

P3: Buffer Overflow

10/24/2023

95/100 Points

Offline Score:

95/100 Add CommentAnonymous Grading: **no**

10/11/2023

▼ Details

[\[Helpful Command Line Utilities for P3 \(https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3\)\]](https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3) [\[Lab 8: Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)\]](https://canvas.cornell.edu/courses/54972/assignments/522706) [\(https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3\)](https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3)

Reminders:






- This is an individual project. Do not show your work to anyone or look at the work of anyone (in the class or online).
- While you may collaborate on the [Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)](https://canvas.cornell.edu/courses/54972/assignments/522706) Lab with other students you should submit your own solution to that lab for this project, written *only by you*.
- It is recommended that you complete the [Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)](https://canvas.cornell.edu/courses/54972/assignments/522706) Lab before starting the project since that contains details on how to set up your environment, and hints for the first problem of the project.

Overview






The goal of this project is to get intimately familiar with the layout and use of call stacks, C memory safety, debugging, assembly, and reverse engineering. As a side benefit, we hope to raise your awareness of computer security issues. To this end, you will write exploits to break programs we give you and cause them to execute whatever code your heart desires, or leak their own memory contents.

WARNING: These kinds of friendly hacking challenges have a long history, and hacking skills are priceless, as they reflect a deep understanding of the operation of a computer system. But you must be responsible and use your skills wisely. Taking over machines or hacking the Internet carries stiff penalties (unless you are doing so *ethically*), is a sure-fire way to get expelled from Cornell, interferes with other people's lives, and is a waste of your talent. It is also plainly wrong.







How long did you (personally) spend working on **q1** of this project?

0-4 hours	115 respondents	39 %	 ✓
5-10 hours	131 respondents	44 %	
11-20 hours	42 respondents	14 %	
21-30 hours	6 respondents	2 %	
31+ hours	3 respondents	1 %	

How long did you (personally) spend working on **q2** of this project?

0-4 hours	114 respondents	38 %	 ✓
5-10 hours	146 respondents	49 %	
11-20 hours	30 respondents	10 %	
20-30 hours	4 respondents	1 %	
31+ hours	3 respondents	1 %	

How long did you (personally) spend working on **q3** of this project?

0-4	190 respondents	64 %	 ✓
5-10	89 respondents	30 %	
11-20	14 respondents	5 %	
21-30	2 respondents	1 %	
31+	1 respondent	0 %	
No Answer	1 respondent	0 %	

Setting up your Environment

The **[Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)](https://canvas.cornell.edu/courses/54972/assignments/522706)** Lab contains instructions for how to set up your environment for this project, and the problem presented in the lab is the first program you must break for this project.

What to Submit

You will submit an exploit file for each of the three programs you are exploiting. We will run these on the autograder as detailed in each of the three questions, on the server corresponding to your netid.

You will also submit a text document for each of the three programs that explains:

- The vulnerability in the original source code
- How you crafted an input that exploited the vulnerability

We will include more details for each of these points in each of the 3 question subsections.

The Story

In this project, you will exploit several users on the server we've given you.

This means that you will exploit vulnerabilities in code either available or running on this machine to take over these users' accounts, or otherwise leak secret values they're trying to protect.

You have already been introduced to buffer overflow vulnerabilities in the lab. The remainder of the project will involve:

1. Executing another buffer overflow attack that requires slightly more careful inputs
2. Executing a memory *safety* vulnerability that causes the program to leak a secret to the user when it should not do so

The only way you get to exploit these programs is by providing them some carefully crafted input. You may not modify, recompile, or otherwise alter the original programs to execute your exploit (however you may do whatever you wish as part of reverse engineering, exploration, or other discovery -- only your final exploit is limited).

Q1: alphatrouble

The details for this problem are included in the [Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)](https://canvas.cornell.edu/courses/54972/assignments/522706) Lab.

What to submit:

- A file called "egg" that we will execute from the lab08 home directory with the command `(./egg && cat) | invoke simple`
- A pdf called "explanation.pdf" that explains the vulnerability in simplebuffer.c and your process for exploiting it.

Your explanation file should describe:

1. what each part of the input string your exploit produces is for
2. how you constructed the specific input (e.g., choosing particular addresses)
3. Evidence for how you came up with (1) and (2) -- which can be EITHER
 - GDB output

OR

- a depiction of the stack before and after your exploit (similar to the hints we gave you in lab) - **with addresses and values taken from your actual execution**

Q2: betastruggle

For questions 2 and 3, you will need to login to 3410containers.cs.cornell.edu as the "proj3" user, with the same port you used before (your password will be different from the "lab08" user).

You will find the resources for this question in the directory named `beta` inside the proj3 user's home directory.

This program involves a slightly more subtle vulnerability, and more difficult input crafting; otherwise, your task is the same. You must take over the user "betastruggle" by exploiting a buffer overflow vulnerability in the executable "size". There are more useful details in the README file supplied in the server.

Just like part 1, we will execute your exploit (in the egg file), with the command: `(./egg && cat) | invoke size`

What to Submit:

This is basically the same as part 1, we would like you to submit:

- A file called "egg" that we will execute from the proj3/beta directory with the command `(./egg && cat) | invoke size`
- A pdf called "explanation.pdf" that explains the vulnerability in sizeconfusion.c and your process for exploiting it.

Your explanation file should be similar in content to that from q1 (i.e., it includes either depictions of the stack, or gdb output that explain how you crafted your input).

Q3: gammaobstacle

This question and the related files can be found in the "gamma" folder of the proj3 user's home directory.

Unlike q1 and q2, this problem is not centered around a buffer overflow vulnerability; instead you are going to get a service to leak some of its secret memory.

