파일이 존재할 경우 ERROR_FILE_EXISTS (80) 오류를 발생시킵니다.

2	CREATE_ALWAYS	항상 새로운 파일을 만듭니다.

파일이 존재할 경우 새로운 파일로 덮어씌웁니다.

3	OPEN_EXISTING	파일이 존재할 경우에만 파일을 엽니다.

파일이 존재하지 않을 경우 ERROR_FILE_NOT_FOUND (2) 오류를 발생시킵니다.

4	OPEN_ALWAYS	무조건 파일을 엽니다.

파일이 존재하지 않을 경우 파일을 새로 만들고 엽니다.

5	TRUNCATE_EXISTING	

파일이 존재할 경우 파일을 연 후 크기를 0으로 만듭니다.

파일이 존재하지 않을 경우 ERROR_FILE_NOT_FOUND (2) 오류를 발생시킵니다.

불러오는 프로세스는 dwDesiredAccess 인수의 GENERIC_WRITE 플래그가 있어야 합니다.

-dwFlagsAndAttributes

생성될 파일의 속성을 지정합니다.

플래그를 중복해서 지정할 수 있습니다.

값	플래그	설명
1	FILE_ATTRIBUTE_READONLY	파일을 읽기 전용으로 지정합니다.
2	FILE_ATTRIBUTE_HIDDEN	파일을 숨김으로 지정합니다.
4	FILE_ATTRIBUTE_SYSTEM	파일을 운영체제 전용으로 지정합니다.
32	FILE_ATTRIBUTE_ARCHIVE	파일을 보관 가능으로 지정합니다.

이 파일이 백업될 필요성이 있다는 것을 알립니다.

128	FILE_ATTRIBUTE_NORMAL	모든 속성을 지정하지 않습니다.

이 플래그만 사용될때 유효하며 다른 플래그와 같이 사용할시 이 플래그는 무시됩니다.

256 FILE_ATTRIBUTE_TEMPORARY 파일을 임시 파일로 지정합니다.
4096 FILE_ATTRIBUTE_OFFLINE
The data of a file is not immediately available.
This attribute indicates that file data is physically moved to offline storage.
This attribute is used by Remote Storage, the hierarchical storage management software.
응용프로그램이 이 속성을 변경해서는 안됩니다.
16384 FILE_ATTRIBUTE_ENCRYPTED
파일이나 폴더를 암호화로 지정합니다.
파일의 경우 모든 내용을 암호화하며 폴더의 경우 새로 만들어지는 파일을 암호화합니다.
FILE_ATTRIBUTE_SYSTEM 플래그가 지정되어 있는 경우 이 플래그는 무시됩니다.
이 플래그는 Windows Home, Home Premium, Starter, ARM에서 지원되지 않습니다.
-hTemplateFile
GENERIC_READ 엑세스 권한을 가진 템플릿 파일의 유효한 핸들입니다.
생성된 파일에 대한 속성을 제공하는 템플릿입니다.
사용하지 않을 경우 NULL 값을 사용합니다.

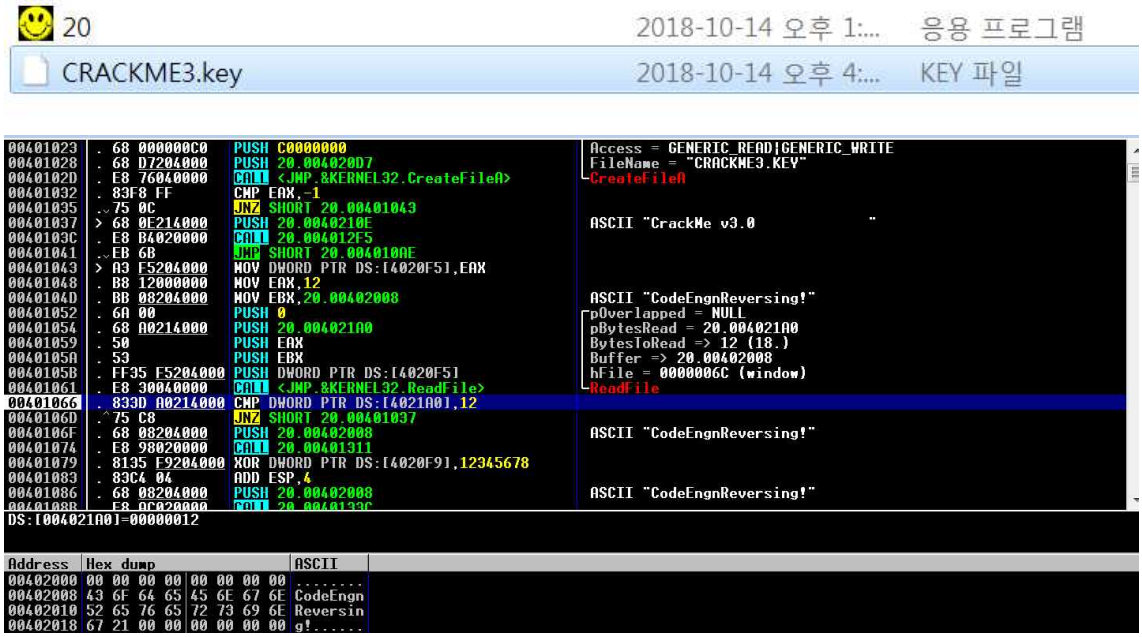
#ReadFile 함수
파일 데이터를 읽는 함수입니다.
이 함수를 쓰기 전에 먼저 CreateFile 함수를 사용해서 파일의 핸들을 받아와야 합니다.
반대로 파일 데이터를 쓰는 함수로는 WriteFile 함수가 있습니다.

