Assignment 5
Hardware Fuzzing
Due date: 12/11/2023 (Monday) at 11:59 pm, CST

Description

In this assignment, your objective is to play the role of a defender (verification engineer) with the goal of detecting vulnerabilities in hardware designs. You are provided with an incomplete hardware fuzzer, TheHuzz, that aims to fuzz the CVA6 processor. You will have to build the missing components of the fuzzer, fuzz the CVA6 processor with it, and report its performance on the CVA6 processor. You will be testing the fuzzer through simulation. This exercise allows you to understand the concept of fuzzing and how it can be used to verify hardware and detect vulnerabilities.

For this assignment, you are expected to

i. You will be given an incomplete hardware fuzzer, TheHuzz, for which you have to design the missing components and fuzz the CVA6 processor with it.

ii. There are four missing components (F1, F2, F3, F4) of the fuzzer:
   a. F1: The coverage calculation function in the feedback engine.
   b. F2: The function to get testcases from the input database.
   c. F3: The testcase selector function in the feedback engine.
   d. F4: The random function in the mutation engine (bonus points).

iii. Appendix A lists the details about each of these functions that you need to complete. Although not required, we recommend you to test the functions you complete before using them with the fuzzer.

iv. Appendix B lists the details about collecting the results and format of the report you will submit for this assignment.

v. For each of the four missing components of the fuzzer:
   a. You have to read and understand the functionality of the missing component from Appendix A.
   b. This component is in the form of one or more functions in the Python source code of the fuzzer provided to you. You can search for the keyword 'EDIT' in these functions that indicates the locations where you need to modify the code.
   c. You will be coding in Python to complete this function.
   d. Update the fuzzer source code with your edits and follow the instructions in Appendix B to collect the results and present them in your report.
   e. You need to add comments to the program files. **Exploits without comments will not be graded. Check an example code snippet in Appendix C with comments inserted.**

Resources required for this assignment

1. Fuzzer setup:

   The fuzzer source code is provided to you along with the files required to fuzz CVA6 processor with it. This processor follows the RISC-V ISA. The source code contains a lot of files in it. You **need not** go through each and every file. Appendix A lists all the relevant files and functions required to understand the functions you will be building. It is sufficient to only check those files.

2. Appendices from this manual:
   a. Appendix A: The details about the missing components of the fuzzer.
   b. Appendix B: Instructions to collect results and writing your report.
   c. Appendix C: Example code with comments.
   d. Appendix D: Setup instructions to fuzz CVA6 processr.

Due date and Deliverables

The due date to submit this assignment is **12/11/2023 (Monday) at 11.59 pm, CST**.

All the Python files you modified to create build the functions should be submitted along with a pdf report as a zipped file with <FuzzLab_lastname_firstname>.zip as the file name in the Canvas.

TA office hours

- 30 Nov 2023, 1 PM to 2 PM and 6 PM to 7 PM (Zoom link)
- 6 Dec 2023, 11 AM to 12 PM and 6 PM to 7 PM (Zoom link)

Rubrics

Maximum points: **100 (+20 bonus point)**

1. Completing functions F1, F2, and F3: 30 points each.
   Of these 30 points:
   a. 10 points for reporting if the function worked correctly or not,
   b. 15 points for the changes made to the code,
   c. 5 points for the screenshots
2. Feedback on the assignment: 10 points.
3. Completing function F4: 20 bonus points.

Note check Appendix B for report submission format.

Appendix: A

This section describes the components of the TheHuzz fuzzer that are required to complete its missing pieces. In addition a lot of low level details are also provided in the form of comments in the corresponding python scripts. We strongly recommend you check these comments along with the information provided in this section for greater understanding.

