기술문서	10. 2. 13. 작성

PADOCON 2010 CTF 예선
CatchMe 풀이

by 6l4ck3y3
6l4ck3y3@gmail.com

Abstract

PADOCON 2010 CTF 예선에서 풀었던 문제, CatchMe에 대한 풀이입니다. CatchMe는 Windows 바이너리 파일에 대한 리버싱 유형의 문제입니다. 이 문서를 이해하기 위해서는 어셈블리어와 리버싱에 대한 기초지식이 필요합니다.

* 본 문서는 대학정보보호동아리연합회(http://www.kucis.org)의 지원을 얻어 작성된 문서입니다.

Table of Content

1. Scenario

잡으실수 있으시면
잡으셔도 됩니다.

If you really catch him,
You should do that.

CatchMeIfYouCan.exe (32.0 KB)

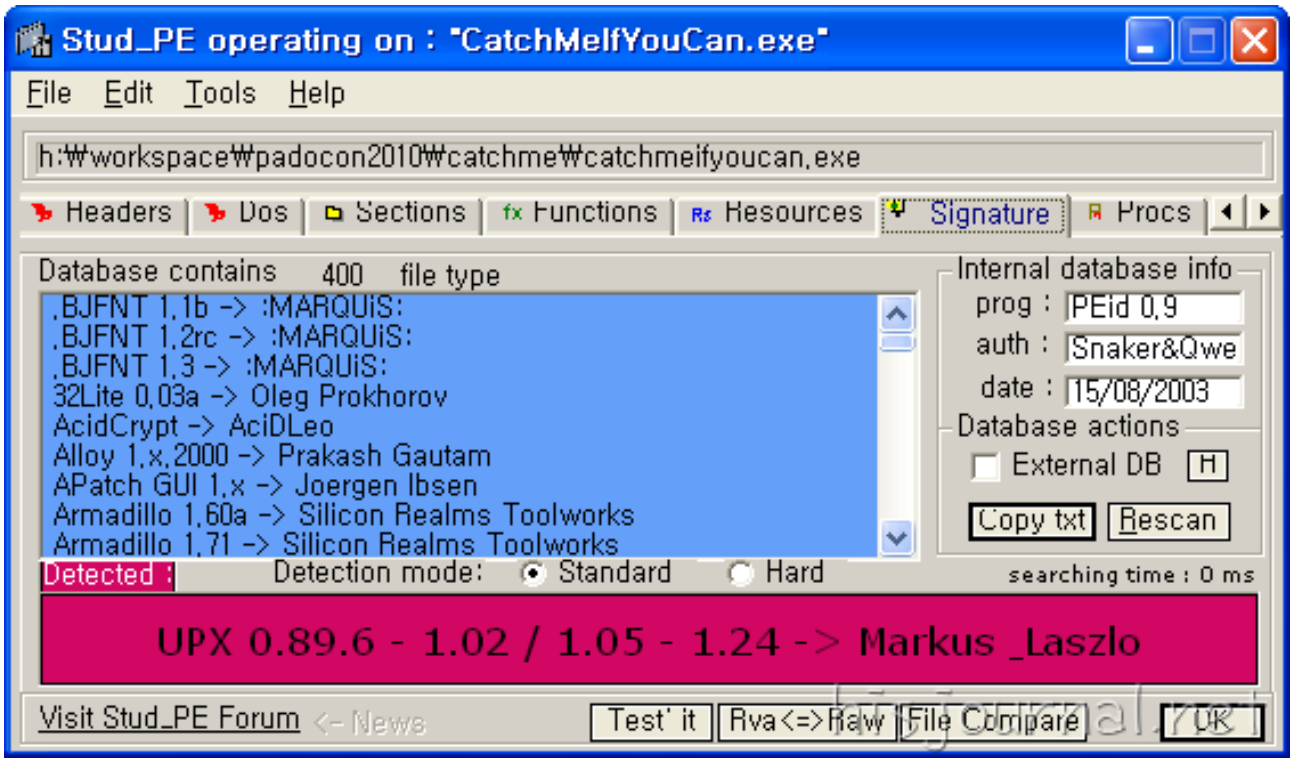
2. Solution

2-1. 문제 분석

파일 제목 그대로 Click 박스를 클릭하면 패스워드가 출력되는 문제입니다.

2-2. 리버싱

리버싱에서 가장 먼저 하는 작업이 문자열을 추출하는 것이지만, 이 문제에서는 아무런 소용이 없었습니다. 문자열을 추출해도 단편적으로 쪼개져있는 의미를 알 수 없는 문자열들만 나왔기 때문이죠. 이것을 통해 이 바이너리 파일이 패킹되었다는 것을 유추할 수 있습니다.

