

## Advance RCE L09

2010년 10월 19일 화요일

오전 3:42

# CodeEngn Advanced RCE L09

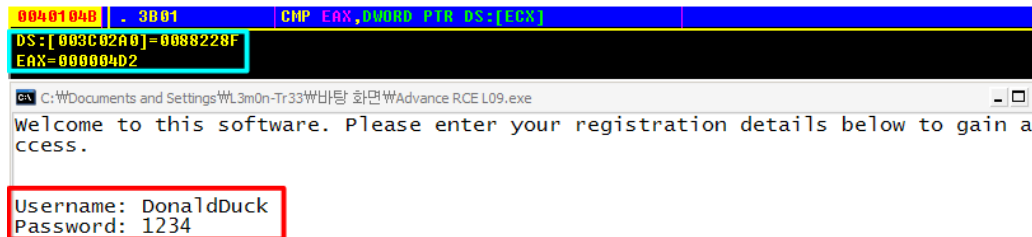
L3m0nTr33  
L3m0nTr33.sur3x5f.org

## 1. Introduction

Advance RCE L09 문제는 Username 과, Password 를 입력하여 성공 또는 실패 구문을 출력하는 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	??...
00404010	FE FF FF FF 01 00 00 00 08 32 40 00 F8 31 40 00	??...
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 68 00 00 00	e: DonaldDuck...

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



## 2. Analyze

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

Address	Hex dump	Disassembly	Comment
00401328	. 6A 04	PUSH 4	
0040132A	. 68 00300000	PUSH 3000	Protect = PAGE_READWRITE
0040132F	. 68 00010000	PUSH 100	AllocationType = MEM_COMMIT MEM_RESERVE
00401334	. 6A 00	PUSH 0	Size = 100 (256.)
00401336	. FF15 00304000	CALL DWORD PTR DS:[<&KERNEL32.VirtualAlloc	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 00020000	ADD EAX,2A0	
00401357	. A3 00404000	MOV DWORD PTR DS:[404040],EAX	
0040135C	. 0FB605 383140	MOVBX 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:[<&SUCP90.?count	
00401376	. 50	PUSH EAX	
00401377	. E8 04020000	CALL Advance_.00401580	
0040137C	. 8A46 01	MOV AL,BYTE PTR DS:[ESI+1]	
00401380	. 50	PUSH EAX	
00401381	. 50	PUSH EAX	
00401382	. 50	PUSH EAX	
00401383	. 50	PUSH EAX	
00401384	. 50	PUSH EAX	
00401385	. 50	PUSH EAX	
00401386	. 50	PUSH EAX	
00401387	. 50	PUSH EAX	
00401388	. 50	PUSH EAX	
00401389	. 50	PUSH EAX	
0040138A	. 50	PUSH EAX	
0040138B	. 50	PUSH EAX	
0040138C	. 50	PUSH EAX	
0040138D	. 50	PUSH EAX	
0040138E	. 50	PUSH EAX	
0040138F	. 50	PUSH EAX	
00401390	. 50	PUSH EAX	
00401391	. 50	PUSH EAX	
00401392	. 50	PUSH EAX	
00401393	. 50	PUSH EAX	
00401394	. 50	PUSH EAX	
00401395	. 50	PUSH EAX	
00401396	. 50	PUSH EAX	
00401397	. 50	PUSH EAX	
00401398	. 50	PUSH EAX	
00401399	. 50	PUSH EAX	
0040139A	. 50	PUSH EAX	
0040139B	. 50	PUSH EAX	
0040139C	. 50	PUSH EAX	
0040139D	. 50	PUSH EAX	
0040139E	. 50	PUSH EAX	
0040139F	. 50	PUSH EAX	
004013A0	. 50	PUSH EAX	
004013A1	. 50	PUSH EAX	
004013A2	. 50	PUSH EAX	
004013A3	. 50	PUSH EAX	
004013A4	. 50	PUSH EAX	
004013A5	. 50	PUSH EAX	
004013A6	. 50	PUSH EAX	
004013A7	. 50	PUSH EAX	
004013A8	. 50	PUSH EAX	
004013A9	. 50	PUSH EAX	
004013AA	. 50	PUSH EAX	
004013AB	. 50	PUSH EAX	
004013AC	. 50	PUSH EAX	
004013AD	. 50	PUSH EAX	
004013AE	. 50	PUSH EAX	
004013AF	. 50	PUSH EAX	
004013B0	. 50	PUSH EAX	
004013B1	. 50	PUSH EAX	
004013B2	. 50	PUSH EAX	
004013B3	. 50	PUSH EAX	
004013B4	. 50	PUSH EAX	
004013B5	. 50	PUSH EAX	
004013B6	. 50	PUSH EAX	
004013B7	. 50	PUSH EAX	
004013B8	. 50	PUSH EAX	
004013B9	. 50	PUSH EAX	
004013BA	. 50	PUSH EAX	
004013BB	. 50	PUSH EAX	
004013BC	. 50	PUSH EAX	
004013BD	. 50	PUSH EAX	
004013BE	. 50	PUSH EAX	
004013BF	. 50	PUSH EAX	
004013C0	. 50	PUSH EAX	
004013C1	. 50	PUSH EAX	
004013C2	. 50	PUSH EAX	
004013C3	. 50	PUSH EAX	
004013C4	. 50	PUSH EAX	
004013C5	. 50	PUSH EAX	
004013C6	. 50	PUSH EAX	
004013C7	. 50	PUSH EAX	
004013C8	. 50	PUSH EAX	
004013C9	. 50	PUSH EAX	
004013CA	. 50	PUSH EAX	
004013CB	. 50	PUSH EAX	
004013CC	. 50	PUSH EAX	
004013CD	. 50	PUSH EAX	
004013CE	. 50	PUSH EAX	
004013CF	. 50	PUSH EAX	
004013D0	. 50	PUSH EAX	
004013D1	. 50	PUSH EAX	
004013D2	. 50	PUSH EAX	
004013D3	. 50	PUSH EAX	
004013D4	. 50	PUSH EAX	
004013D5	. 50	PUSH EAX	
004013D6	. 50	PUSH EAX	
004013D7	. 50	PUSH EAX	
004013D8	. 50	PUSH EAX	
004013D9	. 50	PUSH EAX	
004013DA	. 50	PUSH EAX	
004013DB	. 50	PUSH EAX	
004013DC	. 50	PUSH EAX	
004013DD	. 50	PUSH EAX	
004013DE	. 50	PUSH EAX	
004013DF	. 50	PUSH EAX	
004013E0	. 50	PUSH EAX	
004013E1	. 50	PUSH EAX	
004013E2	. 50	PUSH EAX	
004013E3	. 50	PUSH EAX	
004013E4	. 50	PUSH EAX	
004013E5	. 50	PUSH EAX	
004013E6	. 50	PUSH EAX	
004013E7	. 50	PUSH EAX	
004013E8	. 50	PUSH EAX	
004013E9	. 50	PUSH EAX	
004013EA	. 50	PUSH EAX	
004013EB	. 50	PUSH EAX	
004013EC	. 50	PUSH EAX	
004013ED	. 50	PUSH EAX	
004013EE	. 50	PUSH EAX	
004013EF	. 50	PUSH EAX	
004013F0	. 50	PUSH EAX	
004013F1	. 50	PUSH EAX	
004013F2	. 50	PUSH EAX	
004013F3	. 50	PUSH EAX	
004013F4	. 50	PUSH EAX	
004013F5	. 50	PUSH EAX	
004013F6	. 50	PUSH EAX	
004013F7	. 50	PUSH EAX	
004013F8	. 50	PUSH EAX	
004013F9	. 50	PUSH EAX	
004013FA	. 50	PUSH EAX	
004013FB	. 50	PUSH EAX	
004013FC	. 50	PUSH EAX	
004013FD	. 50	PUSH EAX	
004013FE	. 50	PUSH EAX	
004013FF	. 50	PUSH EAX	

