Systems Security (Spring 2023)

Homework 1

Hand-out: 2 Feb

Due: 12 Feb (Sunday) 23:59

Group number: _____

Student names:

_____, _____

_____, _____

_____, _____

1. The code below provides an authentication mechanism based on password checking.

1) What security principle(s) are violated? In what aspect(s)?
2) Can you recover the password without accessing `_passwd` directly? Please write a code that creates an instance of BlackBox(), and recovers the password by interacting with that instance.
3) Describe how to fix the method. Please write a new class SecureBlackBox() to demonstrate your method.

```
import os
import random

class BlackBox(object):
    def __init__(self):
        len_ = random.randint(100, 1000)
        self._passwd = os.urandom(len_)

    def check(self, user_passwd):
        if len(user_passwd) != len(self._passwd):
            return "Error:len"
        for i in range(len(user_passwd)):
            if user_passwd[i] != self._passwd[i]:
                return "Error:idx:%d" % i
        return "OK"
```

2. Write an application that if executed by any Linux user will print the user's own password's hash from `/etc/shadow`.
Hints:

1) See `man shadow` and Python's `os` package.
2) For compiling a `.py` file to binary you may use `nuitka`.
3) Checkout the use of setuid bit.

3. Find out at least 4 different Linux mechanisms that Docker uses to provide an isolated container environment for applications. Describe (in 5-10 sentences) each Linux mechanism, how Docker uses them, and what kind of security goals that mechanism helps achieve.

Some links that maybe useful (you are highly encouraged to explore mechanisms that are not included below and share with the class):

https://docs.docker.com/engine/security/

- Chroot:
https://help.ubuntu.com/community/BasicChroot

https://www.tecmint.com/restrict-ssh-user-to-directory-using-chrooted-jail/

- Linux namespaces:

https://man7.org/linux/man-pages/man7/namespaces.7.html

- Linux control groups:

https://man7.org/linux/man-pages/man7/cgroups.7.html

- Linux capabilities:
https://www.vultr.com/docs/working-with-linux-capabilities
https://blog.container-solutions.com/linux-capabilities-why-they-exist-and-how-they-work
http://manpages.ubuntu.com/manpages/zesty/man7/capabilities.7.html

4. Given the following information:

Members of group `staff`: `alex`, `benn`, `cloe`
Members of group `gurus`: `cloe`

```
$ ls -ld . * */*
drwxr-xr-x  1 alex staff    32768 Apr  2  2010 .
-rw----r--  1 alex gurus    31359 Jul 24  2011 manual.txt
-r--rw--w-  1 benn gurus     4359 Jul 24  2011 report.txt
-rwsr--r-x  1 benn gurus   141359 Jun  1  2013 microedit
dr--r-xr-x  1 benn staff    32768 Jul 23  2011 src
-rw-r--r--  1 benn staff    81359 Feb 28  2012 src/code.c
-r--rw----  1 cloe gurus      959 Jan 23  2012 src/code.h
```

The file `microedit` is a normal text editor, which allows its users to open, edit and save files.

1) Fill in each cell in the following table with R, W:

	manual.txt	report.txt	microedit	src/code.c	src/code.h
alex					
benn					
cloe					

For each cell, please provide brief explanation how you derive the privilege.

2) Create the specified users, groups, and files (the actual content in the files is not important) in an actual Linux system and verify your results. Please provide the list of commands / or a screenshot (screenshots) of the commands you used to setup the configuration.

5. Cause the following application to print "ACCESS GRANTED". Submit your code. Please describe what is the problem and how to prevent it.

```
#include <stdio.h>
int main(int argc, char *argv[]){
     int arg1, arg2;

     if (argc != 3)
          return -1;

     arg1 = atoi(argv[1]);
     arg2 = atoi(argv[2]);
     if (arg1 < 0 || arg2 < 0)
          return -1;

     if (arg1 + arg2 >= 0)
          return -1;
     printf("ACCESS GRANTED\n");
}
```

6. Draw out the exact memory layout (with the name, size, and correct location of all relevant control and user data) of the stack area when `fun3()` is invoked (i.e., right after its stack frame is added to the top of the stack).

```
void fun3(void){
     char a[8];
     double b;
     return;
}

void fun2(void){
     fun3();
}

void fun1(void){
     char a[16];
     int b;
     float c;
     fun2();
}

int main(void){
     fun1();
}
```

7. Setup a Linux (x86) **32-bit** and gcc environment (you can use VirtualBox and the image provided by https://infsec.ethz.ch/appliedlabbook.html, e.g.: http://www.files.ethz.ch/appliedinfsec/alice_v1.1.ova.zip). Install Guest Additions if using VirtualBox.

   Compile vuln1.c (see below) with the code below. Then find and exploit errors (crashing the application).

```
1 # sudo echo 0 > /proc/sys/kernel/randomize_va_space
2 $ gcc -g -fno-stack-protector -z execstack -o vuln1
vuln1.c
3 $ sudo chown root vuln1
4 $ sudo chgrp root vuln1
5 $ sudo chmod +s vuln1
6 $ gdb vuln1
```

Explanation:

Line 1: Disables Address Space Layout Randomization (ASLR)
Line 2: `-fno-stack-protector` -> Disables Stack Canary
`-z execstack` -> Disables NX bit

Use gdb vuln1 to debug the program. See some useful gdb commands below. Modify the code below to overwrite EIP with 'FFFF'. Use info registers to see value of EIP.

```
r `python -c 'print("F"*4)'`
```

Modify the code to override EBP with 'FFFF' and EIP with 'PPPP'. Submit the list of commands you used and a screenshot to show that you have conducted this attack successfully.

vuln1.c:

```
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv[]) {
     char buf[128];
     strcpy(buf,argv[1]);
     printf("Input:%s\n",buf);
     return 0;
}
```

Useful GDB commands

```
# show debugging symbols
list main
# show the assembly code
```

```
disas main
# run the program, with input
r Hello
r `python -c 'print("A"*3)'`
# confirm overwrite of ebp register
info registers
# examine memory address
x/200x ($esp - 550)
# show value of Instruction Pointer Register (EIP)
p/x $eip
```

https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf

=== END ===