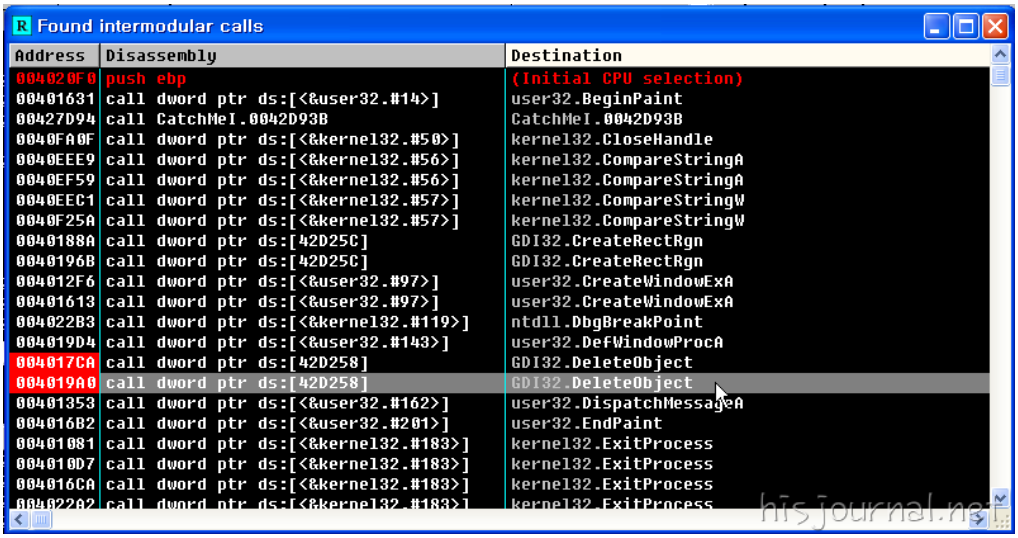


패커의 종류를 확인하기 위해서 PeiD, Stud_PE 같은 툴을 사용합니다. CatchMEIfYouCan.exe 는 UPX 라는 패커로 패킹되었음을 확인할 수 있습니다. UPX를 언패킹하는 방법은 문서로 많이 정리되어 있으므로 이 문서에서는 설명을 생략하겠습니다.

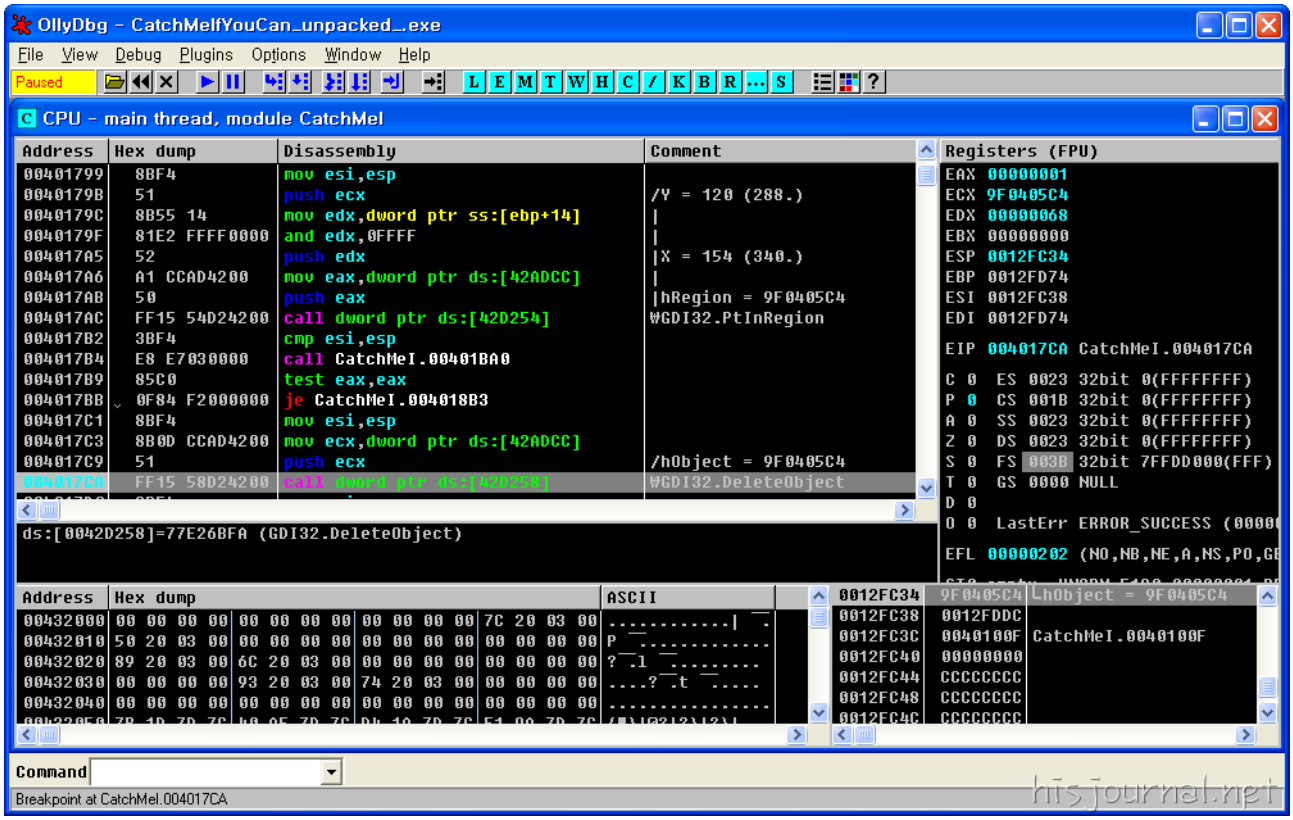
CatchMEIfYouCan.exe 를 수차례 실행해보면서 프로그램의 흐름을 우선적으로 생각해보았습니다. 무작정 바이너리 속을 분석하는 것보다 프로그램의 흐름을 파악하여 어디서부터 접근하는 게 좋을지 판단하는 게 더 쉽기 때문이죠.