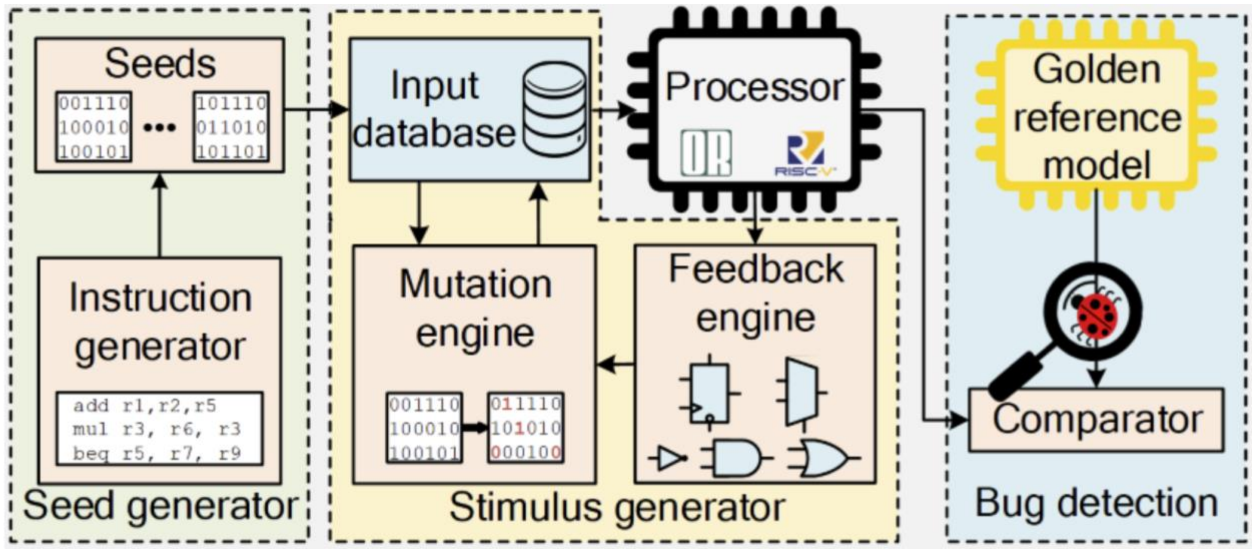


Figure 1 Overview of TheHuzz fuzzer

Check the lecture video taught by professor on hardware fuzzers.

This is the link to the TheHuzz paper (this is only for your reference and you don't need to read the paper to complete the assignment).

F1: The coverage calculation function in the feedback engine
Code location: thehuzz/fuzz.py, "compute_cov_achieved" function

Code details:
This function computes the number of coverage points the fuzzer covered so far and derives the percentage of coverage achieved from it.

F2: The function to get testcases from the input database

Code location: thehuzz/thehuzz_utils.py, "get_testcases_to_sim" function

Code details:
This function provides the required number of testcases from the database to the fuzzer to test the processor.

F3: The testcase selector function in the feedback engine
Code location: thehuzz/fuzz.py, “calc_no_times_to_mut” function

Code details:
This function checks which testcases increased coverage and based on that determines if these testcases should be mutated or not and also how many times to mutate.

F4: The random function in the mutation engine (bonus points)
Code location: thehuzz/prog_mut.py, “my_random” function

Code details:
This function performs the random mutation where the given number of bits are set to random value.

Appendix: B

- For each of the four incomplete functions you will be completing in this lab:
    - Complete all the edits for the function following the instructions in Appendix A and the comments in the python code. **Make sure to include comments when you edit the code. These comments should be your own comments and not copy pasted ones from existing code.**
    - Run the fuzzer (check appendix D for details about setup). Include a screenshot of the terminal showing the command you are using to run the fuzzer.
    - Specify if you were able to successfully run the fuzzer or if the script threw any errors. If the script did not run successfully, try to explain the issue (You will get partial points for the explanation in case your exploit did not work).
    - Include a screenshot of the terminal at the end of fuzzing. This will look like the last screenshot in the Appendix D where it shows the coverage achieved by the fuzzer.

Note that you will need to run fuzzer after each of the completed function and include everything listed above in your report. You wont get full points if you edit all the functions at once and run the fuzzer only once.

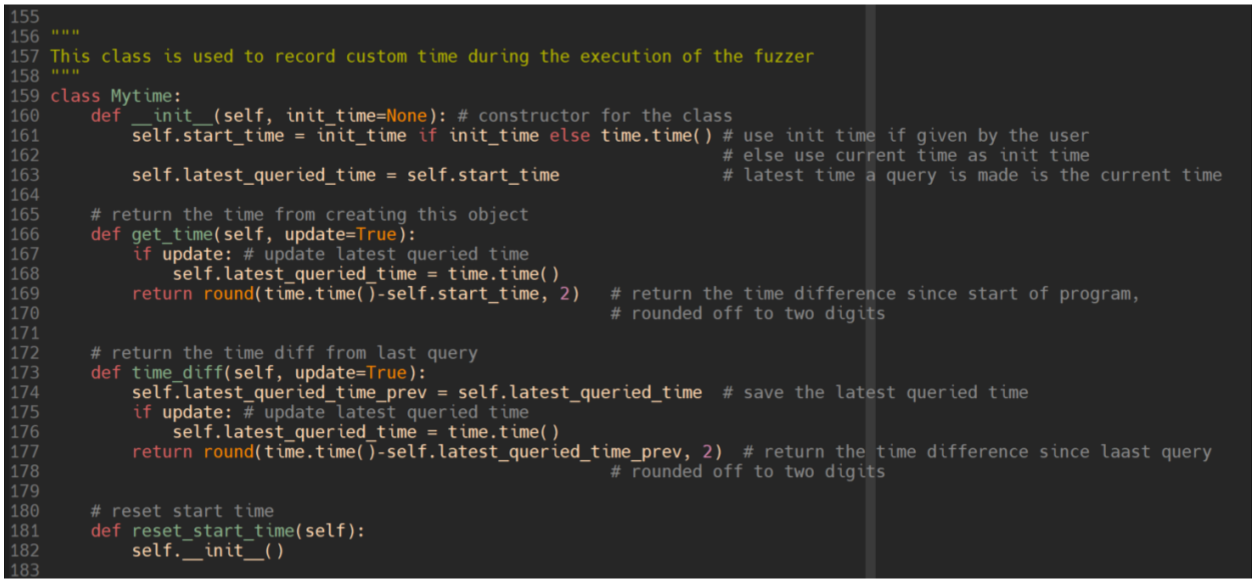
- Also, please include your feedback about this assignment in your report. Any feedback that can improve the lab for upcoming semesters is greatly appreciated.

