CPSC 526 Winter 2025

Assignment 1

Due date: **posted on D2L**
Weight: 10% of your final grade.
Group work is allowed but not required. Maximum group size is 2 students.

For this assignment you are going to find a vulnerability in a badly implemented server program and then exploit it to break its security. You will also fix the vulnerability.

Running the server program

Download the starter code for this assignment and compile the server:

```
$ git clone https://gitlab.com/cpsc526/w25/secretserver.git
$ cd secretserver
$ make
```

The compiled server is called `secretServer`, and it will accept 3 command line arguments: a port number, a password and a secret. Once started, the server will listen for new connections on the specified port. When a client connects to the port, the server will read a single line from the client. If that line consists of the correct password, the server will reveal the secret to the client. If the password is incorrect, the server will not reveal the secret phrase.

Here is an example of how to start the server:

```
$ ./secretServer 1234 P4ss 'My favorite number is 42'
Waiting for a new connection...
```

In the above example the server will listen on port '`1234`' for connections, it will expect client to send 'P4ss' as password, and if the client sends the correct password, the server will send it the secret 'My favorite number is 42'.

To connect to the server, you can use the '`nc`' utility program:

```
$ nc localhost 1234
Secret Server 1.0
Password:
```

When the client connects, the server sends the client the text 'Secret Server 1.0'. At this point the server will be waiting for the client to send a password. Below is an example of the client sending the wrong password, and the server reacting by not revealing the secret:

```
$ nc localhost 1234
Secret Server 1.0
Password:hello
Wrong password.
```

If you provide the correct password, the server will reveal the secret phrase:

```
$ nc localhost 1234
Secret Server 1.0
Password:P4ss
The secret is: My favorite number is 42
```

Whether you supply the right or wrong password, the server will eventually close the connection. The server will then continue listening on the original port for a new connection. The only way to terminate the server is to kill it, for example using `<CTRL-C>`.

Task 1 – Exploit the vulnerability in 2 connections

The secret server program has a buffer overflow bug, which can be exploited. A successful exploit will force the server to reveal the secret without knowing the correct password. Figure out how to make the server reveal the secret using 2 separate connections. The first connection will not reveal the secret, but your exploit will modify the server's stored password to a desired value. The second connection will reveal the secret, by simply sending to the server the password you set during the first connection.

Hint: You do not need to send any special characters to the server at all – regular alphabetic characters should suffice. Your exploit should be executed from command line using standard Linux tools, such as `echo`, `printf` and `nc`.

In your `readme.pdf` you should briefly describe your exploit.

You may assume that for this task the `BUFFSIZE` will be set to 32.

Task 2 – Exploit the vulnerability in a single connection

Design an exploit just like above, but in such a way that a single connection will suffice to make the server reveal the secret.

Hint: The data you will need to send to the server will include the `'\0'` character. Here is an example of how to send the string `"abc\0def"` from command line using the bash built-in `printf`:

```
$ printf 'abc\x00def' | nc localhost 12345
```

In your `readme.pdf` you should briefly describe this exploit.

You may assume that for this task the `BUFFSIZE` will be set to 32.

Task 3 – Write a Python code to automatically run the exploit

Finish implementing the `exploit.py` file from the starter code. This script accepts 2 command line arguments: hostname and port. The script is supposed to connect to the secret server at the specified hostname and port, and figure out the secret, by automatically exploiting the server.

Your `exploit.py` will need to work on any server executable compiled with `BUFFSIZE` between 8-128. To accomplish this, your exploit will likely need to make multiple connections to the server, until it succeeds. A correctly implemented exploit will recover the secret nearly instantaneously, and its output will look like this:

```
$ python3 exploit.py localhost 1234
Running exploit on localhost:1234
Secret = My favorite number is 42
```

In your `readme.pdf` you should briefly describe how you implemented your `exploit.py`.

Task 4 – Fix the vulnerability

For this task you will need to fix the server code by removing the vulnerability from `secretServer.c`. During the demo you will need to demonstrate to your TA that you fixed the vulnerability and be able to explain how you fixed it. You will also submit the fixed source code to D2L in a file called `fixedSecretServer.c`.

Your fixed code in `fixedSecretServer.c` should be as close to the original `secretServer.c` as possible.

Submit `readme.pdf` file

You must submit a `readme.pdf` file with your assignment. At the beginning of this file, you must declare whether you worked alone or with a partner. If you worked with a partner, you need to specify the name and student ID of your partner. The rest of your `readme.pdf` will contain short explanations (~1 paragraph) of how you completed each of the tasks above.

Additional notes

- All exploits and all code you submit for this assignment must run on the departmental cslinuxlab machines.

- You must demo your solutions to the TA. During the demo, you will need to explain how your solutions work in order to get marks. Demo times will be arranged by your TA.
- You are allowed to work on this assignment with another student (max. group size is 2 students). But beware that during the demo you will be asked to demonstrate your familiarity with everything you submit. If you are unable to explain to your TA your solution(s), you may lose (possibly all) marks.
- Whether you work alone, or with a partner, every student needs to submit all files on D2L.

Marking

Category	Marks
Task 1 – describe the exploit in readme.pdf, and during your demo, show to your TA that you can execute the exploit from command line even if the server is compiled with a different BUFFSIZE.	30
Task 2 – describe the exploit in readme.pdf, and during your demo, show to your TA that you can execute the exploit from command line even if the server is compiled with a different BUFFSIZE.	30
Task 3 – your exploit.py works for any value of `BUFFSIZE` value, and you are able to explain to your TA how it works, and you correctly describe it in your readme.pdf.	20
Task 4 – Demonstrate the lack of vulnerability in your fixed server (demo, readme and source).	20

Submission

Every student must submit all files to D2L, even if you worked in a group:

- `exploit.py` containing your automated exploit program;
- `fixedSecretServer.c` with the vulnerability removed;
- `readme.pdf` as described above.

You must submit the above to D2L to receive any marks for this assignment.

General information about all assignments:

- All assignments are due on the date listed on D2L. Late submissions may receive penalties, as described in the outline.
- Extensions may be granted only by the course instructor.
- After you submit your work to D2L, verify your submission by re-downloading it.
- You can submit many times before the due date. D2L will simply overwrite previous submissions with newer ones. It is better to submit incomplete work for a chance of getting partial marks, than not to submit anything. Please bear in mind that you cannot re-submit a single file if you have already submitted other files. Your new submission would delete the previous files you submitted. So please keep a copy of all files you intend to submit and resubmit all of them every time.
- Assignments will be marked by your TAs. If you have questions about assignment marking, contact your TA first. If you still have questions after you have talked to your TA, then you can contact your instructor.
- All programs you submit must run on the departmental cslinuxlab machines. If your TA is unable to run your code on the Linux machines, you will receive 0 marks for the relevant question.
- Assignments must reflect your group's own work. For information on plagiarism, cheating and other academic misconduct, check the information at this link: http://www.ucalgary.ca/pubs/calendar/current/k.html .
- We may use automated similarity detection software to check for plagiarism. Any cases of detected plagiarism or any other academic misconduct will be investigated and reported.