Project Final Report
ECE1724
Amarpreet Singh, 1002513764

Introduction

The fuzzer is a program that tests the input program with invalid input, unexpected input, and random input to find the software vulnerabilities in the input program. The purpose of the project is to study the performance of the American Fuzzy Lop fuzzer and obtain the coverage of the input program when the American Fuzzy Lop fuzzer completes the fuzzing of the input program with the inputs that are placed on multiple machines [1]. The American Fuzzy Lop fuzzer is a single machine program and multiple independent copies would be run on different machines. Each machine would run the American Fuzzy Lop fuzzer with a subset of the total inputs of the input program of the American Fuzzy Lop fuzzer.

The technical challenge of the project could be to implement a system with a simple Application Programming Interface that enables users to complete the fuzzing of the input program with multi-system parallelization of the American Fuzzy Lop fuzzer when the default implementation of the American Fuzzy Lop fuzzer runs with single-system parallelization.

The problem that is associated with the project is that the American Fuzzy Lop fuzzer program could be Central Processing Unit intensive and the action to complete the fuzzing of the input program on one machine could be inefficient when the speed of the American Fuzzy Lop fuzzer to complete fuzzing the input program is slow. The performance of the American Fuzzy Lop fuzzer could be improved with a system that utilizes multiple machines to complete the fuzzing and to obtain the results of the fuzzing of the input program [2]. The action to parallelize the American Fuzzy Lop fuzzer could be interesting to understand the improvement of the American Fuzzy Lop fuzzer when the number of machines in the system increases to improve the performance of the American Fuzzy Lop fuzzer.

The study of the performance of the American Fuzzy Lop fuzzer would be to evaluate the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes. The total paths statistic indicates the number of program paths that were found during the fuzzing of the input program, and the total crashes and unique crashes statistics indicate the number of program paths that resulted in the program to complete unsuccessfully.

The research question that is associated with the project could be: what is the performance improvement of the American Fuzzy Lop fuzzer when the number of machines in the multi-system parallelization of the American Fuzzy Lop fuzzer increases? The performance improvement of the American Fuzzy Lop fuzzer would be obtained after evaluating the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes.

The expected result of the project is that the performance of the American Fuzzy Lop fuzzer would improve when the inputs of the American Fuzzy Lop fuzzer input program are placed on multiple machines to complete the fuzzing of the input program. The results of the American Fuzzy Lop fuzzer after fuzzing the input program in every machine would be would be obtained after logging the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes during the runtime of the American Fuzzy Lop fuzzer and when the attributes of the American Fuzzy Lop fuzzer coverage is modified during the runtime of the American Fuzzy Lop fuzzer.

Background

The related work of the project includes the paper named MapReduce: Simplified Data Processing on Large Clusters that is associated with the idea of placing the inputs of the American Fuzzy Lop fuzzer input program on multiple machines, and the multi-system parallelization implementations of the American Fuzzy Lop fuzzer named AFLplusplus, roving and disfuzz-afl [2, 4, 5, 6].

The AFLplusplus fuzzer runs multiple instances of the American Fuzzy Lop fuzzer on multiple machines [4]. The roving fuzzer runs multiple copies of AFL on multiple machines, all fuzzing the same input program. The contribution of the roving fuzzer is to enable multiple machines to share and benefit from each other's work. If machine A finds a test case that increases the coverage of the input program, then machines B, C and D all utilize the test case to continue fuzzing the input program and obtain more coverage of the test program.

The disfuzz-afl fuzzer includes a server that provides

the input program to the fuzzer and the client that runs on the machines that downloads the input program depending on the amount of CPU cores, downloads the queue and starts fuzzing. The generated test cases are uploaded to the queue to sync to other clients and the information about the crashes are uploaded to be manually studied. The client periodically checks if the input program is modified, and updates and restarts the fuzzer when an update to the input program is detected.

Implementation status

The project includes running the American Fuzzy Lop fuzzer on one Google Cloud Platform machine with the GNU Coreutils input programs named base64, md5sum, uniq and who, and running the American Fuzzy Lop fuzzer with the GNU Coreutils input programs named base64, md5sum, uniq and who on multiple Google Cloud Platform machines. The GNU Coreutils program named base64 transforms data in a file into and from base64 encoded form, md5sum computes a 128-bit checksum of the specified file, uniq writes the unique lines in the input, and who prints the information about users that are at the present time utilizing the machine [3].

The design of the system includes multiple Google Cloud Platform machines that are the worker machines with one Google Cloud Platform machine that is the primary machine that orchestrates the operations of the American Fuzzy Lop fuzzer on the worker Google Cloud Platform machines. The primary Google Cloud Platform machine would provide the results of the American Fuzzy Lop fuzzer to another machine that would read the results of the American Fuzzy Lop fuzzer that was obtained from the worker machine and indicate the combined results of the American Fuzzy Lop fuzzer.

The results of the American Fuzzy Lop fuzzer after fuzzing the input program in every machine is obtained after logging the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes during the runtime of the American Fuzzy Lop fuzzer and when the attributes of the American Fuzzy Lop fuzzer coverage is modified during the runtime of the American Fuzzy Lop fuzzer.