이 프로그램은 Click 박스에 마우스 커서를 가져가면 Click 박스가 이동을 하여서 클릭을 할 수 없게 합니다. 그렇다면, Click 박스가 이동하는 것은 어떤 루틴을 이용하는 것일까? 이런 의문을 가졌습니다. 보통 GUI 프로그래밍을 할 때 이런 UI를 구성할 때가 있습니다. 저는 이런 경우, 기존의 컴포넌트를 삭

제하고 새로운 컴포넌트를 생성하는 방법을 사용했었습니다.



그래서 OllyDbg에서 intermodular calls를 검색하여 DeleteObject 함수에 브레이크포인트를 걸었습니다. 혹은 반대로 CreateRectRgn 함수에 브레이크포인트를 걸어도 되지요. 그리고 실행하였습니다.

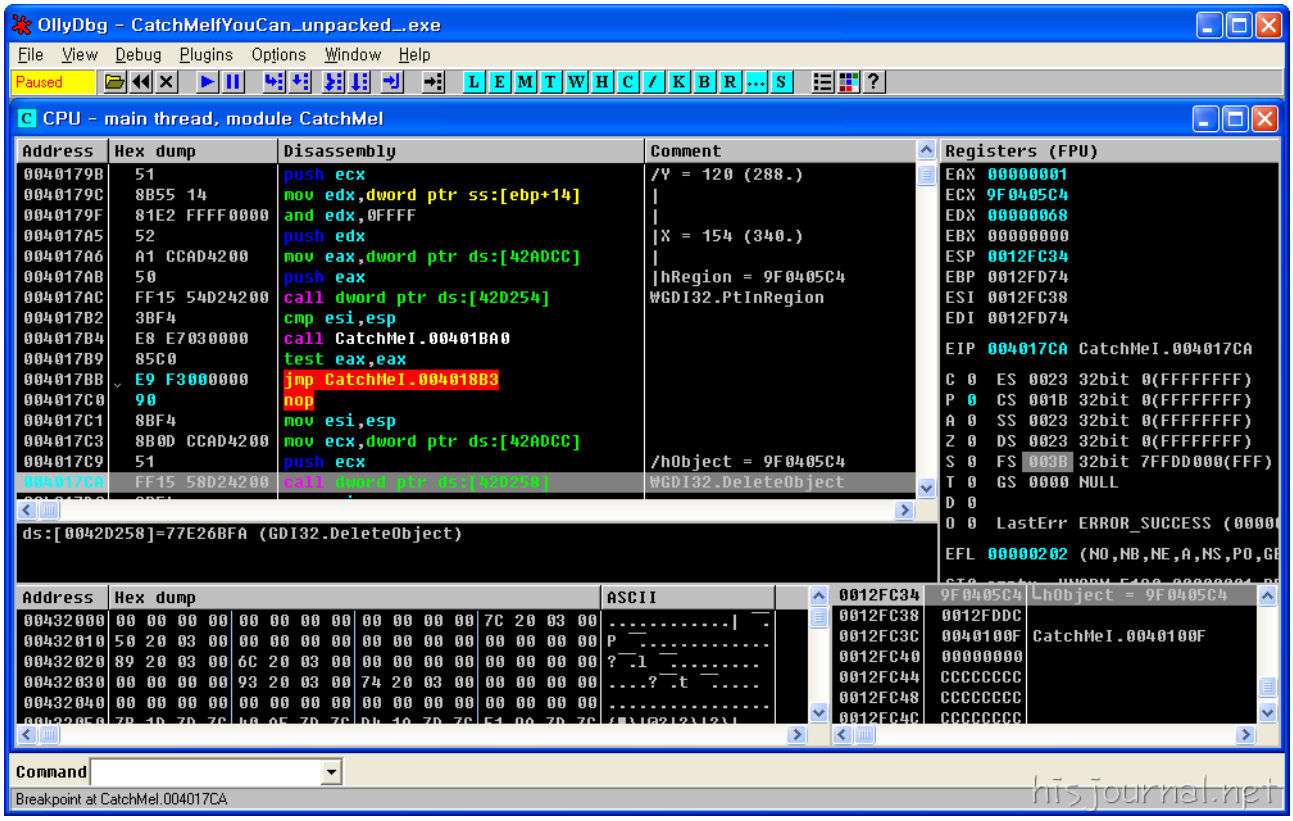


역시 마우스 커서를 Click 박스에 가져가는 순간, 일시정지 상태가 되고 EIP가 DeleteObject 함수를 호출하는 부분에서 멈추었습니다.

```
0040179B   51              push ecx                          ; /Y = 120 (288.)
0040179C   8B55 14         mov edx,dword ptr ss:[ebp+14]     ; |
0040179F   81E2 FFFF0000   and edx,0FFFF                     ; |
004017A5   52              push edx                          ; |X = 154 (340.)
004017A6   A1 CCAD4200     mov eax,dword ptr ds:[42ADCC]     ; |
004017AB   50              push eax                          ; |hRegion = 9F0405C4
004017AC   FF15 54D24200 c all dword ptr ds:[42D254]         ; \GDI32.PtInRegion
004017B2   3BF4            cmp esi,esp
004017B4   E8 E7030000     call CatchMeI.00401BA0
