Computer Security and Networks: Exercise 5

Deadline: 4 May 2023, 4pm

Application Security

This final exercise looks at reverse engineering applications and buffer overflow attacks. These powerful exploits will allow you to take complete control of the VM.

For this exercise you should log into the new VM as the Bob user: `bobpark`, password: `redfiveworldequal`. In Bob's home directory you will find a jar file, and two ELF executables and the tool JD-GUI. For this exercise you need to use JD-GUI and Ghidra or gdb to analyse the applications and gain a complete understanding of what they do.

Full marks will be awarded if two tokens are submitted, and 50% of the mark will be awarded if one token is submitted.

The current virtual machine for the Apple Arm architecture is not suitable for some parts of this exercise. The details are as follows. Because the C-binaries are x86-binaries, you will need to download a new virtual machine which uses the x86-architecture from canvas - see the link on the exercise sheet from canvas. This virtual machine is emulated, hence any application using graphics is too slow to be useful. You will need to use the command line and gdb. Hence I propose to use the Arm Virtual machine for the first part of the exercise, so that you can use jd-gui. However, on this machine there is no server running on port 5003 - you will need to use the new virtual machine to get the token once you know the password.

Question 1: Java Byte Code, javaPasswordCheck.jar

The jar file in Bob's home directory employs one of the most common methods of protecting code: encryption. As you will see, this method will not stop a determined analyst. The jar file encrypts some of its code, however, the decryption key must be embedded in the application, and so an analyst can read the code. This is an example of "packing" which is a protection method often use by malware. This is done mainly to avoid signature based detection from malware scanners; the malware will re-encrypt itself with a different key each time it infects a computer, so making it look different each time it spreads.

The jar file is a simple password check (you can run the password check jar file by typing `java -jar javaPasswordCheck.jar` at the command line). Use JD-GUI to find the password for this program. Another version of this application (with the same password) is running on port 5003 of the VM. The version of the application listening on the port will give you a token in response to the correct password.

[5 marks]

Question 2: ELF Binary, cpasswordcheck

Executable and Linkable Format (ELF) is the standard format for linux executables. In Bob's home directory, you can find a simple password check program. You can run this from the command line by typing `./cpasswordcheck`.

You will need to enter a password in order to be given a message. Examine the assembly code using ghidra or gdb. Work out how the password is being checked and what the message is. Another version of this application (with the same password) is running on port 5001 of the

VM. The version of the application listening on the port will give you a token in response to the correct password.

Warning: If port 5001 is not open/reachable, try rebooting the VM

[5 marks]

Question 3: Buffer Overflows, namecheck

The file `nameCheck` (in Bob's home directory) was compiled using the command:

```
gcc -fno-stack-protector -z execstack -o nameCheck nameCheck.c -m32
```

and the following code:

```
#include <stdio.h>

void function1(void) {
    printf("Enter your name:\n");
    char buffer[64];
    gets(buffer);
    printf("No token for you %s!\n",buffer);
    printf("By the way buffer was at: %p",buffer);
}
void function2(void) {
    // Open file
    FILE *fptr;
    fptr = fopen("overflowToken1", "r");
    // Read contents from file
    char c = fgetc(fptr);
    while (c != EOF)
    {
        printf ("%c", c);
        c = fgetc(fptr);
    }
    fflush(stdout);
    fclose(fptr);
}
int main(void) {
    function1();
    return 0;
}
```

i.e. all the memory protections have been turned off. Additionally, ASLR has been disabled on the VM.

Find a buffer overflow attack against the above code that will let you read the contents of the file `overflowToken1`. N.B. the program executes with `root` permissions and the tokens are only readable by root. For this question do not try to execute code on the stack.

You may find it helpful to sketch the layout of the stack before and after your attack. Once you have the token submit it to the token submission site.

[5 marks]

Question 4

Find a second buffer overflow attack against this program that will let you open a shell and change the root password. You may find it helpful to sketch the layout of the stack before and after your

attack. Once you have root access to the VM go to the directory `/root/` and submit the token in this directory to the token submission website. **[5 marks]**

Hints

- Before starting you should review the lecture videos and make sure you understand everything I did in lectures and how the x86-64 function call works.
- When you execute a program in gdb it always executes with your permissions, therefore you cannot access the tokens when running the program in gdb.
- A program can have different stack offsets when executed in gdb vs when it is executed normally.
- The commands `while read -r line; do echo -e $line; done | ./nameCheck` will let you run a program and enter bytes using hex notation, e.g. using these commands in the input \x00 will enter a byte of 0s into memory, instead of the ascii characters for "\" "x" "0" and "0" as you would normally get.
- gdb uses `stdin` for inputting gdb commands. If you want to use `stdin` for input into the program being debugged, use the gdb-command `run < <filename>`.
- Do not try to execute `/bin/sh` as part of the shell code - this will not work, as discussed in the lecture. Instead try to execute the program whose source is contained in the file `msh.c`, which you find on the exercise section on canvas.
- The command `cat` can be more reliable than `more` for viewing files, as `more` will try to display exactly one page and may fail if it can't work out what a page is.
- You can find the shell code I used in lectures on canvas.