The machine that reads the results of the American Fuzzy Lop fuzzer to indicate the combined results of the American Fuzzy Lop fuzzer combines the results by adding the count of the attributes of the American Fuzzy Lop coverage at the corresponding timestamps. The current implementation does not filter out the duplicates in the total paths attribute of the American Fuzzy Lop coverage.

The method to filter out the duplicates in the total paths attribute of the American Fuzzy Lop coverage could be to obtain an encoding of the control flow graph path that is associated with every instance of the total path attribute of the American Fuzzy Lop coverage from the bitmap of the American Fuzzy Lop fuzzer, and to add the encoding of the control flow graph path that is associated with every instance of the total path attribute of the American Fuzzy Lop coverage to the log file. The machine that reads the results of the American Fuzzy Lop fuzzer to indicate the combined results of the American Fuzzy Lop fuzzer would not count the total paths attribute with the same control flow graph path.

Experimental methodology

The first step of the experimental methodology of the project is to run the American Fuzzy Lop fuzzer on one Google Cloud Platform machine.

Table 1: The configuration of the primary and worker Google Cloud Platform machines [7].

The name of the attribute in the configuration of the machine	The value of the attribute in the configuration of the machine
Region	Northamerica-northeast 2, Toronto
Zone	northamerica-northeast 2-a
Machine type	e2-micro
vCPU	2
Memory	1 GB
Disk size	10 GB
Image	Debian GNU/Linux 11, bullseye
Architecture	x86/64

Table 2: The steps to start the American Fuzzy Lop fuzzer on one Google Cloud Platform machine.

Step number	Step	Definition of the step
1	sudo apt-get -y install wget	Install the program named wget on the Google Cloud Platform machine.
2	wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/worker.sh	Download the worker script that starts the American Fuzzy Lop fuzzer on the Google Cloud Platform machine.
3	sudo apt-get -y install unzip	Install the program named unzip on the Google Cloud Platform machine
4	wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/fuzzer_input.zip	Download the inputs of the American Fuzzy Lop fuzzer input program.
5	mkdir {the directory name of American Fuzzy Lop fuzzer inputs}	Create the directory that will store the inputs of the American Fuzzy Lop fuzzer.
6	unzip -qq /home/user/fuzzer_input.zip -d {the directory name of American Fuzzy Lop fuzzer inputs	Extract the inputs of the American Fuzzy Lop fuzzer in the directory.
7	sudo apt-get -y install screen	Install the program named screen on the Google Cloud Platform machine.
8	screen -S {the name of the input program}	Create the screen session with the name of the input program.
9	mkdir {the name of the input program}	Create the directory to save the results of the American Fuzzy Lop fuzzer.
10	bash /home/user/ece1776_afl_tutorial.sh --input_program={the name of the input program} --directory_path={the name of the input program} --map_size_pow2={the size of the American Fuzzy Lop fuzzer bitmap} --max_dict_file={the size of the testcase} --input_program_inputs={the directory name of American Fuzzy Lop fuzzer inputs}	Start the script that starts the American Fuzzy Lop fuzzer and provide the name of the input program, the path of the directory that would include the results of the American Fuzzy Lop fuzzer, the size of the bitmap, the size of the test case of the American Fuzzy Lop fuzzer, and the the directory name with the American Fuzzy Lop fuzzer inputs.

The second step of the experimental methodology of the project is to start and bootstrap the primary Google Cloud Platform machine.

Table 3: The steps to bootstrap the primary Google Cloud Platform machine.

Step number	Step	Definition of the step
1	https://cloud.google.com/compute/docs/connect/add-ssh-keys#console_2	Add the Secure Shell Protocol public key of the primary and worker Google Cloud Platform machines to the Google Cloud Platform project.
2	scp -i {Secure Shell Protocol private key} {Secure Shell Protocol private key} user@{primary Google Cloud Platform machine external IP address}:/home/user/.ssh	Place the Secure Shell Protocol private key of the Google Cloud Platform project on the primary Google Cloud Platform machine.
3	sudo apt-get -y install wget	Install the program named wget on the primary Google Cloud Platform machine.
4	wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/primary.sh	Download the script that starts the American Fuzzy Lop fuzzer on the worker Google Cloud Platform machine.

The third step of the experimental methodology of the project is to start the worker Google Cloud Platform machines from the Google Cloud Platform primary machine.

Table 4: The steps to start the worker Google Cloud Platform machines from the Google Cloud Platform primary machine

Step number	Step	Definition of the step
1	bash /home/user/primary.sh --directory_path={directory path} --project_id={Google Cloud Platform project id} --number_of_workers={number of worker machines} --fuzzer_input_program={input program of the American Fuzzy Lop fuzzer}	Start the primary script that starts the American Fuzzy Lop fuzzer on the worker Google Cloud Platform machines and provide the path of the directory that includes the primary script, the Google Cloud Platform project id, the number of worker Google Cloud Platform machines to start, and the input program of the American Fuzzy Lop fuzzer.

The fourth step of the experimental methodology of the project will be to obtain the results of the American Fuzzy Lop fuzzer with the GNU Coreutils programs named base64, md5sum, uniq and who; and with 1, 3, and 5 worker machines.

