**CPS 475/575 - Secure Application Development**

**Spring 2020**

Dr. Phu Phung

Department of Computer Science,  
University of Dayton

Lab 2 - Secure & Client Socket Programming in C

Released: Lecture 5, Tuesday, January 28

Deadline: 11:55 PM Tuesday, February 4

Introduction

In this lab, you will do hands-on experiments to understand the vulnerabilities of the C programming language due to memory flaws (format string, buffer overflow). You will also develop your first (simple) application in C that connects to a TCP server at an address and port number given by the user. The program will get a message from the user and send to the server, and then display data received from the server. You will revise the code further to develop a simple yet secure HTTP client program. You will be asked to review your application to ensure that the vulnerabilities in the previous tasks do not exist in your application.

Please note that in addition to submitting your code of this lab on Isidore, you must have all of your code in your private repo and push to bitbucket. **Include the URL of this folder at the beginning of your report**. For example, in my case, the URL is:  
<https://bitbucket.org/phu-udayton/secad-pphung1/src/master/labs/lab2/>

**You can use the Tasks and questions but DO NOT copy all lab instructions to your report. Revise or write in your own words and understanding instead.**

This lab has three tasks with 30 points:

Task 1 (10 points). Exploiting and fixing vulnerable C programs.

Task 2 (6 points). Develop a simple TCP client application in C.

Task 3 (14 points). Develop a simple yet secure HTTP client application in C.

*(Notes: Task 1 should be done in Lecture 5; Task 2 and Task 3 should be done in Lecture 6. This lab is to ensure you have gained all the hands-on steps in the lectures).*

There is an extra credit task that you can do to gain more experience and earn 1% extra for your final grade.

Report and submission

You need to write your report to submit via Isidore.  Your report must be submitted in PDF file (reports not in PDF format will be returned without grading). Your report MUST have the course number, course name, the instructor, your name and UD email. You MUST organize your report as in this instruction, for example:

Task 1. xxx

1. Yyy

You need to demonstrate your experiments by screenshots, therefore, **it is important to capture screenshots when you perform the lab**. **Your screenshots must have a short description or a caption to explain the task.**

I recommend you to use Google Docs to write your report as it is convenient to include the screenshots for your report. **Do NOT copy the lab’s instructions in your report.** From this lab, **you are required to provide your code as plaintext** (not screenshot) in your report. *If you use Google Docs to write the report, you can use an add-on such as “Code Blocks” (*[*https://gsuite.google.com/marketplace/app/code\_blocks/100740430168*](https://gsuite.google.com/marketplace/app/code_blocks/100740430168)*) to format and indent your code.*

Task 1 (10 points). Exploiting and fixing vulnerable C programs

Follow the steps in **Lecture 5** to develop the myecho0.c and myecho.c programs save them to the labs/lab2 folder in your private repo.

1. Format string vulnerability
   1. (1 point) Develop the myecho0.c program with the code below (as guided in Lecture 5).   
      A picture containing man, table

      Description automatically generated  
      Compile and run the program to demonstrate the format string vulnerability. Explain the reason and capture the screenshot to demonstrate that.

**Push the file to your bitbucket repo**

* 1. (2 points) Modify the myecho0.c program to avoid the format string vulnerability. Justify your solution and capture the screenshot to demonstrate that the revised application does not have the vulnerability. Hint: redo the above attack and show that it failed.  
     **Push the new update to your bitbucket repo and provide the commit link here in your report.**

Capture the screenshot on bitbucket that compares the old version and the revised of the myecho0.c program to include in your report.

1. Buffer overflow vulnerability
   1. (1 point) Copy the myecho0.c program above and rename it to myecho.c. Modify the code as below (as guided in Lecture 5).  
      A screen shot of a computer

      Description automatically generated  
      Compile and run the program to experience the buffer overflow vulnerability. Explain the reason and capture the screenshot.

**Push the file to your bitbucket repo**

* 1. (2 points) Modify the myecho.c program to avoid the buffer overflow vulnerability. Justify your solution and capture the screenshot to demonstrate that the revised application does not have the vulnerability (Hint: redo the above attack and show that it failed.). For protection, you can either use the compiler options (see Lecture 5 handouts) or just remove the vulnerable function.   
     **Push the new update to your bitbucket repo and provide the commit link here in your report.**

Capture the screenshot on bitbucket that compares the old version and the revised of the myecho.c program to include in your report.

1. Identify possible vulnerabilities in a binary program.

You are given a binary application “myecho” in the course repository *(Note: this program is already compiled and executable)*. To get this program, pull the course repo (see Lab 0 if you are not sure how to do this). This application is located in labs/lab2 folder in the course repository. This application takes an input string from the user and prints out that input (similar functionality as myecho0.c and myecho.c programs in (a) and (b)).   
You can run the program directly as demonstrated below:  
A close up of a sign

Description automatically generated  
In this task, answer and demonstrate the following questions:

* 1. (1 point) Is this application vulnerable to the format string vulnerability? How do you know that? Capture the screenshot to demonstrate this.
  2. (3 points) Is this application vulnerable to the buffer overflow vulnerability? What is the size of the buffer? How do you know these? Capture the screenshot to demonstrate this.

