Joseph Guzman

1/3/2020

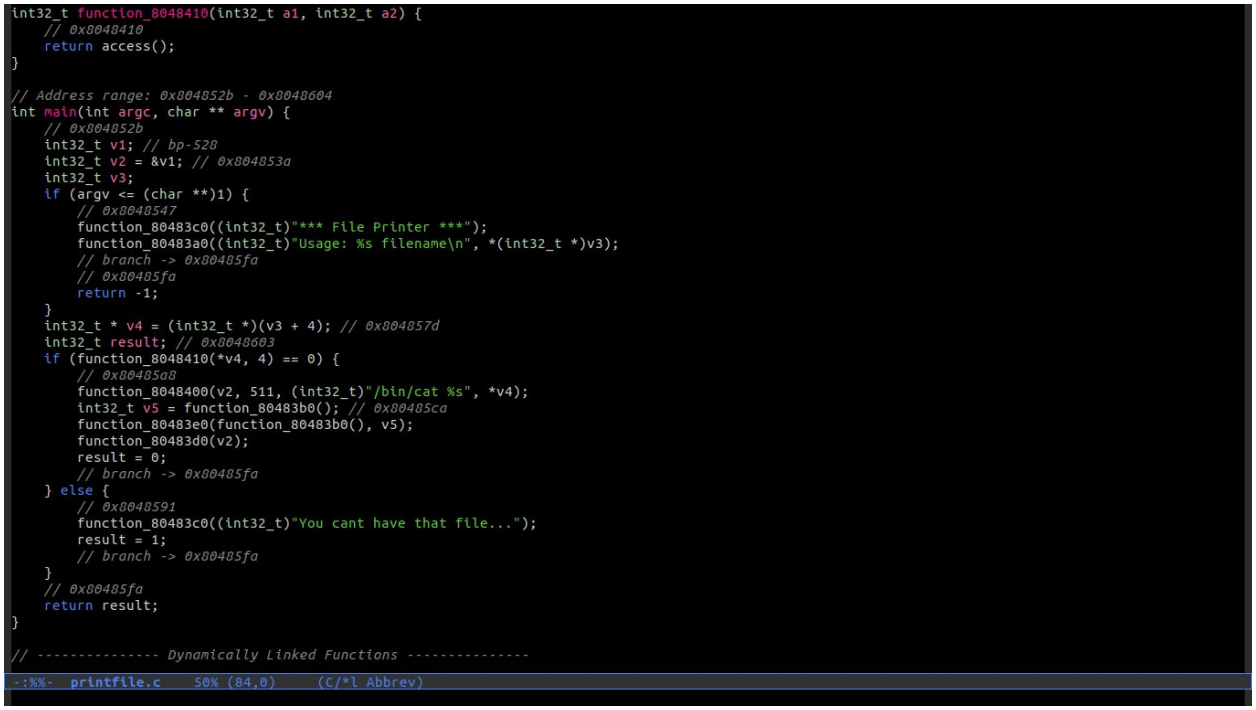
Leviathan02

This is the login info for this lab:

```
ssh server: leviathan.labs.overthewire.org
port: 2223
username: leviathan2
password: ougahZi8Ta
```

In this lab you are again only given an executable. The executable in this case is called `printfile`. It appears to just be a clone of the cat command. It will print to stdout whatever file you provide as a command-line argument. After running it a couple times I noticed that it only printed the first file you provide as a command-line argument to stdout.