- All the Python files you modified to create build the functions should be submitted along with a pdf report as a zipped file with <FuzzLab_lastname_firstname>.zip as the file name in the Canvas.

Appendix: C

Example program with comments:

The example program below shows a python program with lots of comments. You do not need to write this many comments in your submission, but you should have enough comments to explain your code.



```python

"""
This class is used to record custom time during the execution of the fuzzer
"""
class Mytime:
    def __init__(self, init_time=None): # constructor for the class
        self.start_time = init_time if init_time else time.time() # use init time if given by the user
                                                                  # else use current time as init time
        self.latest_queried_time = self.start_time                # latest time a query is made is the current time

    # return the time from creating this object
    def get_time(self, update=True):
        if update: # update latest queried time
            self.latest_queried_time = time.time()
        return round(time.time()-self.start_time, 2)   # return the time difference since start of program,
                                                       # rounded off to two digits

    # return the time diff from last query
    def time_diff(self, update=True):
        self.latest_queried_time_prev = self.latest_queried_time  # save the latest queried time
        if update: # update latest queried time
            self.latest_queried_time = time.time()
        return round(time.time()-self.latest_queried_time_prev, 2)  # return the time difference since laast query
                                                       # rounded off to two digits

    # reset start time
    def reset_start_time(self):
        self.__init__()

```

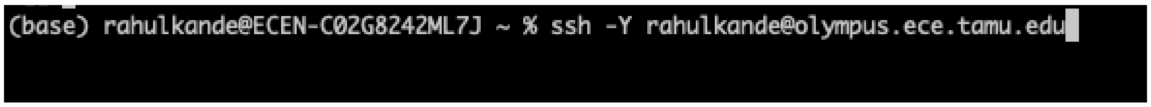
Appendix: D

NOTE: DO NOT SHARE THE SOURCE CODE OF THE FUZZER WITH ANYONE OUTSIDE THE CLASS AS THE CODE IS STILL UNDER THE PROCESS OF LICENSING.

Setup for the assignment:

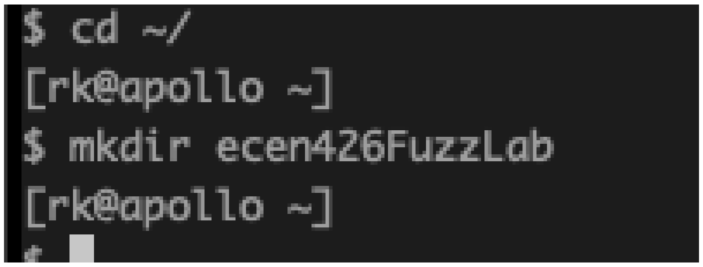
You will be doing the assignment on the TAMU Apollo server. The setup documents are provided in the form of Gitlab repository.

1. Login to the TAMU olympus server:
   a. Download the Mobaxterm software.
   b. ssh into the Olympus server with this command: `ssh -Y <TAMU username>@olympus.ece.tamu.edu.`
   Enter your TAMU password when prompted.



