Homework #1

David Brumley, Peter Chapman, Dominic Chen, Jiyong Jang, Roy Ragsdale, Zachary Weinberg

Due: 12:01 am on September 23, 2015
Release: September 14, 2015

1 Introduction

As a special agent from the 487 hacking group, you have been assigned a secret mission. In this mission (a.k.a. homework), you need to break into the DJB company’s systems and steal some secret documents. Keep in mind that tasks 1 through 3 should be performed on the SECURESSHHELL machine, for which ASLR is turned off, and tasks 4 and 5 should be performed on the SUPERSECURESSHHELL machine, for which ASLR is turned on.

1.1 Grading

1.1.1 Rules

• You can discuss with others, but you must create your own exploits.
• Use Piazza to ask questions.
• DO NOT try to obtain root access on the systems.
• DO NOT brute-force the systems.
• DO NOT solve problem instances for which you were not assigned.
• DO NOT share any secret keys.

1.1.2 Criteria

Each task will be graded based upon the following 3 criteria. Partial credit will be given as deemed appropriate by the graders.

• Reversing (10 pts): Explain what the program does, especially how the user input is processed (feel free to present a pseudocode version of the program). Then, identify the vulnerability in the program, i.e., which kind of vulnerability it is and where the vulnerability is. Be sure to explicitly name the functions relevant to the vulnerability (strcpy, printf, etc.).
• Exploiting (10 pts): Create your own exploit and provide a full description of how your exploit works with a stack diagram. Be sure to explain how your exploit gets around any countermeasures in the target program.
• Pwning (5 pts): Obtain a secret key using your exploit.
1.1.3 Submission

First, submit secret keys for each solved problem to the SUBMISSION server. Next, submit the following items as a ZIP archive labeled \{your andrew id\}.zip to Blackboard:

- a typed writeup in PDF format describing the three above criteria for every task.
- working exploits for each problem. Your exploit will be tested from the command line on the appropriate shell server, not from within gdb.

1.2 Scoreboard

To register on the system, visit the SUBMISSION server, and register a new user account using your Andrew ID. After logging in, join the class 18487-f15 with a teacher username of deploy. The problem text under the Problems tab will specify the location of each problem binary for your user account. To interact with the problem, you can download the binary by clicking on the link in the problem text, connect to the server via the Shell tab of the SUBMISSION website, or directly to one of the shell servers using your account credentials.

When you complete a task, you will get a secret key (a.k.a. flag) in the form of an MD5 hash. A valid secret key is the evidence showing you solved the problem. This must be submitted promptly to the submission server using your account, and should not be disclosed to anyone else.

SUBMISSION: http://snowshoes.ece.cmu.edu
SECURESHELL: ssh://snowshoes.ece.cmu.edu:2201
SUPERSECURESHELL: ssh://snowshoes.ece.cmu.edu:2202

NOTE: Both shell servers are configured to restart daily at 4 am.

2 Task 1: recoverpw (25 pts)

Your first task is to infiltrate the DJB company’s network. By eavesdropping on conversations between employees of the DJB company, you have obtained access to a password recovery service on the SECURESHELL system. Your goal is to retrieve the credentials by exploiting the password recovery service (or maybe by guessing a correct PIN...). You should analyze and understand the binary recoverpw first, then find the vulnerability, and make an exploit for it. A successful attack will give you the login flag. (Hint: you might be interested in the recover passwd function.)

Tips:

1. You can print hexadecimals using perl or python. For example:
   perl -e 'print \xFF\xFF"x4''
   python -c 'print("\xFF\xFF"*4)''

2. Don’t forget about endianness of memory addresses.

3 Task 2: cfo.snote (25 pts)

Now you are in the DJB company’s SECURESHELL system. The Chief Financial Officer (CFO) of DJB uses a program called snote (secure note) to record confidential messages. Your goal is to read flag.txt by exploiting cfo.snote. (Hint: you might be interested in note function.)
Tips:

1. Due to the environment variables, the stack addresses when a program runs in gdb may be different from those when the program is run directly. To mitigate this problem, run the following commands in gdb before running the program:

   ```bash
   unset env COLUMNS
   unset env LINES
   unset env OLDPWD
   set env _ $HOME
   ```

   Then, use the absolute path when running at the command line.

2. You can find various shellcode for Linux at
   `http://www.shell-storm.org/shellcode/shellcode-linux.php`. For example, in order to spawn a shell, you can use this shellcode:

3. Environment variables related to the name of the user or even the path to the current working directly can cause an exploit to be unreliable. One technique for building a robust exploit is to use a series of NOP instructions (0x90) to form a NOP slide:

4. Running a program under gdb disables its setuid privileges. In order to read the various ‘flag.txt’ files you must come up with an exploit that works when the program is run directly from the shell.

5. The whoami command is helpful if you are not sure whether you have successfully launched a shell with extra privileges.

4 Task 3: cto.snote (25 pts)

On the same SECSUREHELL system, the Chief Technology Officer (CTO) of DJB has developed a more secure version of snote to make secret notes by adding a buffer-overflow checking routine. Your goal is to, again, read flag.txt by exploiting cto.snote. (Hint: you might be interested in note function.)
5 Task 4: ceo.snote (25 pts)

The Chief Executive Officer (CEO) of DJB is worried about hacking and has enabled Address Space Layout Randomization (ASLR) on the SUPERSECURESHELL system. The CEO has also improved the buffer-overflow checking routine. Your goal is to read flag.txt by exploiting snote.

NOTE 1: For the reversing writeup criterion, describe why your exploit for cto.snote does not work for ceo.snote. Explain the purpose of ASLR and its effect on achieving a successful exploit.

NOTE 2: Your exploits for cto.snote and ceo.snote should be different. The main difference would be figuring out stack addresses by using gdb vs. bypassing ASLR altogether. DO NOT brute-force the system.

6 Task 5: chair.snote (25 pts)

Your last task is to read the Chairman’s (chair) flag.txt message by exploiting snote. The chairman is also concerned about hacking and so the chairman copied CFO’s snote to the SUPERSECURESHELL system, made a few modifications, and further made the stack non-executable. Good luck on your final task!

NOTE: For the reversing writeup criterion, describe why your exploit for cfo.snote does not work for chair.snote. Explain the purpose of a non-executable stack and its effect on achieving a successful exploit. Also, explain whether this binary is vulnerable to a stack overflow.

Tips:
1. The Global Offset Table, which contains the addresses of dynamically linked functions, is not placed in a randomized location by ASLR and can be viewed with readelf -r.
2. The short write variation of a format string attack drastically reduces the amount printed to standard output.