COM-402 exercises 2023, Session 6: Web and software vulnerabilities

Exercise 6.1

You are writing a web application to sell Whisky. You want to save your customers' names in a database. You know that single quotes (') can break SQL request.

- What can you do to be able to accept single quotes in names and still have no problem with SQL injections?

Exercise 6.2

Consider the following code, taken from the Mitre Common Weakness Enumeration website (https://cwe.mitre.org). A web application has a function to run a backup of a database. The backup can be of type `full` or `incremental`. The type of backup is selected by the user and is sent to the server in the parameter named `backuptype`. The following code is used on the server:

```
...
String btype = request.getParameter("backuptype");
String cmd = new String("cmd.exe /K \"
c:\\util\\rmanDB.bat\"
+ btype +
\"&& c:\\util\\cleanup.bat\"")
System.Runtime.getRuntime().exec(cmd);
```

- Name the type of attack that could happen here and explain its possible consequences.

Exercise 6.3

Memory pages can be protected against writing or execution.

- Explain why it is dangerous to have pages where both execution and writing are permitted.

Exercise 6.4

At the end of a function call, two addresses are often popped from the stack.

- What are those two addresses used for ?

Exercise 6.5

Local variables are on the top of the stack and the return address at the bottom.

- How can a buffer overflow overwrite the return address that is below the variable on the stack ?

Exercise 6.6

- Why must a stack canary have a random value ?

Exercise 6.7

Imagine a program where the name and the price of a product are stored in memory just above a variable containing the shipping address of the product.

Be entering an extra long shipping address, the customers are able to modify the price of the product and buy it for cheaper.

For each of the following protection methods, explain if it could prevent this attack:

- address space layout randomization (ASLR)
- marking the memory page as non executable or non writeable
- using a stack canary

Solutions to the Exercises

Solution 6.1

You can always encode the data, so that special characters lose their special meaning. In SQL a literal single quote (`'`) can be encoded by prepending a backslash (`\'`).

Solution 6.2

This code is vulnerable to *command injections*. By inserting the correct characters into the `backuptype` and attacker can execute any command on the server, for example deleting all the files.

(In our case the program runs the Windows command interpreter `cmd`. CMD accepts multiple commands if they are separated by `&` or `&&`. Thus, the following value for `backuptype` would delete all files in the current directory: `& del *.*` .

Solution 6.3

If a page is both writable and executable, an attacker could store data that corresponds to instructions (a *shellcode*) and then manipulate the instruction pointer to have this code executed.

Writable memory pages are used to store data received by the program. Executable pages contain instructions of the program.

As there is often no need to modify the instructions of a program that is executed, executable pages are set to read-only and writeable pages to non-executable. This prevents the execution of shellcode, without interfering with the correct execution of the program.

Solution 6.4

One address is the *base pointer* of the function to which we are returning. The base pointer points the start of the memory region where the local variables of a function are stored.

(As the current function is about to end, its local variables will not be needed anymore. However, the function we are returning to will need to access its local variables. This is why we restore the value of the base pointer to the value we stored on the stack before we started the function.)

The second address that is popped from the stack is the *return address* which indicates at which address in the code the execution has to continue after the end of the function.

Solution 6.5

The stack actually grows from the top of the memory to the bottom of the memory. Thus, the address of the bottom of the stack is higher than the addresses at the top of the stack. When we overflow a buffer, we write to addresses that get higher and higher and thus reach the bottom of the stack.

Solution 6.6

The stack canary is used to detect if the saved base pointer and the return address have been overwritten by an overflow. If an attacker could know the value of the canary, they could overwrite it with the same value and the overflow would not be detected.

Solution 6.7

ASLR will place the memory page of the variables at a random location. They will still be adjacent and it will not prevent the attack.

The program will not work if the memory page is not writeable, The attack does not write any code into the memory. Marking the page non-executable will not prevent the attack.

Finally, if the shipping address does not overflow more than the memory used for the price, it will not affect any canaries. Thus, non of these methods can prevent the attack.