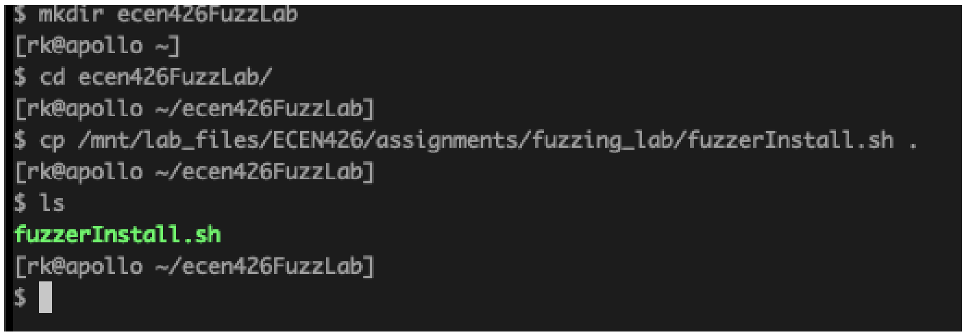
```
(base) rahulkande@ECEN-C02G8242ML7J ~ % ssh -Y rahulkande@olympus.ece.tamu.edu
```

2. Create a folder for the assignment:
   a. `mkdir <some name for the folder>`



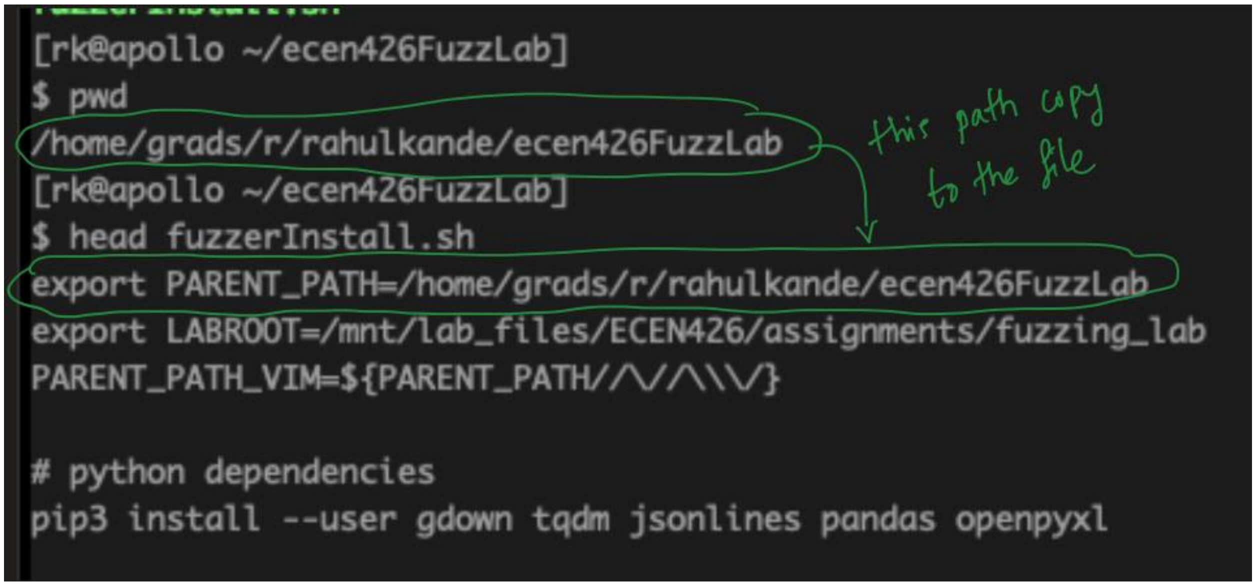
```
$ cd ~/
[rk@apollo ~]
$ mkdir ecen426FuzzLab
[rk@apollo ~]
$
```

3. Enter the folder and copy the install script:
   a. `cd <path to the assignment folder created in previous step>`
   b. `cp /mnt/lab_files/ECEN426/assignments/fuzzing_lab/fuzzerInstall.sh .`



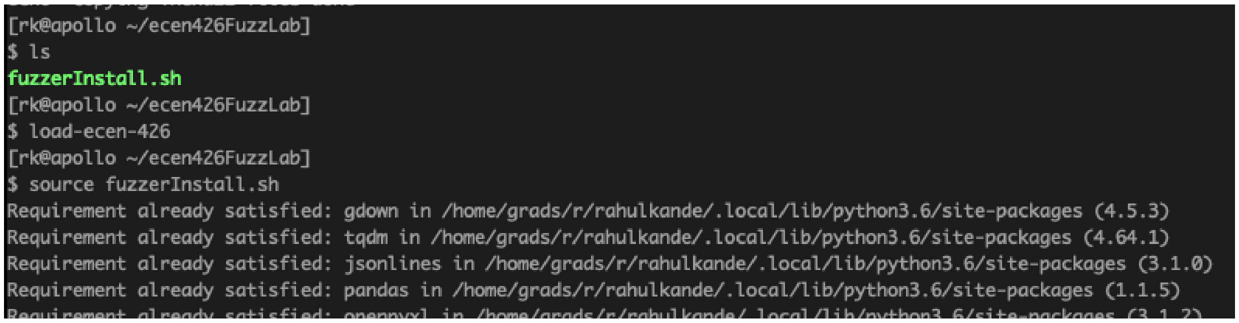
```
$ mkdir ecen426FuzzLab
[rk@apollo ~]
$ cd ecen426FuzzLab/
[rk@apollo ~/ecen426FuzzLab]
$ cp /mnt/lab_files/ECEN426/assignments/fuzzing_lab/fuzzerInstall.sh .
[rk@apollo ~/ecen426FuzzLab]
$ ls
fuzzerInstall.sh
[rk@apollo ~/ecen426FuzzLab]
$ 
```