Task 2 (6 points). A simple TCP client application in C.

In this task, you develop your first application in C that connects to a TCP server at an address and port number given by the user (A simple TCP client application). A detailed tutorial for this development is given in Lecture 6. More reading for this task can be found at <https://beej.us/guide/bgnet/> (Beej's Guide to Network Programming Using Internet Sockets).

The program will get a message from the user and send it to a TCP Echo Server, and then display data received from the server. You will test this version with the EchoServer.java program provided in Lecture 3. Follow the steps in Lecture 6 to develop this application.

1. (3 points) Test your program with the EchoServer.java and capture the screenshot to demonstrate this. Your screenshot should be similar to the one below.  
   A screenshot of a newspaper

   Description automatically generated
2. (3 points) Include your code in text (not screenshot) in your report.

Task 3 (14 points). Develop a simple yet secure HTTP client application in C.

In this task, you will develop a simple HTTP client program that connects to a web server and sends a minimal HTTP request to the webserver and then prints out the data received from the server.  Follow the steps below (in Lecture 6) to develop the program:  
A screenshot of a cell phone

Description automatically generated

1. (3 points) Test your program with the webpage <http://academic.udayton.edu/secad.html>  (as performed the experiments in Lab 1):  
   Servername: academic.udayton.edu  
   Path: /PhuPhung/secad.html

Capture the screenshot similar to below to demonstrate the program in your report.  
A screenshot of text

Description automatically generated

1. (3 points) Include your code (myhttpclient.c) in text (not screenshot) in your report.
2. (3 points) Examine your program with Wireshark

Perform the Wireshark network traffic examination as in Lab 1 - Task 1 (instead of using a browser, use your myhttpclient.c program). Capture the screenshots of HTTP Request and HTTP Response and identify if there is any difference in the messages compared with Lab 1 - Task 1.

1. Security analysis
   1. (2 points) What are the input data in your program? Hints: List the data that are from outside your program (Hints: there are 4 sources of the inputs).
   2. (3 points) Review to examine if the data can cause **any security problems**. Analyze your program to ensure that there is no format string or buffer overflow vulnerabilities, i.e., try to run the attacks and demonstrate they are failed. Explain how did you avoid these vulnerabilities in your code?

Your Lab 2 Report should be done here.

Task 4 (1% extra credit). Develop a secure HTTP client application in C to download a file from a URL.

This is an extra credit lab for students who are highly self-motivated. Besides getting 1% extra credit for your final grade, you will learn more hands-on and programming skills in this exercise.

This task is an extension of Lab 2 - Task 3, which is to develop a simple yet secure, robust, and fully functional HTTP client application to download load a file in a given URL. The application gets a URL from the user, e.g., from program arguments and connects to the corresponding webserver to send an HTTP request. If the server replies 200 OK, the application saves the received to a file accordingly, otherwise, just display a meaningful message from the server. The developed program is a simple downloading program similar to, e.g., curl (<https://en.wikipedia.org/wiki/CURL>)

Hints and tutorials are available on Isidore for this task. This extra credit lab has 100 points with below distribution:

1. Establish a connection to the server (10 points)
2. Construct a correct HTTP request and send to the server (10 points)
3. Handle HTTP response code correctly (10 points)
4. Store data to a corresponding file (15 points), i.e., the filename to save must be extracted from the URL. If there is no filename in a given URL, save it to index.html
5. Files are displayed correctly (4 test cases x 5 = 20 points). The test cases are:
   1. http://academic.udayton.edu/PhuPhung/secad/
   2. http://beej.us/guide/bgnet/html/single/bgnet.html
   3. http://academic.udayton.edu/PhuPhung/red\_blue\_chapel\_logo\_111610.jpg
   4. http://beej.us/guide/bgnet/pdf/bgnet\_USLetter.pdf
6. Identify the possible risks and ensure no possible vulnerability or defect (15 points)

*The hints in the tutorial may be insecure and non-robust. You need to do the review for any inputs/ API calls and revise the code as early as possible. Write these analyses as comments in your code accordingly.*

1. Handle the input to get hostname and path [can be done after the main part] (10 points)
2. Security tests
   1. (5 points) No Input
   2. (5 points) An input with, e.g., 3000 characters as the URL

You only need to submit the source code (change the filename to .txt, e.g., myhttpclient.txt) on Isidore (Lab 2 - Task 4 - Extra Credit) and demo your program to the instructor for this extra credit lab. The deadline is to be determined.

Some demonstrations are shown in the screenshots below.

A close up of a newspaper

Description automatically generated

Test cases of successful download an HTML file and invalid inputs

A close up of text on a white background

Description automatically generated

A sign on the side of a building

Description automatically generated

A test case of an image URL