-함수 원형
```
BOOL ReadFile(
    HANDLE hFile,
    LPVOID lpBuffer,
    DWORD nNumberOfBytesToRead,
    LPDWORD lpNumberOfBytesRead,
    LPOVERLAPPED lpOverlapped);
```

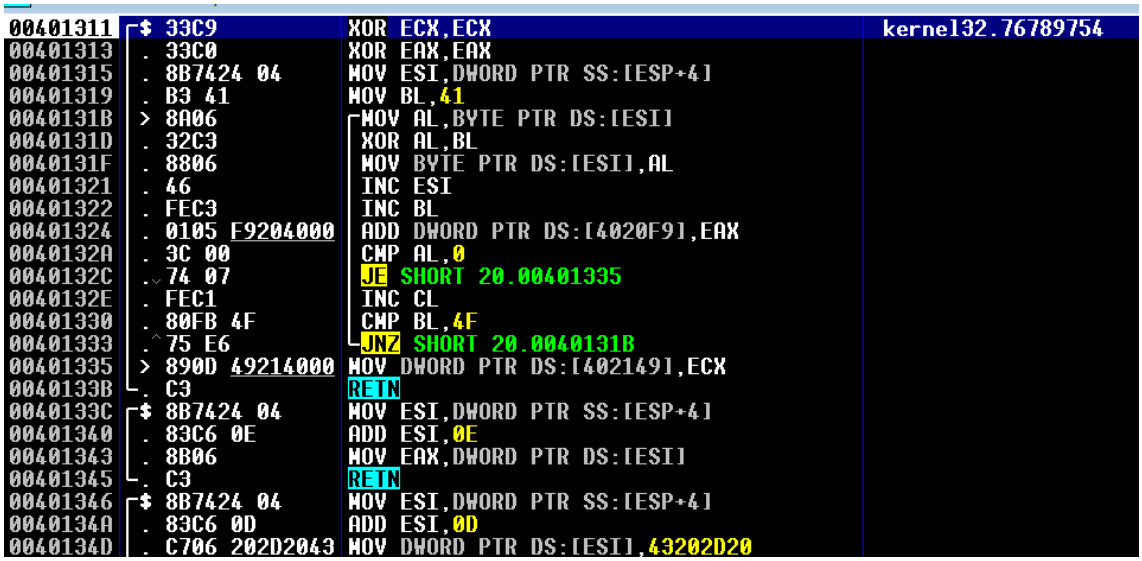
-인수
hFile
파일이나 장치의 핸들입니다.
반드시 읽기 권한이 있어야 합니다.
비동기식 읽기 작업을 하려면 CreateFile 함수를 사용해서 FILE_FLAG_OVERLAPPED 플래그를
지정하거나 socket 함수나 accept 함수를 사용해서 받아온 핸들이어야 합니다.
lpBuffer
파일이나 장치로부터 읽은 데이터를 받아올 버퍼의 포인터입니다.
nNumberOfBytesToRead
읽어들일 데이터의 길이입니다.
당연하게도 읽을 데이터의 길이는 파일의 길이보다 클 수 없습니다.
lpNumberOfBytesRead

읽어들인 데이터 바이트의 수를 리턴받는 인수입니다.
이 인수는 함수가 호출되었을 때 0으로 초기화됩니다.
lpOverlapped 인수가 NULL 값을 가지지 않으면 이 인수는 NULL로 지정해야 합니다.
lpOverlapped
비동기 입출력을 위한 OVERLAPPED 구조체의 포인터입니다.
만약 파일을 FILE_FLAG_OVERLAPPED 플래그를 지정해 열었다면 OVERLAPPED 구조체를 반드시 지정해야 합니다.
비동기 입출력을 사용하지 않을 경우에는 NULL 값으로 지정해야 합니다.

