Advance RCE L09

2010년 10월 19일 화요일
오전 3:42

CodeEngn Advanced RCE L09

L3m0nTr33
L3m0nTr33.sur3x5f.org

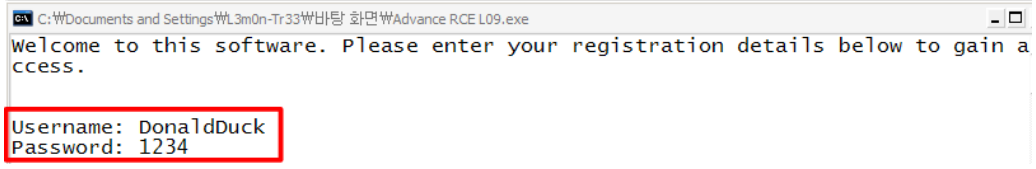
1. Introduction

Advance RCE L09 문제는 **Username 과, Password 를 입력하여 성공 또는 실패 구문을 출력**하는 Program 이다. Debugger 로 열어보면, " **ThisIsTheUserName : DonaldDuck**" 이라는 글이 Dump 창에 보이고, 이를 통해 Username 을 알 수 있다.

```
Address   Hex dump                                        ASCII
00404000  84 75 01 37 7B 8A FE C8 FF FF FF FF FF FF FF FF  ?u.7{?쳗ÿÿÿÿÿÿÿÿ
00404010  FE FF FF FF 01 00 00 00 08 32 40 00 F8 31 40 00  ?ÿÿÿ....2@.?@.
00404020  52 61 6E 64 6F 6D 47 69 72 6C 00 00 00 00 00 00  RandomGirl......
00404030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00404040  4B 69 6E 67 4B 6F 6E 67 00 00 00 00 00 00 00 00  KingKong........
00404050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00404060  46 61 74 61 4D 6F 67 61 6E 61 00 00 00 00 00 00  FataMogana......
00404070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00404080  54 68 69 73 49 73 54 68 65 55 73 65 72 6E 61 6D  ThisIsTheUsernam
00404090  65 3A 20 44 6F 6E 61 6C 64 44 75 63 6B 00 00 00  e: DonaldDuck...
```

Username : **DonaldDuck**, Password : **1234 (0x4D2) 를 입력**하여 Program을 실행 하였을 때, Password 값과 실제 유효 Password 값을 비교하는 곳을 찾을 수 있었다.

```
00401048  . 3B01          CMP EAX,DWORD PTR DS:[ECX]
DS:[003C02A0]=0088228F
EAX=000004D2
```



2. Analyze

ECX 의 값인 003C02A0 을 주소로 하여 안의 값인 **0x0088228F (8921743) 을 구하는 과정**을 찾아 보았다.

```
Address   Hex dump     Disassembly                          Comment
00401328  . 6A 04      PUSH 4                               Protect = PAGE_READWRITE
0040132A  . 68 00300000 PUSH 3000                           AllocationType = MEM_COMMIT|MEM_RESERVE
0040132F  . 68 00010000 PUSH 100                            Size = 100 (256.)
00401334  . 6A 00      PUSH 0                               Address = NULL
00401336  . FF15 00304000 CALL DWORD PTR DS:[<&KERNEL32.VirtualAl VirtualAlloc
0040133C  . 33C9       XOR ECX,ECX
0040133E  . 8BFF       MOV EDI,EDI
00401340  > C70488 8F2288 MOV DWORD PTR DS:[EAX+ECX*4],88228F
00401347  . 83C1 04    ADD ECX,4
0040134A  . 81F9 00010000 CMP ECX,100
00401350  .^72 EE      JB SHORT Advance_.00401340
00401352  . 05 A0020000 ADD EAX,2A0
00401357  . A3 04444000 MOV DWORD PTR DS:[404404],EAX
0040135C  . 0FB605 383140 MOVZX EAX,BYTE PTR DS:[403138]
00401363  . 84C0       TEST AL,AL
00401365  . 74 20      JE SHORT Advance_.00401387
00401367  . BE 38314000 MOV ESI,Advance_.00403138           ASCII "Welcome to this software. Please
0040136C  . 8D6424 00  LEA ESP,DWORD PTR SS:[ESP]
00401370  > 50         PUSH EAX
00401371  . A1 78304000 MOV EAX,DWORD PTR DS:[<&MSVCP90.?cout@
00401376  . 50         PUSH EAX
00401377  . E8 04020000 CALL Advance_.00401580
0040137C  . 8A46 01    MOV AL,BYTE PTR DS:[ESI+1]
EAX=003C0000, (ASCII "??")
```

