Random Password Generator 2

Author:
Matteo Bordignon, 03/05/2024

Background

Reverse engineering 1-2 (also known as **backwards engineering** or **back engineering**) is a process or method through which one attempts to understand through deductive reasoning how a previously made device, process, system, or piece of software accomplishes a task with very little (if any) insight into exactly how it does so. Depending on the system under consideration and the technologies employed, the knowledge gained during reverse engineering can help with repurposing obsolete objects, doing security analysis, or learning how something works.
Software Reverse Engineering can be employed for various purposes such as: understanding legacy software, finding bugs and vulnerabilities, replicating functionalities, modifying or adding features, ensuring interoperability with other systems, bypass verification, etc.
A Reverse Engineering Capture The Flag (CTF) 3-4 challenge typically involves analyzing and understanding pre-built software or firmware to uncover hidden information, vulnerabilities, or secrets. Contestants are usually given a binary executable or a piece of software and are required to analyze it.

Vulnerability

The program calls a function `rand_pass` which generates a random password using the Process Identifier (**PID**) as seed, in this way the seed will be different at every execution of the program.
The password is created generating a random (printable) character at a time:

rand_pass()

```
1   char * rand_pass(ulong length){
2           int current;
3           char *buffer;
4           ulong i;
5           buffer = (char *)malloc(length + 1);
6           for(i = 0; i < length; i++){
7                   current = rand();
8                   buffer[i] = (char)current + (char)(current % 0x5e) +
    '!';
9           }
10          buffer[length] = '\0';
11          return buffer;
```

```
}
```

The problem is that the program allows two attempts to guess the password and since the seed is fixed (at each execution) we can use the first attempt to find the right seed and the second to actually guess the password. In other words, once we have found the seed we are able to predict the next password.

Solution

To solve this challenge we'll make use of Ghidra which is a decompiler, CDLL and pwn which are python libraries.
First in first we can decompile the given binary to try to understand how the program works. This is what we'll find in the main function:

main

```
seed = getpid();
srand(seed);
for (i = 0; i < 2; i = i + 1) {
    password = (char *)rand_pass(0x10);
    printf("Hello, give me the password: ");
    scanf(&DAT_00102022,user_input);
        isEqual = strcmp(password, user_input);
    if (isEqual == 0) {
            puts("Good job! Here is your flag:");
            system("cat flag.txt");
    }
    else {
            printf("Nope, the password was: %s\n",password);
    }
    free(password);
}
```

We can notice that the `getpid()` function is used to set the seed and the program allows two attempt for guessing the password. With these information we can build a python script 5 that solve the challenge as follows:

solution.py

```
for i in range(1, 32769):

        libc.srand(i)
        password = ''.join([chr(libc.rand() % 0x5e + ord('!')) for _ in range(16)])

        if first_attempt == password:
                print("found:" + password)
                break
```

```
9
10  password = ''.join([chr(libc.rand() % 0x5e + ord('!')) for _ in
    range(16)])
```

- the script uses a `for` to loop through all the possible PIDs (the maximum is 32768);
- for each possible PID, uses it as seed
- generates the password using the same procedure of the program;
- checks is the obtained password is equal to the password provided by the program at the first attempt. If true we found the right seed (PID)
- generates the password using the right seed and sends it back to the server.

References

1. Reverse Engineering wikipedia
2. Reverse Engineering Malware Unicorn's
3. Reverse Engineering CTF
4. What is a reverse engineering ctf?
5. Full python script