○ 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 비슷하다고 생각된다.

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

00401035	> 8B 0D 04444000	MOV ECX,DWORD PTR DS:[4044000]	
0040103B	. 8B15 50304000	MOV EDI,DWORD PTR DS:[<8MSUCP90.?end10es	MSUCP90.?end10esd@YAAAU?.\$basic_ostream
00401041	. 85C8	TEST EAX,EAX	
00401043	. A1 2C444000	MOV EAX,DWORD PTR DS:[40442C]	
00401048	. 0F94C3	SETB BL	
0040104B	. 3B01	CMP EAX,DWORD PTR DS:[ECX]	
DS:[00404404]=003C02A0, (ASCII "??")			
ECX=004031F8 (Advance_.004031F8)			
Jump from 0040102E			

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

Address	Hex dump	Disassembly	Comment
0040105F	. 89 18324000	MOV ECX,Advance_.00403218	ASCII "DonaldDuck"
00401064	. B8 0C444000	MOV EAX,Advance_.0040440C	ASCII "DonaldDuck"
00401069	. 8DA424 000000	LEA ESP,DWORD PTR SS:[ESP]	
00401070	> 8A10	MOV DL,BYTE PTR DS:[EAX]	
00401072	. 3A11	CMP DL,BYTE PTR DS:[ECX]	
00401074	. 75 1A	JNZ SHORT Advance_.00401090	
00401076	. 84D2	TEST DL,DL	
00401078	. 74 12	JE SHORT Advance_.0040108C	
0040107A	. 8A50 01	MOV DL,BYTE PTR DS:[EAX+1]	
0040107D	. 3A51 01	CMP DL,BYTE PTR DS:[ECX+1]	
00401080	. 75 0E	JNZ SHORT Advance_.00401090	
00401082	. 83C0 02	ADD EAX,2	
00401085	. 83C1 02	ADD ECX,2	
00401088	. 84D2	TEST DL,DL	
0040108A	. 75 E4	JNZ SHORT Advance_.00401070	

- 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 읽어들이어서 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 ( 여러 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.

00401200	> 84D2	TEST BL,BL	
0040120D	. 75 30	JNZ SHORT Advance_.004012DF	
004012AF	. 807C24 0B 00	CMP BYTE PTR SS:[ESP+B],0	
004012B4	. 74 29	JE SHORT Advance_.004012DF	
004012B6	. 0FB605 C83140	MOVZX EAX,BYTE PTR DS:[4031C0]	
004012BD	. 84C0	TEST AL,AL	
004012BF	. 74 47	JE SHORT Advance_.00401308	
004012C1	. BE C8314000	MOV ESI,Advance_.00403120	ASCII "Welcome, TheRightName. You've gai

C:\Documents and Settings\WL3m0n-Tr33\바탕 화면\Advance RCE L09.exe  
Welcome to this software. Please enter your registration details below to gain access.  
  
Username: DonaldDuck  
Password: 8921743  
  
I can't believe you felt for that! xD  
  
Welcome, TheRightName. You've gained access!

### 3.Conclusion

해당 문제는 HEX Dump 창을 볼 수 있는지, Heap 영역에 대한 기초적인 개념 및 Heap Spray 기법에 대한 약간의 소개, Code 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 수정하였다.

답 : 8921743