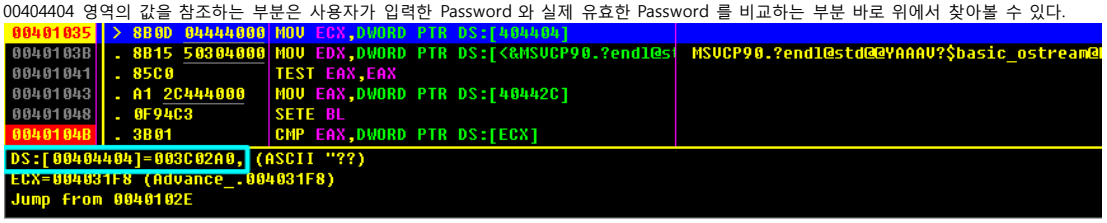
```
Address   Hex dump                                        ASCII
003C0240  8F 22 88 00 00 00 00 00 00 00 00 00 00 00 00 00  ??..............
003C0250  8F 22 88 00 00 00 00 00 00 00 00 00 00 00 00 00  ??..............
003C0260  8F 22 88 00 00 00 00 00 00 00 00 00 00 00 00 00  ??..............
```

- 분석

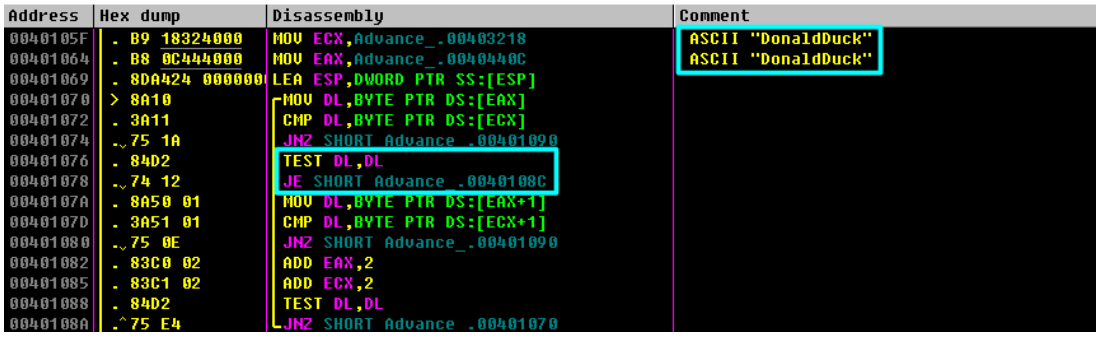
  우선 VirtualAlloc 함수로 256 size 의 동적할당을 실행한 후, EAX (003C0000) + (ECX (첫번째 Loop 에는 0) * 4) 의 주소에 88228F 를 넣는다.
  ECX 를 4씩 증가시키며 ECX 값이 100 이 될 때 까지 해당 Loop 를 수행한다.
  EAX (003C0000) 의 값에 2A0 을 더한 후, 00404404 에 해당 값 (003C02A0) 을 복사한다.
  이러한 과정을 한 후에, " Welcome to this software. ~ " 구문을 출력하며 Program 을 시작한다.

- 정리하면 **256 만큼 동적할당한 Heap 영역에 003C0000 에서 003C03F0 까지 0x0088228F (8921743) 라는 값을 넣고, 이 범위중 하나의 주소를 00404404 에 넣는다.**
  - 이는 **Exploit 기법 중 하나인 Heap Spray 기법의 원리와 비슷**하다고 생각된다.

00404404 영역의 값을 참조하는 부분은 사용자가 입력한 Password 와 실제 유효한 Password 를 비교하는 부분 바로 위에서 찾아볼 수 있다.



Trace 를 하며 아래로 내려가 보면 **" DonaldDuck " 이라는 Username 이 맞는지 확인하는 Loop** 도 찾을 수 있다.

