Login to Linux server

- Email jeff@cse.tamu.edu with your UIN to get your username and password
- You can then login via `ssh username@programming-llm.org`

Mock Challenge Project

The folder `/mock-cp` contains a minimal sample AIxCC ASC Challenge Project.

Your goal is to use LLMs to find and patch security vulnerabilities in the project.

Copy mock-cp to your home directory: `cp -r /mock-cp $HOME/`

Tasks:

1. (4pt) Write Python code to generate two files: a proof of vulnerabilty (POV) `x.bin` and a patch `x.diff`, such that
    - `x.bin` triggers a vulnerability and `x.diff` patches the vulnerability
    - You can use the best LLMs (such as those from OpenAI, Anthropic, Google) for this task
    - Measure the cost and speed
2. (Bonus 1pt) Modify your code to use a local LLM (e.g., llama3.1-8b), and optimize the speed
    - llama3.1-8b is already running on the server by ollama at port 11434, try the following:

```
curl http://localhost:11434/v1/chat/completions -H "Content-Type: application/json" -d '{
     "model": "llama3.1",
     "max_tokens": 28,
     "messages": [
         {
             "role": "system",
             "content": "You are a helpful assistant."
         },
         {
             "role": "user",
             "content": "Where is Texas A&M?"
         }
     ]
}'
```

Submission

- Python code for Task 1 and Task 2 (optional)
- A report that describes your solution and results (including remaining challenges and failures if any)
- Limit your report to three pages with 10pt font size

What's in this repository

`src/` - this directory is where CP source code is loaded, analyzed, and built from.

`run.sh` - a script that provides a CRS with a standardized interface to interact with the challenge project.

`exemplar_only` - this folder contains sample POVs and patches.

Initial Building

```
git -C src/samples reset --hard HEAD

./run.sh -x build
```

How to validate a proof of vulnerability

The following command fails:

- `./run.sh -x run_pov x.bin filein_harness`
- The output contains `ERROR: AddressSanitizer: global-buffer-overflow`

How to validate a patch

The following two commands both succeed:

- `./run.sh -x build x.diff samples`
- `./run.sh -x run_pov x.bin filein_harness`

Sample Usage

The below sample usage mirrors what the challenge evaluator does in the challenge project verification pipeline

```
./run.sh -x build

./run.sh -x run_tests

./run.sh -x run_pov exemplar_only/cpv_1/blobs/sample_solve.bin filein_harness
./run.sh -x run_pov exemplar_only/cpv_2/blobs/sample_solve.bin filein_harness

./run.sh -x build exemplar_only/cpv_1/patches/samples/good_patch.diff samples
./run.sh -x run_pov exemplar_only/cpv_1/blobs/sample_solve.bin filein_harness
./run.sh -x run_tests

git -C src/samples reset --hard HEAD
./run.sh -x build exemplar_only/cpv_2/patches/samples/good_patch.diff samples
./run.sh -x run_pov exemplar_only/cpv_2/blobs/sample_solve.bin filein_harness
./run.sh -x run_tests

git -C src/samples reset --hard HEAD
./run.sh -x build exemplar_only/cpv_1/patches/samples/bad_patch.diff samples
./run.sh -x run_pov exemplar_only/cpv_1/blobs/sample_solve.bin filein_harness
./run.sh -x run_tests

git -C src/samples reset --hard HEAD
./run.sh -x build exemplar_only/cpv_2/patches/samples/bad_patch.diff samples
./run.sh -x run_pov exemplar_only/cpv_2/blobs/sample_solve.bin filein_harness
./run.sh -x run_tests
```