Sicurezza e Privatezza
LAB 4 – Metasploit and buffer overflow

The learning objective of this lab is for students to get the basics of exploitation and post-exploitation with Metasploit and a little dive into memory errors with buffer overflow attack.

Students are invited to complete the following exercises by write the exploit in a txt file or attach the screenshot that prove the success of the exercise. The submission will be through upload.di.unimi.it by zipping all the files in an archive named **`Lab4NameSurnameUniversityid`** (eg. Lab4MatteoZoia123456.zip). Files not conforming to these rules will be automatically rejected.

The first student who will respond correctly to the entire lab will receive a bonus of 0.5pt on the final grade.

Exercise #1: First attack

In this task, we will use the `unix/misc/distcc_exec` exploit to demonstrate the overall process. Make sure that both VM1 (Kali) and VM2 (metasploitable) are booted. In VM1 launch Armitage and find the exploit distcc_exec: `exploit/unix/misc/distcc_exec.`

Determine the available options by double click with Armitage on the selected exploit (or with Metasploit: `msf exploit(distcc_exec) > show options`).

To make the exploit work, you still need to specify the values of related environment variables, e.g., *RHOST*, the target IP address of vulnerable machine (`ifconfig` on the metasploitable). You can leave *PAYLOAD* to the default value or you can use something else (`cmd/unix/reverse`).

Finally, run the exploit by simply running `exploit` in metasploit or launch in Armitage. **For this exercise submit a screenshot of the window with the working exploit (prove that you gain root privilege on the matasploitable machine) and reports all command used.**

Exercise #2: Tomcat server attack

In this task you will learn about reconnaissance and scanning ports. Armitage contains a plugin that wraps nmap for us. As it scans a host, the results are stored within a database for later plugins to access. **Run a portscan** with Armitage or throught msf on the metasploitable target (`db_nmap -sV 10.10.15.x`). You can view scanned hosts with `db_hosts`.

Your portscan should reveal that the target is running Apache Tomcat. You can find an exploit for this service named *multi/http/tomcat_mgr_deploy*. Be sure to set your exploit options appropriately. Google for **default credentials for this service**. Use the nmap data to set RPORT. **Use the techniques you learned from exercise 1 to gain**

remote control. For this task you need to report all your commands, options and a screenshot (to prove that you gain root privilege on the matasploitable machine).

Exercise #3: First buffer overflow

The following program too accepts user-input through the gets function and then looks for a specific value in a local variable named secret. If this value is equal to a certain pre-defined constant, `printf` function is used to show a you win! message to the user. There is no direct means of modifying the content of the secret variable. The gets function will keep reading from the stdin device until it encounters a newline or `EOF` character. Since this reading loop fails to honor the size of the destination buffer, a classic buffer overflow vulnerability is introduced in the program. **The student has to find this vulnerability and exploit the program es3.c so that it prints the "you win!" message to `stdout`. Attach a screenshot of the `stdout` and the payload given as input to the program.**

Compile with canary guard disabled:

```
gcc -fno-stack-protector -z execstack es3.c -o es3
```

Disable ASLR countermeasure before running:

```
sysctl -w kernel.randomize_va_space=0
```

```
int main() {

   int secret;
   char buf[16];

   printf("buf: %08x secret: %08x\n", &buf, &secret);
   gets(buf);

   if (secret == 0x4349414F)
      printf("you win!\n\n");
}
```

Exercise #4: Unprintable buffer overflow

The following program has a buffer overflow vulnerability, most of the concepts are very similar to the previous exercise, but pay attention to unprintable characters. **The student has to find this vulnerability and exploit the program es4.c so that it prints the "you win!" message to `stdout`. Attach a screenshot of the `stdout` and the payload given as input to the program.**

```
int main() {

  int secret;
  char buf[64];

  printf("buf: %08x secret: %08x\n", &buf, &secret);
  gets(buf);

  if (secret == 0x01020305)
     printf("you win!\n");
}
```

Exercise #5: Change the flow

The following program has a buffer overflow vulnerability, most of the concepts are very similar to the previous exercises but at this time there's a `0x00` character that truncate the input stream of the `gets`. The goal is the same: you need to print out the string "you win!", think about if you can smash the stack until you reach the return address pushed by the main. **The student has to find this vulnerability and exploit the program es5.c so that it prints the "you win!" message to `stdout`. Attach a screenshot of the `stdout` and the payload given as input to the program.**

```
int main() {

  int secret;
  char buf[80];

  printf("buf: %08x secret: %08x\n", &buf, &secret);
  gets(buf);

  if (secret == 0x01020005)
    printf("you win!\n");
}
```