This kind of vulnerability is more generally categorized as a "memory safety vulnerability" where memory is not accessed in the intended manner, leading to unintended consequences (buffer overflows are of course also memory safety vulnerabilities).

In this question, the "gammaobstacle" user is running a guessing game that is listening on one of the server's local network ports.

When someone sends a string to this port, the game responds saying either that "they guessed the secret right" or they "guessed the secret wrong".

We have supplied you with their code for the guessing game - your job is to write a script that prints out the game's secret, just by querying the game itself!

We have also supplied you with the beginning of a script that shows you how to send inputs to the guessing game.

You'll find more tips in the README file of the gamma directory.

What to Submit:

- A file "springLeak" that, when executed, prints out the game's secret. **NOTE: YOU MUST EXPLOIT THE GAME IN YOUR SCRIPT - IF YOU HARDCODE THE SECRET IT WILL FAIL THE AUTOGRADER. WE WILL RUN THIS WITH A DIFFERENT SECRET VALUE WHEN WE AUTOGRADE.**
- A pdf called "explanation.pdf" that explains the vulnerability in server.c and your process for exploiting it.
 - Unlike the prior parts, you don't need a super detailed explanation of stack/memory state since you don't actually need to execute the program.
 - However, you should have some clear explanation for how you crafted your exploit and why it works.

Other Useful Tips

Check out the [Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)](https://canvas.cornell.edu/courses/54972/assignments/522706) document, which includes a ton of useful tips. See also [Helpful Command Line Utilities for P3 \(https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3\)](https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3).

We're here to help. Take advantage of our office hours if you are stuck.

For an entertaining (and a somewhat dated) read on buffer overflow attacks, check out:

Aleph One. Smashing the Stack for Fun and Profit. *Phrack Magazine*, 7(49), November 1996.

<http://www.phrack.org/issues.html?issue=49&id=14> ⇨ <http://www.phrack.org/issues.html?issue=49&id=14>

And finally, to reiterate: a friendly hacking challenge can be fun, and hacking skills are invaluable for working with real systems. But you must be responsible for your own behavior. We are **not** giving you free reign to launch attacks on CMS, fellow students' machines, or any anything else. Such behavior is unethical and most likely illegal as well.

Project Skeleton Files

Here we have included the skeleton code files for each of the 3 project questions for your reference. These are exactly the same as the original files in the server, but in case you overwrote or deleted the originals, you can find them here.

- [q1_egg \(https://canvas.cornell.edu/courses/54972/files/8369475?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369475?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369475/download?download_frd=1) - this is the original **egg** file from the lab08 directory (question 1)
- [q2_egg \(https://canvas.cornell.edu/courses/54972/files/8369487?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369487?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369487/download?download_frd=1) - this is the **egg** file from the proj3/beta directory (question 2)
- [springLeak \(https://canvas.cornell.edu/courses/54972/files/8369476?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369476?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369476/download?download_frd=1) - this the skeleton file from the proj3/gamma directory (question 3)

Project Vulnerable Files

Here we have the source code for the *vulnerable* files you are meant to exploit. These are purely for your reference or convenience and you can also just read the ones included in the server.

- [simplebuffer.c \(https://canvas.cornell.edu/courses/54972/files/8369484?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369484?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369484/download?download_frd=1) - the vulnerable source code in lab08 (question 1)
- [sizeconfusion.c \(https://canvas.cornell.edu/courses/54972/files/8369483?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369483?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369483/download?download_frd=1) - the vulnerable source code in proj3/beta (question 2)
- [server.c \(https://canvas.cornell.edu/courses/54972/files/8369481?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369481?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369481/download?download_frd=1) , [server.h \(https://canvas.cornell.edu/courses/54972/files/8369482?wrap=1\)](https://canvas.cornell.edu/courses/54972/files/8369482?wrap=1) ↓
(https://canvas.cornell.edu/courses/54972/files/8369482/download?download_frd=1) - the vulnerable source code in proj3/gamma (question 3)

Server Rebuilds

In the event of a server rebuild or shutdown, you'll receive a notification that the remote host identification has changed and you will not be able to connect to the server. If this occurs, remove the entry for the server in ~/.ssh/known_hosts.

Please note that only the original project files will be saved on the server in the event of a rebuild or shutdown. Any extra files you create on the server will not be saved.

EDStem Guidelines

- A **big part** of this project is reading and understanding C code to figure out what's going on and what's wrong. **DON'T SPOIL THE FUN FOR YOUR CLASSMATES.**
 - Exploiting vulnerabilities is a lot like a puzzle or a riddle - if you give away the answer, there's no way to "unlearn" it and have the same experience.
 - We will consider "giving away hints" as AI violations if they clearly violate this guideline. It's better to be safe and not respond, or wait for an instructor to respond first.
- **HOWEVER**, this project also comes with a lot of tool and scripting use that many of you are likely uncomfortable with (GDB, byte manipulation, network tools, etc.). **You should definitely feel free to post and respond to questions about "how do I do Y with tool X" on ED.**
 - This will be the *best use of Ed for P3 - but please try to not give anything away about the vulnerabilities that isn't already included in the project documentation when making these posts.*

[\[Helpful Command Line Utilities for P3 \(https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3\)\]](https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3) [\[Lab 8: Intro to Buffer Overflow \(https://canvas.cornell.edu/courses/54972/assignments/522706\)\]](https://canvas.cornell.edu/courses/54972/assignments/522706) [\[https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3\]](https://canvas.cornell.edu/courses/54972/pages/helpful-command-line-utilities-for-p3)



<https://canvas.cornell.edu/courses/54972/modules/items/2052983>



<https://canvas.cornell.edu/courses/54972/module>