Computer Security Midterm Assignment

Parand Mohri

October 2022

The goal is to access secret message inside a C file provided called assignment.c and decode the message. At line 24 in this file there is an array made with length 256 which suppose to save the input provided by the user, but there is no checking if the size of input provided is ¡=256. That make the file vulnerable to buffer overflow attack. inside the file there exist a method called `print_secret_message()` that print the secret message. Buffer overflow attack can be used to call this method and get access to the secret message.

1 Bufferoverflow attack

First Step: Getting the memory address of the method `print_secret_message()`. For this a virtual machine being used to access ubuntu linux, to disable the ASLR(Address space layout randomization) which is used to protect the program from buffer overflow attacks. This only can be done in ubuntu using:

```
echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
```

Then downloading gcc for running C codes:

```
sudo apt install gcc
```

Shared file option of virtual machine was used to access the assignment.c file. The file was compiled using:

```
gcc -fno-stack-protector -z execstack -g assignment.c -o assignment
```

This command compile the file and disable the stack protector to allow the buffer overflow attack and be able to open the code in gdb. First name is the name of the file(assignment.c) and the second is the name of the executable file that it make after compiling(assignment) . Open the file in gdb which can be used as debugger with following command:

```
gdb assignment -tui
```

(-tui give access to a ui that can be used to edit the code as well, but process can be done without ui and only using `gdb assignment`)
To access the memory point of the method the code need to break in the middle of running process for this a break point was added at the start of main method using:

```
Break main
```

Start the running using:

`Start`

Access the address of the method using

`Info address print_secret_message`

Address given: 0x5555555552db

Second Step: Use the address to attack.

Make another file called exploit.c which write 264 "x"'s, 256 for filling the input array and 8 more for overwriting the pointer, and add the above address to the end of the string. Using the command below compile the exploit.c file with name exploit:

`gcc exploit.c -o exploit`

Using the following command ran the output of exploit file as input on assignment file:

`./exploit | ./assignment`

(./ in ubuntu means run)

Which revealed the secret message:

OLZLJYLATLZZHNLPZHNYLLHISLULZZLZ

This is happening because the address of `print_secret_message()` was the first thing to run after the input array and the pointer was overwrited so the method was called.

2 Decoding the message

Now that secret message is known it should be decoded. It's known that the secret message is in English language and the encryption is happening in Caesar cipher manner. Using this information it's known that the most frequent letter in the message should be replaced by "E" (because that's the most frequent letter in English language). After that its just reordering the alphabet.

Following website was used to find the most frequent letter https://www.dcode.fr/frequency-analysis which is L with 31.25%. With this its shown that:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

is replaced by:

T, U, V, W, X, Y, Z, A, B, C,D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S.

Doing this the message is:

hesecretsmessageisagreeablenesses

Where it's easily seen that it means "the secret message is agreeable " but the first T is deleted and some letter are added to the end as a adding layer to the encryption.