I was able to convert this ELF 32-bit executable into both a readable assembly file and a decompiled C source code file via the retdec decompiler. With the decompiled version I could clearly see that the `/bin/cat` binary was being executed with the first command-line argument. See below for a screenshot of this code. You can see the full file in “printfile.c”.



```
int32_t function_8048410(int32_t a1, int32_t a2) {
    // 0x8048410
    return access();
}

// Address range: 0x804852b - 0x8048604
int main(int argc, char ** argv) {
    // 0x804852b
    int32_t v1; // bp-528
    int32_t v2 = &v1; // 0x804853a
    int32_t v3;
    if (argv <= (char **)1) {
        // 0x8048547
        function_80483c0((int32_t)"*** File Printer ***");
        function_80483a0((int32_t)"Usage: %s filename\n", *(int32_t *)v3);
        // branch -> 0x80485fa
        // 0x80485fa
        return -1;
    }
    int32_t * v4 = (int32_t *)(v3 + 4); // 0x804857d
    int32_t result; // 0x8048603
    if (function_8048410(*v4, 4) == 0) {
        // 0x80485a8
        function_8048400(v2, 511, (int32_t)"/bin/cat %s", *v4);
        int32_t v5 = function_80483b0(); // 0x80485ca
        function_80483e0(function_80483b0(), v5);
        function_80483d0(v2);
        result = 0;
        // branch -> 0x80485fa
    } else {
        // 0x8048591
        function_80483c0((int32_t)"You cant have that file...");
        result = 1;
        // branch -> 0x80485fa
    }
    // 0x80485fa
    return result;
}

// --------------- Dynamically Linked Functions ---------------
-:%%-  printfile.c    50% (84,0)     (C/*l Abbrev)
```

Figure 1: Screenshot of decompiled executable

My first thought was to pass as a command-line argument: `"; /bin/cat /etc/leviathan_pass/leviathan3"`. Which I thought may be executed with two commands. There were two problems. One is that the system call which executes the command likely is not capable of running multiple commands. Second is that the executable checks if the user running the program has access to the file. In the if statement of the above screenshot you can see the check `if (function_8048410(*v4, 4) == 0) {`. I have a screenshot of the definition of that function. It is just a system call to the access function. If you look at the documentation of access() linked here: http://man7.org/linux/man-pages/man2/access.2.html. You'll see that the purpose of this function call is to check if the user running this program permission to access the file.

I was stumped at this point so I looked over the writeup provided at: https://jhalon.github.io/over-the-wire-leviathan/. He explains it more briefly what he did is create a symbolic link to a file and use exploit a small problem in the executable. The executable checks if the real user (leviathan2) has permission to a file with access() but then uses `/bin/cat` to print a file as the effective user (leviathan3). Spaces are allowed in the access() call but spaces will separate arguments for `/bin/cat %s`.

So we need to provide a command-line argument that meets the following conditions:

1. leviathan2 has permission to read it, to pass the access() check
2. leviathan3 has permission to read it via `/bin/cat %s`, can be multiple space-separated files
3. Contains the password for leviathan3 so we can make it to the next lab

The solution to the 3rd problem is covered by creating a symbolic link. The solution to the 1st and 2nd problems are to create 2 files one in this case called "pass file.txt" which leviathan2 creates and has the right to read. Then a file called pass which is symbolically linked to "/etc/leviathan_pass/leviathan3". The argument we provide to the executable is `"pass file.txt"`. The executable will check if leviathan2 has access to the file `pass file.txt` which it does. Then the executable will execute `/bin/cat pass file.txt` which will cause it to interpret pass and file.txt as two separate arguments. pass is the symbolically linked file so we get the next password.

```
leviathan2@leviathan:/tmp/my_tmp$ ln -s /etc/leviathan_pass/leviathan3
pass
leviathan2@leviathan:/tmp/my_tmp$ ls -la
total 228
drwxr-sr-x    2 leviathan2 root   4096 Jan  4 02:55 .
drwxrws-wt 3578 root       root 225280 Jan  4 02:55 ..
```

```
lrwxrwxrwx    1 leviathan2 root     30 Jan  4 02:55 pass ->
/etc/leviathan_pass/leviathan3
leviathan2@leviathan:/tmp/my_tmp$ touch "pass file.txt"
leviathan2@leviathan:/tmp/my_tmp$ ~/printfile "pass file.txt"
Ahdiemoo1j
/bin/cat: file.txt: No such file or directory
leviathan2@leviathan:/tmp/my_tmp$
```

This is the login info for the next lab:

```
ssh server: leviathan.labs.overthewire.org
port: 2223
username: leviathan3
password: Ahdiemoo1j
```