[출처] ReadFile 함수|작성자 히니즈



위와 같이 CRACKME3.key 파일을 읽어 들인 후 00401074 주소의 명령을 살펴보면 00301311을 호출하며 해당 호출 주소로 따라가 본 결과는 아래와 같다



-00401311 ECX를 0으로 만듬

-00402424 EAX를 0으로 만듬

-00401315 ESP+4의 주소값을 ESI에 복사 ->CRACKME3.key 파일의 내용이 적힌 주소 값을 ESI 에 복사

-00401319 41을 EBX의 최하위 1바이트에 복사

-0040131B ESI의 주소값에 담긴 최하위 1바이트를 EAX에 복사

-0040131D EAX와 EBX 최하위 1바이트를 XOR 연산 결과를 EAX에 복사(EAX=00000002)

-0040131F EAX의 최하위 1바이트를 ESI 최하위 바이트에 복사 (ESI 00402008 =>"od")

-00401321 ESI값을 1증가

-00401322 EBX의 최하위 1바이트 값 증가

-00401324 EAX 값을 4020F9에 복사

-0040132A EAX의 최하위 1바이트를 0과 비교

-0040132C 0040132A의 명령이 참이라면 00401335로 분기

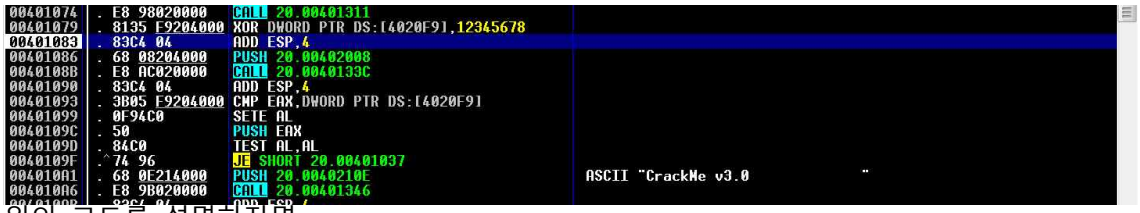
-0040132E ECX의 최하위 1바이트 값을 증가

-00401330 EBX의 최하위 1바이트와 0x4F 비교

```
BL=42 ('B')
```

-00401333 00401330의 명령이 거짓이라면 0040131B로 분기
-00401335 ECX의 값을 402149로 복사
0040133B 리턴

위 루틴은 결국 반복문이 진행되는 동안 문자열 1 바이트와 0x41~0x4F까지 XOR 한 결과 값인 AL을 4020F9에 계속 더한다. 0x41~0x4F의 길이는 14이며 ,CRACKME.KEY 의 14 글자는 XOR연산에 쓰이지만 나머지 4글자는 쓰이지 않음을 의미한다. 정리하자면 결국 14글자가 키 값을 생성하는 알고리즘에 사용되는 것이고 나머지 4글자가 키에 해당한다. 다시 코드로 돌아가서



위의 코드를 설명하자면
-00401074의 XOR연산 결과 값이 담긴 주소의 값이랑 12345678이랑 XOR연산
-0040108B에서 CALL 0040133C의 명령을 통해 XOR연산에 쓰이지 않은 나머지 4글자를 가져와 EAX에 반환.
-00401093의 비교 구문을 통해 EAX와 4020F9를 비교하고
-00401099의 SETE AL 명령을 통해 위 조건문의 결과가 참일 경우 1을 반환하며 아닐 시 보수를 AL 사이즈에 맞게 변환
-00401093의 비교구문의 거짓을 의미하는 보수값이 AL에 담기면 0040109F의 명령을 통해 실패로 분기
-참을의미하는 값 1이 담긴다면 004010A6의 CALL00401346의 명령을 통해 크랙 성공 메시지가 출력됨