14-741: Homework 2

Due: Monday, October 15, 2018 (by 2:30pm Eastern)

Name:

Andrew ID:

Scores

Part 1 (50 pts max):

Part 2 (50 pts max):

Total (100 pts max):

Guidelines

- Be neat and concise in your explanations.
- You must use at most one page for your explanation for each problem (2.1, 2.2, ... each on separate pages) (code you wrote may be on additional pages). Start each problem on a new page. You will need to map the sections of your PDF to problems in Gradescope.
- Please check your English. You won't be penalized for using incorrect grammar, but you will get penalized if we can't understand what you are writing.
- Proofs (including mathematical proofs) get full credit. Statements without proof or argumentation get no credit.
- There is an old saying from one of my math teachers in college: "In math, anything that is only partially right is totally wrong." While I am not as loathe to give partial credit, please check your derivations.
- **This is not a group assignment. Feel free to discuss the assignment in general terms with other people, but the answers must be your own.** Our academic integrity policy strictly follows the current INI Student Handbook `http://www.ini.cmu.edu/current_students/handbook/`, section IV-C.
- Write a report using your favorite editor **Only PDF submissions will be graded.**
- Submit to Gradescope a PDF file containing your explanations and your code files before 2:30pm Eastern on the due date. Late submissions incur penalties as described on the syllabus (first you use up grace credits, then you lose points).
- Post any clarifications or questions regarding this homework in Piazza.
- Good luck!

1 Exploiting a Buffer Overflow

You will use the CTF server from HW1 for this assignment!

Reverse Engineering Each of these program has a vulnerability similar to that described in AlephOne paper assigned in class. Please use the AlephOne paper as guideline, and go through the following steps:

1. Use `objdump` to dump the assembly of this program. You only need to focus on two functions (main and vuln) in the dump result.
2. Analyze the program using `gdb` and identify the assembly code that is/are vulnerable.
3. Draw a map of the stack with addresses and contents (such as return address, variables, ...) stored in the stack.
4. the stack address when the program is executed without `gdb` is 0x30 or 0x40 higher than the stack address when the program is executed with `gdb` because of environment variables used by `gdb`
5. use `Python` to compose the non-ASCII input of your program.

1.1 My first buffer overflow (10 points)

Overflow the stack and replace the return address with the address of the win method. Solve the Buffer Overflow problem on the 14741 CTF Server. Submit a writeup containing your CTF teamname and what steps you used to solve the problem. *Make sure to cd to the folder containing the problem, or the program will seg fault when you win and it attempts to read the flag.*

1.2 Buffer Overflow 2 (10 points)

Now you will need to get shellcode to run. Solve the Buffer Overflow 2 problem on the 14741 CTF Server. Submit a writeup containing your CTF teamname and what steps you used to solve the problem. *Pay careful attention to the hint about "cat" or else you might get a shell but not be able to do anything because stdin is closed*

1.3 Buffer Overflow 3 (10 points)

We've added something like a stack canary. Can you figure out what the stack canary is? Solve the Buffer Overflow 3 problem on the 14741 CTF Server. Submit a writeup containing your CTF teamname and what steps you used to solve the problem.

1.4 Buffer Overflow 4 (10 points)

The stack is no longer executable! You need to do a return to libc attack. Solve the Buffer Overflow 4 problem on the 14741 CTF Server. Submit a writeup containing your CTF teamname and what steps you used to solve the problem. *Here's some python that lets you interact with an executable. Put it in your home folder as buffer4.py, cd to the folder with the problem and do python ~/buffer4.py*

```
from subprocess import Popen, PIPE, STDOUT
p = Popen(['./vuln'], stdout=PIPE, stdin=PIPE, stderr=STDOUT)
print p.stdout.readline().rstrip()
print p.communicate("a"*190)[0].rstrip()
```

1.5 Buffer Overflow 5 (10 points)

The stack has been randomized! But on a 32 bit machine, there aren't a lot of bits of randomness. Solve the Buffer Overflow 5 problem on the 14741 CTF Server. Submit a writeup containing your CTF teamname and what steps you used to solve the problem.

2 Access Control

University C's instructors of class X decide to set up the assignment submission system using the file system of X. LJ and MC are instructors, Alice and Bob are TAs, and ST_1, .., ST_n are n students enrolled in this class. Files of interest include assignment handout, submitted assignments, and grade reports. The desired access control policies are as follows.

1. Only LJ, MC, Alice, and Bob are allowed to write to the assignment handouts.
2. Everyone can read the assignment handouts.
3. LJ, MC, Alice, and Bob can read students' assignment submissions.
4. A student can read his or her own assignment submissions, but not other students' submissions.
5. A student can write (modify) his or her own assignment submissions, but not other students' assignment submissions.
6. LJ, MC, Alice, and Bob are not allowed to write to students' assignment submissions.
7. LJ and MC can read and write the grade reports.
8. Alice and Bob can read the grade reports, but not write any grade reports.
9. A student is allowed to read his or her own grade report, but not other students' grade reports.
10. A student is not allowed to write any grade reports.

Please answer the following questions.

2.1 Unix File System (10 pts)

Use Unix file system permissions to implement the above policy. Describe your ACL.

2.2 Multi-level/multi-lateral Read (10 pts)

Implement all the read policies using BLP/Biba (use a lattice). Explain which model (BLP or Biba) did you choose and why. Present your lattice and describe how the policies can be enforced.

2.3 Multi-level/multi-lateral Write (10 pts)

Implement all the write policies using BLP/Biba (use a lattice) Explain which model (BLP or Biba) did you choose and why. Present your lattice and describe how the policies can be enforced.

2.4 Role-based Access Control (10 pts)

Implement the above policies using role-based access control (RBAC). Write down the roles, role assignments, and the access control matrix that you come up with.

2.5 Revocation (10 pts)

One of the TAs (Bob) is fired from the university in the middle of the semester. Describe the procedure to revoke his access to files in the Unix file system, in BLP/Biba model, and in RBAC model.