4. Open the "fuzzerInstall.sh" file and edit the path in the first line to match the folder you created in step 2.



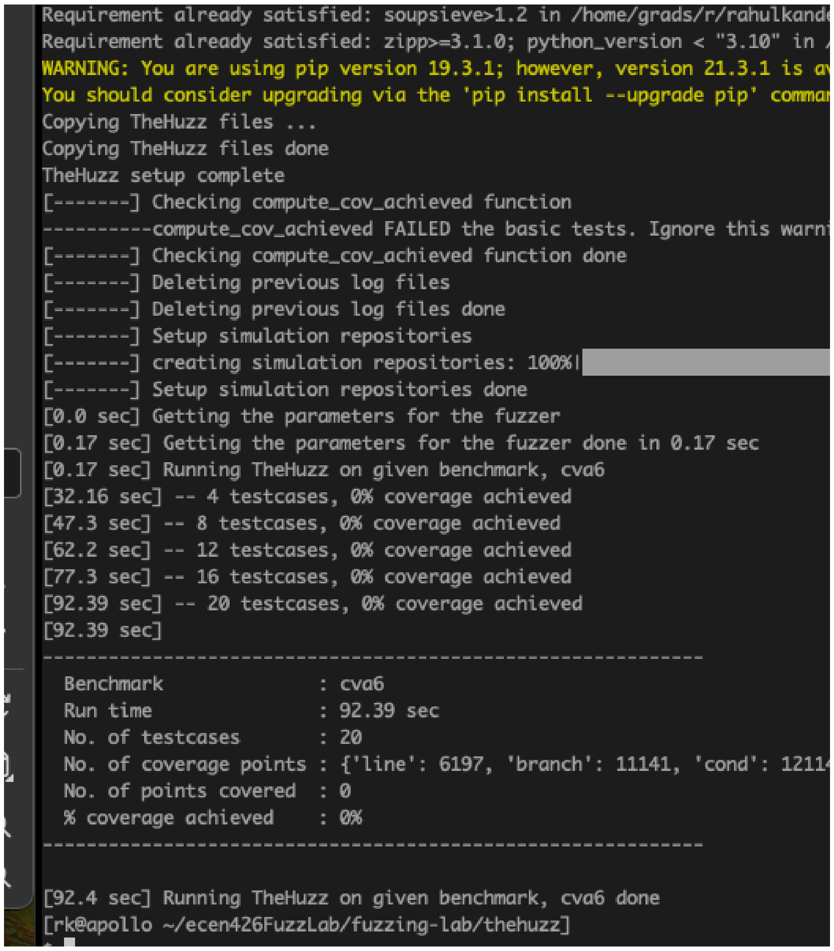
```
[rk@apollo ~/ecen426FuzzLab]
$ pwd
/home/grads/r/rahulkande/ecen426FuzzLab
[rk@apollo ~/ecen426FuzzLab]
$ head fuzzerInstall.sh
export PARENT_PATH=/home/grads/r/rahulkande/ecen426FuzzLab
export LABROOT=/mnt/lab_files/ECEN426/assignments/fuzzing_lab
PARENT_PATH_VIM=${PARENT_PATH//\//\\/}

# python dependencies
pip3 install --user gdown tqdm jsonlines pandas openpyxl
```

5. Install the lab material (**You only need to do this once for the assignment**):
   a. `load-ecen-426`
   b. `source fuzzerInstall.sh`



```
[rk@apollo ~/ecen426FuzzLab]
$ ls
fuzzerInstall.sh
[rk@apollo ~/ecen426FuzzLab]
$ load-ecen-426
[rk@apollo ~/ecen426FuzzLab]
$ source fuzzerInstall.sh
Requirement already satisfied: gdown in /home/grads/r/rahulkande/.local/lib/python3.6/site-packages (4.5.3)
Requirement already satisfied: tqdm in /home/grads/r/rahulkande/.local/lib/python3.6/site-packages (4.64.1)
Requirement already satisfied: jsonlines in /home/grads/r/rahulkande/.local/lib/python3.6/site-packages (3.1.0)
Requirement already satisfied: pandas in /home/grads/r/rahulkande/.local/lib/python3.6/site-packages (1.1.5)
Requirement already satisfied: openpyxl in /home/grads/r/rahulkande/.local/lib/python3.6/site-packages (3.1.2)
```

   c. You should see an output like this once at the end of installation.