004017B9   85C0            test eax,eax
004017BB   0F84 F2000000   je CatchMeI.004018B3
004017C1   8BF4            mov esi,esp
004017C3   8B0D CCAD4200   mov ecx,dword ptr ds:[42ADCC]
004017C9   51              push ecx                          ; /hObject = 9F0405C4
004017CA   FF15 58D24200   call dword ptr ds:[42D258]        ; \GDI32.DeleteObject
```

DeleteObject 함수의 위 쪽을 살펴보면, PtInRegion 함수가 호출되고 분기문이 있는 것을 볼 수 있습니다. 이 함수는 특정 영역 안에 포인트가 있는지 없는지 확인하는 함수입니다. 만약 정해진 공간 안에 포인트가 위치한다면 0이 아닌 값을 반환하고, 영역 밖이라면 0을 반환합니다. 분기문에서는 함수의 반환값이 0이 아니면 아래의 DeleteObject 함수로 흐름이 넘어갑니다. 물론, DeleteObject 함수는 Click 박스의 핸들을 인자로 가집니다.

이로소 CatchMEIfYouCan.exe 의 흐름이 파악되었고, 어떻게 하면 문제를 해결할 수 있는지도 알았습니다. PtInRegion 함수의 결과에 의해 DeleteObject 함수로 실행 흐름이 넘어갈지 아닐지 결정되기 때문에, 이 흐름을 무조건 DeleteObject 함수로 안 넘어가도록 바이너리를 수정하면 Click 박스가 더 이상 움직이지 않을 것입니다.



분기문 je 를 jmp 로 수정합니다. 바이트 수가 달라지지만, 다행이도 모자라는 게 아니라 남습니다. 남은 1 바이트는 nop 으로 채웁니다. 그리고 다시 실행하면, 패스워드가 나옵니다.

Reference

Intel Instruction Set pages [링크]

UnPacking UPX0.89(Manual Unpack : MUP) [링크]

DeleteObject Function [링크]

PtInRegion Function [링크]