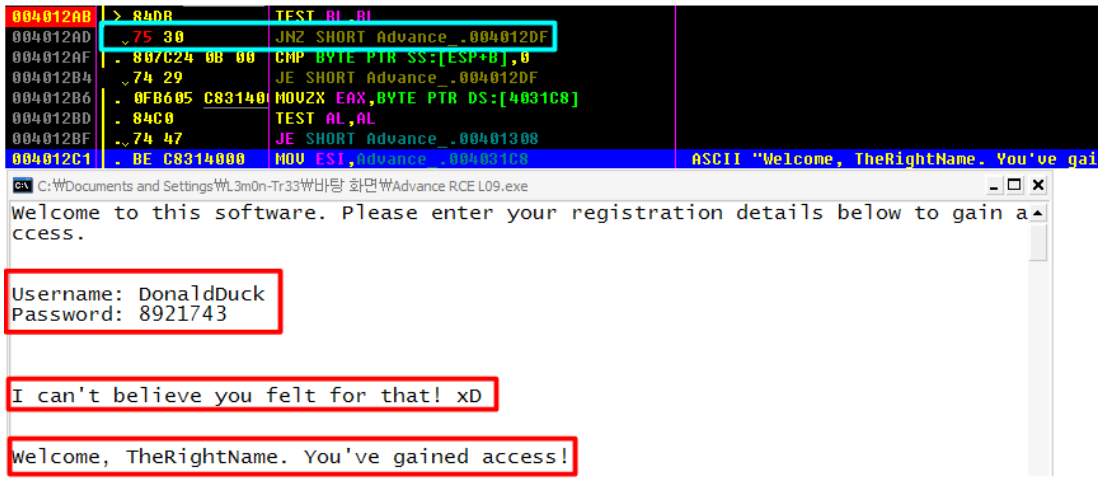


- 분석
  0040440C (사용자의 입력) 값중 앞의 1Byte 를 00403218 에 있는 값중 앞의 1Byte 와 비교하여 같으면 Loop 수행 , 다르면 빠져나온다.
  사용자가 입력한 값이 있는 지 없는지 검사하는 TEST DL, DL 을 거친 후, 입력이 끝나면 해당 Loop 를 빠져나오고, 다르면 Loop 를 수행한다.
  1Byte 다음 값을 읽어들여서 해당 비교하여 같으면 Loop 수행, 다르면 빠져나온다.
  EAX 와 ECX (사용자의 입력값과, 원래 값) 을 각각 2씩 증가 시킨 후, DL 이 0이면 해당 Loop 를 빠져나오고, 0이 아니면 Loop 를 수행한다.

- 비교한 값이 틀려서가 아닌, 문자열이 끝났을 경우 0040108C 로 JMP 하여, EAX (사용자의 입력) register 를 XOR 로 0 으로 만든 후, TEST EAX, EAX 를 통해 0 인지 확인한 후, 아래의 Code 를 수행한다. 여기서 아래의 Code 는 **" I can't believe you felt for that! xD " 를 출력하는 Code** 이다.

- 계속 Trace 해 보면, **TEST BL, BL 후 JE 004012DF Code (에러 출력)**가 보인다.
  - 그러나 **BL 은 Program 실행 시에 XOR EBX, EBX 로 0으로 초기화 된 후 한번도 값이 들어가지 않았다.**
    그러므로 이는 무조건 JMP 하게 되었다.
  - 반면, **ESP+B 인 0012FF6B 이 경우 Password 가 맞을 경우 1 로 Setting** 이 되어 올바르게 작동한다.

해당 문제가 직접 Code 수정을 요하는 문제인것 같아서, Code 를 수정하여 성공 구문을 출력 하였다.



3.Conclusion

해당 문제는 **HEX Dump 창**을 볼 수 있는지, **Heap 영역에 대한 기초적인 개념 및 Heap Spray 기법** 에 대한 약간의 소개, **Code 수동 수정** 등을 알 수 있는 좋은 문제였다. 마지막에 Code 를 수동으로 수정하는것이 확실하지 않지만 **CodeEngn 에서 요하는 Password 를 구하는 데는 아무 지장이 없으므로 임의로 수정**하였다.

답 : 8921743