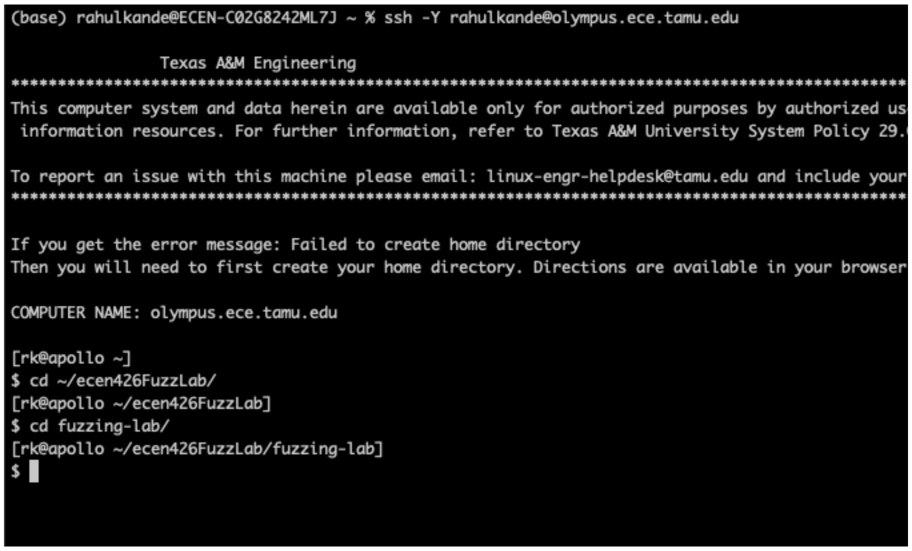


```
Requirement already satisfied: soupsieve>1.2 in /home/grads/r/rahulkande
Requirement already satisfied: zipp>=3.1.0; python_version < "3.10" in /
WARNING: You are using pip version 19.3.1; however, version 21.3.1 is av
You should consider upgrading via the 'pip install --upgrade pip' comman
Copying TheHuzz files ...
Copying TheHuzz files done
TheHuzz setup complete
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved FAILED the basic tests. Ignore this warni
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 100%|
[-------] Setup simulation repositories done
[0.0 sec] Getting the parameters for the fuzzer
[0.17 sec] Getting the parameters for the fuzzer done in 0.17 sec
[0.17 sec] Running TheHuzz on given benchmark, cva6
[32.16 sec] -- 4 testcases, 0% coverage achieved
[47.3 sec] -- 8 testcases, 0% coverage achieved
[62.2 sec] -- 12 testcases, 0% coverage achieved
[77.3 sec] -- 16 testcases, 0% coverage achieved
[92.39 sec] -- 20 testcases, 0% coverage achieved
[92.39 sec]
------------------------------------------------------------
  Benchmark                : cva6
  Run time                 : 92.39 sec
  No. of testcases         : 20
  No. of coverage points : {'line': 6197, 'branch': 11141, 'cond': 1211
  No. of points covered   : 0
  % coverage achieved     : 0%
------------------------------------------------------------

[92.4 sec] Running TheHuzz on given benchmark, cva6 done
[rk@apollo ~/ecen426FuzzLab/fuzzing-lab/thehuzz]
```

d. Note that the other setup files and tools required for this assignment are already pre-loaded on the TAMU servers and will be automatically used by the TheHuzz source code. So, you don't need to do anything else for setting up TheHuzz.

6. Running the TheHuzz fuzzer **(You need to do this everytime you want to run the fuzzer):**
   a. Go to the git repository you created for this assignment and then enter the "fuzzing-lab" folder.
      i. `cd <path to fuzzing-lab folder>`

