CS4238 Assignment 1: Buffer Overflow Attacks on 32-Bit and 64-Bit Programs

Due Date: Sunday, 3rd Mar, 2024 23:59 SGT
Points: 150

1 Instructions and Deadline

1.1 Instructions

This is an **individual assignment**. You MUST finish the assignment and report **independently**. Note that your report may be checked with the available anti-plagiarism service. By correctly answering all the questions in this assignment, you will get the possible **150 marks**. This assignment is worth **15%** of your final marks.

For your submission, please do the following steps by replacing `a0000000x` with your NUS student number:

1. Create a directory on your computer named by your student number (`a0000000x`), with two subdirectories: `code` and `report`.
2. Copy your code and data to the directory `a0000000x/code`, and copy your report to the directory `a0000000x/report`. The report must be submitted in PDF format. The report should also contain your name, student number, and email address on its first page.
3. Run the following command in the parent directory of `a0000000x` in order to generate your submission package file: `tar zcvf a0000000x.tar.gz a0000000x`.
4. Submit the `a0000000x.tar.gz` generated in the previous step to Assignment section on Canvas.

Deadline: Do submit your package file by **Sunday, 3rd Mar 2024, 23:59 SGT**. There will be **no deadline extensions**, and there will be **penalties for late submissions** as follows:

- Late up to 3 days: You will be evaluated for a maximum of 80% of the total points.

2 Assignment Mission

2.1 Objective

The **learning objective** of this assignment is for you to gain the first-hand experience on buffer-overflow vulnerability and exploitation on both 32-bit and 64-bit programs.

2.2 Set-Up

Please use **64-bit Ubuntu 20.04** Linux system for this assignment. Please note that docker containers do not work for this assignment although any Ubuntu VM should work. If you want, you can also use the SEED Ubuntu 20.04 VM, which is mentioned in the attached accompanying PDF document titled *Buffer Overflow Attack Lab (Set-UID Version)*. If you plan to use any other setup, please confirm the feasibility of the setup with the instructors before starting.

Special Note for Mac M1/M2 Users: Ensure that the Ubuntu/Linux image you are using is the x64 (i.e., x86-64bit) version. If your image is an ARM64 image, the assembly instructions and addresses will differ significantly and you may not be able to use the support documents effectively.

Starter Files: This assignment requires you to perform buffer overflow on a given program. The program files are packaged into `Labsetup.zip`, uploaded to Canvas.

Makefile: Please do not proceed before you make the following changes to the Makefile. You must change the below lines in the `code/Makefile` file:

```
L1 = 200
L2 = 150
L3 = 300
L4 = 10
```

Shell Prompt: Additionally, do **personalize your shell prompt** to show your matric no (e.g. `a0000000x`) in your screenshots by executing:

```
export PS1="a0000000x@\W\$ "
```

Your marks *may* be deducted if your prompts in your attached scrernshots do not show your matric no. You can refer to `https://phoenixnap.com/kb/change-bash-prompt-linux` if needed.

2.3 Tasks Assigned and Grading Scheme

Please refer to the accompanying PDF document titled *Buffer Overflow Attack Lab (Set-UID Version)*, which is attached together with this brief. Complete the following required tasks **as labeled in the accompanying PDF document**, and check the accompanying grading criteria given below. **Please note that the accompanying document has many more tasks (e.g., Task 4) than what is required for Assignment-1. You do not have to perform those tasks.**

Make options for compilation: For Task 2, you can just test overflowing the two following target programs: stack-L1 and stack-L3. For Task 3, you just need to attack stack-L1, since stack-L2 is for Task 4 which is not included in our A1. For Task 5, you just need to attack stack-L3, since stack-L4 is for Task 6 which is not included in our A1 either. For Task 7, using your extended shellcode, you need to attack stack-L1 (similar to Task 3) & stack-L3 (similar to Task 5). Note that modification of both L2 and L4 values in the Makefile is actually not needed for our Assignment 1, since they should be relevant only to stack-L2 and stack-L4, respectively.

What to submit? You need to submit a detailed lab report, with screenshots (with customized shell prompt as mentioned previously), to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.

2.3.1 Task 1 (20 marks)

Run `a32.out` (32-bit) and `a64.out` (64-bit) based on the given shellcode. Do attach screenshots from running them, and describe your observations.

2.3.2 Task 2 (20 marks)

Compile the given vulnerable program, and run it with a badfile consisting of a long string. Do attach a screenshot, and describe your observations.

Note that you need to use the given `Labsetup.zip`, whose values of $L1$, $L2$, $L3$ and $L4$ have been modified. For this task, you should use the Level 1 configuration (i.e., make stack-L1).

2.3.3 Task 3 (30 marks)

For this task, you should use the Level 1 configuration (i.e., make stack-L1). Launch attack on the 32-bit program, and do explain the following:

1. Smashed stack layout explanation (10 marks): Please explain the stack layout of the target program to smash (using a diagram), and not the stack layout in general.
2. Exploit code explanation (10 marks): Explain how you craft your string to overflow the victim program, including how you find the address of the saved return address in the call stack, and set/guess the target address (the one used to overwrite the original saved return address). Include your helper code/script as well as screenshots to explain your exploitation.
3. Getting the shell (10 marks): Provide a screenshot showing that you can successfully obtain the shell.

2.3.4 Task 5 (40 marks)

For this task, you should use the Level 3 configuration (i.e., make stack-L3). Launch attack on the 64-bit program, and do explain the following:

1. Smashed stack layout explanation (10 marks): Please explain the stack layout of the target program to smash (using a diagram), and not the stack layout in general.
2. Exploit code explanation (10 marks): Explain how you craft your string to overflow the victim program, including how you find the address of the saved return address in the call stack, and set/guess the target address (the one used to overwrite the original saved return address). Include your helper code/script as well as screenshots to explain your exploitation.
3. Challenge-solution explanation (10 marks): Note that, as mentioned in the accompanying PDF, compared to attacks on 32-bit machines, attacks on 64-bit machines is more difficult. This is due to the address that may contain zero bytes. Do explain how you solve this issue.
4. Getting the shell (10 marks): Provide a screenshot showing that you can successfully obtain the shell.

2.3.5 Task 7 (40 marks)

Explain how you defeat dash's countermeasure on both `a32.out` and `a64.out` by performing:

1. Shellcode extension (15+15 marks): Please show how you extend your shellcode so that the protection in bash can be bypassed.
2. Getting the root shell (5+5 marks): Provide a screenshot showing that you can successfully obtain the root shell.

3 Queries

If you have any queries regarding this assignment, do email your instructors `prasanna@comp.nus.edu.sg` with the following subject prefix ”`[CS4238 A1]:`”.

Good luck, and have fun with your assignment tasks!

— End of Assignment —