CSE 365: Introduction to Information Assurance

SCHEDULE SYLLABUS & POLICY ASSIGNMENTS

Assignment 1

Assignment 1 is *due 1/24/21* on or before 11:59:59pm MST.

Part 1 (5 points)

Sign up for the course Piazza. We will know if you registered by your name and/or ASU ID. If we have any questions, we will contact you directly.

Once you're on Piazza, register for a Gradescope account so that you can submit the rest of the assignment.

Part 2 — Bandit v1 (25 points)

For a future homework assignment you will be hacking on a Linux server. The goal of this assignment is to familiarize yourself with accessing a Linux environment via SSH, along with developing skills on command line interaction and wargames.

First, register for a wechall account. You will need to submit your wechall username so that we can track your progress on the levels. After registering, you will need to link OverTheWire.org to your wechall account by doing the following:

1. Click "Account" on the top of wechall.net
2. Clink on the "Linked Sites" button
3. On the "Select a site" dropdown, select "OverTheWire.org"
4. Then click the "Link Site" button

Now, OverTheWire.org should show up in your list of linked sites, and we will be able to track your progress on Bandit from your user profile.

Then, the goal is to solve 5 levels (in other words reach level 6) on the overthewire.org Bandit challenges.

Before you start, be sure to read how to register your bandit progress with wechall and do so. This way, your bandit progress will be captured on wechall, which we will use to grade your progress.

Also, keep track in your README how you solved each level.

Note that Bandit is an open system, and the goal of this assignment is to practice and develop your own skills, so be honorable and do not read walkthroughs.

Submission Instructions

Submit on GradeScope the plain text file README to gradescope.

Note: you must include the following line in your README (replace the `INSERT_WECHALL_NAME_HERE` will your wechall username), or else the autograder won't be able to give you a grade:

```
wechall name: INSERT_WECHALL_NAME_HERE
```

Part 3 — Make this (25 points)

All of the coding assignments in this course (including Part 4 of this assignment) allow you to write your assignment in any programming language. To allow this, you will need to write a Makefile that creates an executable file based on your source code.

In this assignment, you'll practice writing a Makefile for two different types of programs, one a C program that must be compiled, and the other a Python program.

Here are some Makefile resources:

- https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html
- https://web.stanford.edu/class/archive/cs/cs107/cs107.1174/guide_make.html
- https://www.gnu.org/software/make/manual/make.html
- http://mrbook.org/blog/tutorials/make/

C Program Makefile

The C Program Makefile must compile the file `c_program.c` into the executable `c_program`.

Assume that your C Program Makefile is called `Makefile.c` and is in the same directory as the `c_program.c`, and therefore, when you run `make -f Makefile.c` in that directory, the compiled program `c_program` is created using `gcc`.

Also, your C Program Makefile must recompile the binary when `c_program.c` changes.

Python Makefile

The Python Makefile must use the file `python_program.py` to create an executable file called `python_program`.

Assume that your Python Makefile is called `Makefile.python` and is in the same directory as the `python_program.py`, and therefore, when you run `make -f Makefile.python` in that directory, your Python Makefile will create a `python_program` executable.

Also, your Python Makefile must recreate `python_program` when `python_program.py` changes.

There are several ways to approach this (because there's nothing to compile). For instance, you can take advantage of the shebang functionality, which allows a file that can be interpreted to be executed. This requires that the file is executable (`chmod +x filename`).

Submission Instructions

[Submit on GradeScope][https://www.gradescope.com/courses/228340] the Python Makefile as `Makefile.python` and the C Makefile as `Makefile.c`.

Part 4 — Commands (45 points)

One of the ways that programs receive input from users is through command line arguments. We will also use this in future assignments.

Your goal is to write, in any language, a program which first prints out the number of command line arguments and the next line prints them out in reverse order, separated by space (so that the last command line argument is printed first).

The name of your program will be called `command`.

Examples

When your program is executed with the following:

```
./command foo bar
```

It must output exactly:

```
2
bar foo
```

Other examples:

```
./command
```

Output:

```
0
```

Running:

```
./command a b "test input" c d e
```

Output:

```
6
e d c test input b a
```

Implementation

Your program must work on Ubuntu 18.04 64-bit with the default packages installed. If there's a package that you need, please ask on the course piazza and I'll have it installed for everyone. Java is already installed.

You'll also need to write a Makefile that, when the `make` command is run, will create the executable called `command`.

Submission Instructions

Submit on GradeScope your source code, along with a Makefile (called `Makefile`) and a README. The Makefile must create your executable, called `command`, when `make` is ran. Your README file must be plain text and should contain your name, ASU ID, and a description of how your program works.

CSE 365: Introduction to Information Assurance

Adam Doupé and Tiffany Bao

Website Design Acknowledgement: Carolina Zarate and Zilin Jiang