ii. `cd fuzzing-lab`

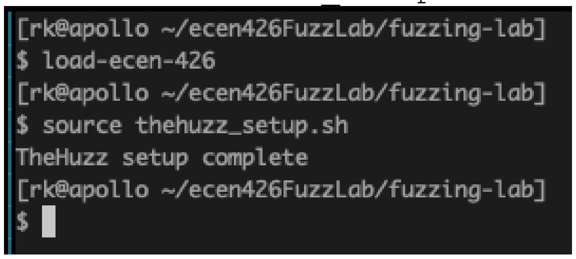


```
(base) rahulkande@ECEN-C02G8242ML7J ~ % ssh -Y rahulkande@olympus.ece.tamu.edu

               Texas A&M Engineering
*********************************************************************************************
This computer system and data herein are available only for authorized purposes by authorized us
 information resources. For further information, refer to Texas A&M University System Policy 29.

To report an issue with this machine please email: linux-engr-helpdesk@tamu.edu and include your
*********************************************************************************************

If you get the error message: Failed to create home directory
Then you will need to first create your home directory. Directions are available in your browser

COMPUTER NAME: olympus.ece.tamu.edu

[rk@apollo ~]
$ cd ~/ecen426FuzzLab/
[rk@apollo ~/ecen426FuzzLab]
$ cd fuzzing-lab/
[rk@apollo ~/ecen426FuzzLab/fuzzing-lab]
$
```

b. Setup the environment (do this once every time you open a new terminal window or connect to the Olympus server):

i. `load-ecen-426`

ii. `source thehuzz_setup.sh`



```
[rk@apollo ~/ecen426FuzzLab/fuzzing-lab]
$ load-ecen-426
[rk@apollo ~/ecen426FuzzLab/fuzzing-lab]
$ source thehuzz_setup.sh
TheHuzz setup complete
[rk@apollo ~/ecen426FuzzLab/fuzzing-lab]
$
```

c. Run the fuzzer:

i. `cd thehuzz`

ii. `python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 512`

iii. Use `python3 fuzz.py --help` to learn about the details of the arguments (this step is optional, you don't actually need to understand them).

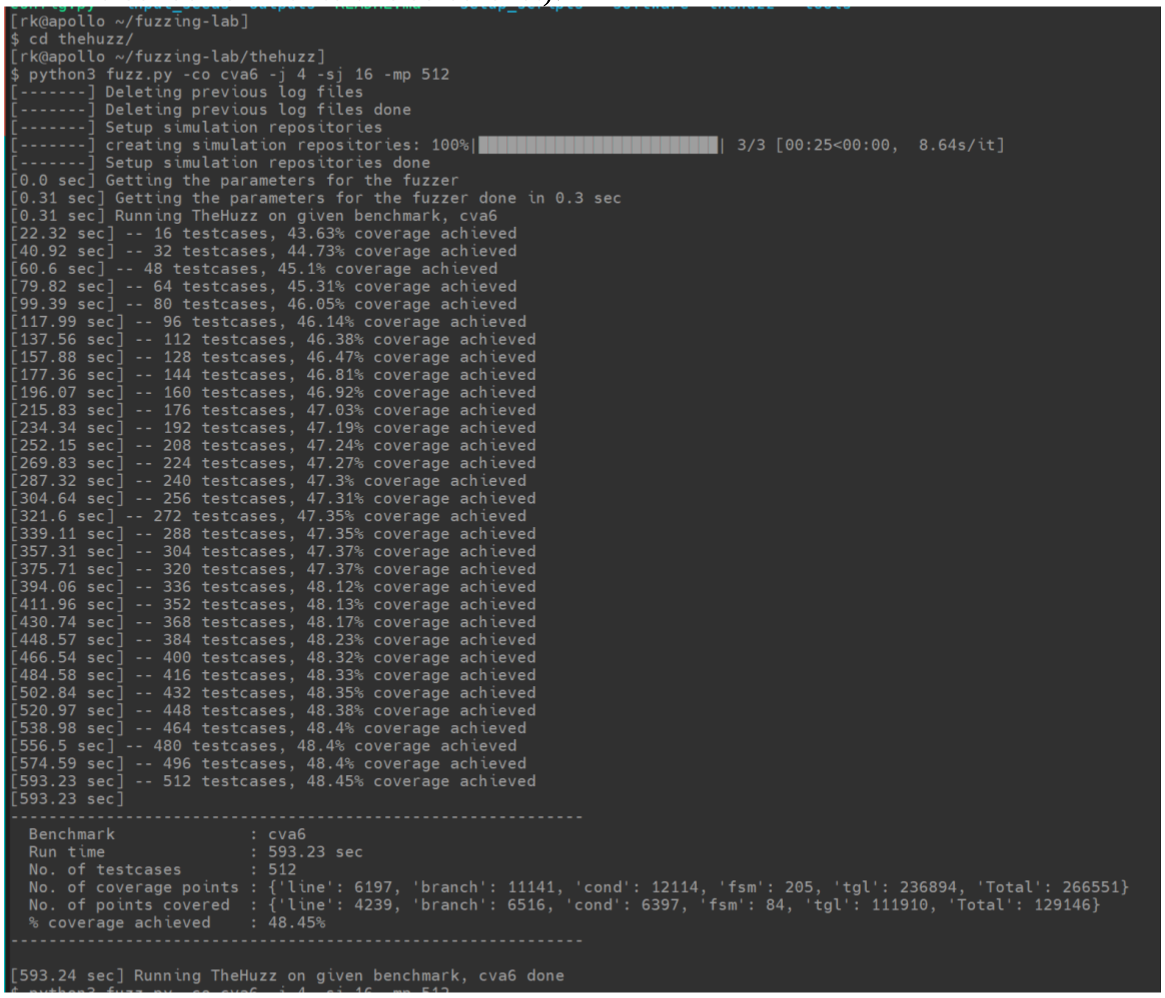
iv. **Note:** When running TheHuzz fuzzer to get the results for the assignment, you use the same arguments as present in the command above. Don't change them.

