HW1-2: simpleshell

0016302 heron

Problem

Get flag string from the binary "simpleshell"

Step

Goal: Get the permission of admin by cracking the password, and run flag

Tool: radare2 + gdb

Observation in radare2

```
>> r2 ./simpleshell
[0x080486e0]> fs
00   strings
01   symbols
02   relocs
03   imports
04 * sections
[0x080486e0]> fs symbols
[0x080486e0]> fs
00   strings
01 * symbols
02   relocs
03   imports
04   sections
[0x080486e0]> f
0x0804a520 4 sym.stdin
0x08048e0c 4 sym._IO_stdin_used
0x0804a540 4 sym.stdout
0x080486e0 64 entry0
0x08048c76 64 main
[0x080486e0]> s main
[0x08048c76]> af
[0x08048c76]> pdf
```

then, you get asm code of the main function. Here's an important part:

```
|      ,==< 0x08048d1c     773a             ja loc.08048d58
|      ||   0x08048d1e     8b442468         mov eax, [esp+0x68]
|      ||   0x08048d22     c1e002           shl eax, 0x2
|      ||   0x08048d25     05b0910408       add eax, 0x80491b0
|      ||   0x08048d2a     8b00             mov eax, [eax]
|      ||   0x08048d2c     ffe0             jmp eax
|      ||   0x08048d2e     e8cafbffff       call dword fcn.080488fd
```

where **fcn.080488fd** is our login function. And,

```
[0x08048c76]> s fcn.080488fd
[0x080488fd]> pdf
```

you get asm code of the login function. We can know that the function is comparing strings to see if the password is correct or not:

```
|       |   0x08048a50     c7442404178f0408 mov dword [esp+0x4],
str.DoYouThinkThisIsPassword
|       |   0x08048a58     8d45ac           lea eax, [ebp-0x54]
|       |   0x08048a5b     890424           mov [esp], eax
|       |   0x08048a5e     e85dfbffff       call dword imp.strcmp
```

GDB for password guessing

Break at the line right before strcmp is called (0x08048a5e):

```
>> gdb ./simpleshell
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/secure-programming/wargame/05/simpleshell...(no
debugging symbols found)...done.
(gdb) b * 0x08048a5e
Breakpoint 1 at 0x8048a5e
(gdb) run
Starting program: /root/secure-programming/wargame/05/simpleshell
  _________            __________                          ___ ___          ____
 /   _____/ ____   ____\______   \_______ ____    ____    /   |   \__  _  _/_   |
 \_____  \_/ __ \_/ ___\|     ___/\_  __ \/  _ \ /___\  /    ~    \ \/ \/ /|   |
 /        \  ___/\  \___|    |    |  | \(  <_> ) /_/  > \     Y    /\     / |   |
/_______  /\___  >\___  >____|    |__|   \____/\___  /   \___|_  /  \/\_/  |___|
        \/     \/     \/                        /_____/          \/
Welcome! Type help for usage.
(anonymous) # login
Enter your name: admin
Enter your password: abcde

Breakpoint 1, 0x08048a5e in ?? ()
(gdb) x/30cb 0x08048f17
0x8048f17:      68 'D'  111 'o' 89 'Y'  111 'o' 117 'u' 84 'T'  104 'h' 105 'i'
0x8048f1f:      110 'n' 107 'k' 84 'T'  104 'h' 105 'i' 115 's' 73 'I'  115 's'
0x8048f27:      80 'P'  97 'a'  115 's' 115 's' 119 'w' 111 'o' 114 'r' 100 'd'
```

```
0x8048f2f:      0 '\000'        80 'P'  97 'a'  115 's' 115 's' 119 'w'
(gdb) info registers
eax            0xffffd2a4       -11612
ecx            0x33     51
edx            0x33     51
ebx            0x6      6
esp            0xffffd290       0xffffd290
ebp            0xffffd2f8       0xffffd2f8
esi            0x0      0
edi            0x0      0
eip            0x8048a5e        0x8048a5e
eflags         0x293    [ CF AF SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
(gdb) x/30cb 0xffffd2a4
0xffffd2a4:     -105 '\227'     98 'b'  80 'P'  100 'd' -109 '\223'     0 '\000'
51 '3'  8 '\b'
0xffffd2ac:     0 '\000'        0 '\000'        0 '\000'        0 '\000'        68
'D'  -57 '\307'      -26 '\346'      -9 '\367'
0xffffd2b4:     -12 '\364'      -1 '\377'       -5 '\373'       -9 '\367'       0
'\000'        0 '\000'        0 '\000'        0 '\000'
0xffffd2bc:     0 '\000'        0 '\000'        0 '\000'        0 '\000'        -8
'\370'       -46 '\322'
```

- the password answer is saved in **0x08048f17**
- the input password is saved in [eax]

Crack the password

By comparing our input string with the final password, we can guess the decode method applied in the simpleshell program.

4 characters in a group:

- **4*n+0**: input char XOR random char
- **4*n+1**: fixed (no need to change)
- **4*n+2**: static mapping(input char)
- **4*n+3**: fixed (no need to change)

There's a random number in **0x804a548**, so I did some brute froce thing to get though.

Solution

Make the follow output as input of simpleshell shall get the flag!

```
for guess in range(255):

    m1_o = 'DuniPw'
    m2 = 'hYexBC'

    s = 'ooTikhssasod'

    m1 = ''
    for i in range(len(m1_o)):
        m1 = m1 + chr(ord(m1_o[i])^guess)

    pwd = ''
    for i in range(6):
        pwd = pwd + m1[i] + s[i*2] + m2[i] + s[i*2+1]

    print 'login\nadmin\n' + pwd + '\nflag'
```

Issue

To solve this problem correctly, I've tried few decompilers as below:

- Hopper Disassembler: http://www.hopperapp.com/ (Limited for Free)
- Retargetable Decompiler: http://decompiler.fit.vutbr.cz/ (Free)
- Hex-Rays: https://www.hex-rays.com/index.shtml

Hex-Rays gave the best result so far.

However, for disassembler, radare2 is really nice since I like command lines.

More

- Faster Solution: Hex-Rays (IDA Pro Plugin)
- Radare2 Tutorial: http://maijin.github.io/radare2book/