d. **Output of TheHuzz fuzzer:**

i. The fuzzer generates a lot of data as output once you run it.

ii. For this assignment, only one output file needs to be included in your report which is the output that gets printed on your terminal. You can find the same output at `fuzzing-lab/outputs/0_cva6_thehuzz_<time when you ran fuzzer>/fuzz_log.txt` file but this file has some other information also in it.

iii. Here is how the output looks on the terminal once you run TheHuzz (note that your coverage numbers can be lower than what you see in this screenshot. Once you fix all the four functions, your coverage numbers will also look similar to this screenshot):



```
[rk@apollo ~/fuzzing-lab]
$ cd thehuzz/
[rk@apollo ~/fuzzing-lab/thehuzz]
$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 512
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 100%|█████████████████████████| 3/3 [00:25<00:00,  8.64s/it]
[-------] Setup simulation repositories done
[0.0 sec] Getting the parameters for the fuzzer
[0.31 sec] Getting the parameters for the fuzzer done in 0.3 sec
[0.31 sec] Running TheHuzz on given benchmark, cva6
[22.32 sec] -- 16 testcases, 43.63% coverage achieved
[40.92 sec] -- 32 testcases, 44.73% coverage achieved
[60.6 sec] -- 48 testcases, 45.1% coverage achieved
[79.82 sec] -- 64 testcases, 45.31% coverage achieved
[99.39 sec] -- 80 testcases, 46.05% coverage achieved
[117.99 sec] -- 96 testcases, 46.14% coverage achieved
[137.56 sec] -- 112 testcases, 46.38% coverage achieved
[157.88 sec] -- 128 testcases, 46.47% coverage achieved
[177.36 sec] -- 144 testcases, 46.81% coverage achieved
[196.07 sec] -- 160 testcases, 46.92% coverage achieved
[215.83 sec] -- 176 testcases, 47.03% coverage achieved
[234.34 sec] -- 192 testcases, 47.19% coverage achieved
[252.15 sec] -- 208 testcases, 47.24% coverage achieved
[269.83 sec] -- 224 testcases, 47.27% coverage achieved
[287.32 sec] -- 240 testcases, 47.3% coverage achieved
[304.64 sec] -- 256 testcases, 47.31% coverage achieved
[321.6 sec] -- 272 testcases, 47.35% coverage achieved
[339.11 sec] -- 288 testcases, 47.35% coverage achieved
[357.31 sec] -- 304 testcases, 47.37% coverage achieved
[375.71 sec] -- 320 testcases, 47.37% coverage achieved
[394.06 sec] -- 336 testcases, 48.12% coverage achieved
[411.96 sec] -- 352 testcases, 48.13% coverage achieved
[430.74 sec] -- 368 testcases, 48.17% coverage achieved
[448.57 sec] -- 384 testcases, 48.23% coverage achieved
[466.54 sec] -- 400 testcases, 48.32% coverage achieved
[484.58 sec] -- 416 testcases, 48.33% coverage achieved
[502.84 sec] -- 432 testcases, 48.35% coverage achieved
[520.97 sec] -- 448 testcases, 48.38% coverage achieved
[538.98 sec] -- 464 testcases, 48.4% coverage achieved
[556.5 sec] -- 480 testcases, 48.4% coverage achieved
[574.59 sec] -- 496 testcases, 48.4% coverage achieved
[593.23 sec] -- 512 testcases, 48.45% coverage achieved
[593.23 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 593.23 sec
  No. of testcases       : 512
  No. of coverage points : {'line': 6197, 'branch': 11141, 'cond': 12114, 'fsm': 205, 'tgl': 236894, 'Total': 266551}
  No. of points covered  : {'line': 4239, 'branch': 6516, 'cond': 6397, 'fsm': 84, 'tgl': 111910, 'Total': 129146}
  % coverage achieved    : 48.45%
------------------------------------------------------------

[593.24 sec] Running TheHuzz on given benchmark, cva6 done
```