The fifth step of the experimental methodology of the project will be to read the results of the American Fuzzy Lop fuzzer that was obtained from the worker Google Cloud Platform machine and indicate the combined results of the American Fuzzy Lop fuzzer.

Table 5: The steps to indicate the combined results of the American Fuzzy Lop fuzzer.

Step number	Step	Definition of the step
1	scp -r -i {Secure Shell Protocol private key} {Secure Shell Protocol private key} user@{primary Google Cloud Platform machine external IP address}: {directory path with the results of the American Fuzzy Lop fuzzer} {data directory path}	Obtain the results of the American Fuzzy Lop Fuzzer from the worker Google Cloud Platform machines.
2	wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/results/Machine.py wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/results/Utilities.py wget https://raw.githubusercontent.com/singh264/ece1724/project_final_report/results/Visualizer.py	Downloads the files to indicate the combined results of the American Fuzzy Lop fuzzer.
3	python3 Visualizer.py	Start the script to indicate the combined results of the American Fuzzy Lop fuzzer and provide the data directory path.

The primary Google Cloud Platform machine starts the American Fuzzy Lop fuzzer on the worker Google Cloud Platform machines with the primary script. The primary script downloads the worker script that is utilized to start the American Fuzzy Lop fuzzer on the worker Google Cloud Platform machines. The primary script downloads the inputs of the American Fuzzy Lop fuzzer input program. The primary script divides the inputs of the American Fuzzy Lop input program based on the number of worker machines and places the inputs of the American Fuzzy Lop input program into subfolders. The primary script authenticates into the Google Cloud Platform to utilize the gcloud Command Line Interface [8]. The primary script starts the worker Google Cloud Platform machines with the configuration that is detailed in Table 1. The primary script bootstraps the worker Google Cloud Platform machines with the following steps. The primary script places the inputs of the American Fuzzy Lop fuzzer input program on the worker machine. The primary script places the worker script that starts the American Fuzzy Lop fuzzer input program on the worker machine. The primary script bootstraps the worker Google Cloud Platform machine before starting the worker script on the worker Google Cloud Platform machine.

The worker Google Cloud Platform machine runs the American Fuzzy Lop fuzzer with the worker script. The worker script builds the American Fuzzy Lop fuzzer. The worker script obtains the test cases of the input program of the American Fuzzy Lop fuzzer. The worker script installs the dependencies to create

the American Fuzzy Lop coverage of the input program. The worker script creates the American Fuzzy Lop coverage of the input program. The worker script generates the output of the American Fuzzy Lop coverage that is run with the input program.

Table 6: The steps of the primary script to bootstrap the worker Google Cloud Platform machines.

Step number	Step	Definition of the step
1	scp -r -i {Secure Shell Protocol private key} {directory with the inputs of the American Fuzzy Lop fuzzer input program} user@{worker Google Cloud Platform machine external IP address}:/home/user	Place the inputs of the American Fuzzy Lop fuzzer input program on the worker Google Cloud Platform machine.
2	scp -i {Secure Shell Protocol private key} {path to the worker script} user@{worker Google Cloud Platform machine external IP address}:/home/user	Place the worker script on the worker Google Cloud Platform machine.
3	ssh -i {Secure Shell Protocol private key} user@{worker Google Cloud Platform machine external IP address} "sudo apt-get -y install screen"	Install the program named screen on the worker Google Cloud Platform machine.
4	ssh -i {Secure Shell Protocol private key} user@{worker Google Cloud Platform machine external IP address} screen -d -m "mkdir {directory path of the American Fuzzy Lop fuzzer}"	Create the directory on the worker Google Cloud Platform machine that will store the results of the American Fuzzy Lop fuzzer on the worker machine.
5	ssh -i {Secure Shell Protocol private key} user@{worker Google Cloud Platform machine external IP address} screen -d -m "bash worker.sh --input_program={input program of the American Fuzzy Lop fuzzer} --directory_path={directory path of the American Fuzzy Lop fuzzer} --map_size_pow2=16 --max_dict_file=256 --input_program_inputs={directory with the inputs of the American Fuzzy Lop	Start the worker script on the worker Google Cloud Platform machine.

	fuzzer input program}	

Results

The result of the project is that the performance of the American Fuzzy Lop fuzzer improved when the inputs of the American Fuzzy Lop fuzzer input program were placed on multiple machines to complete the fuzzing of the input program. The results of the American Fuzzy Lop fuzzer after fuzzing the input program in every machine were obtained after logging the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes during the runtime of the American Fuzzy Lop fuzzer and when the attributes of the American Fuzzy Lop fuzzer coverage is modified during the runtime of the American Fuzzy Lop fuzzer.

The diagrams below indicate the performance of the American Fuzzy Lop fuzzer to find the total paths in the input program throughout the duration of multiple hours with the input programs named base64, md5sum, uniq and who; where the total paths statistic indicates the number of program paths that were found during the fuzzing of the input program. The results indicate that the American Fuzzy Lop fuzzer found more total paths in the input program when the inputs of the American Fuzzy Lop input program were placed on multiple machines.

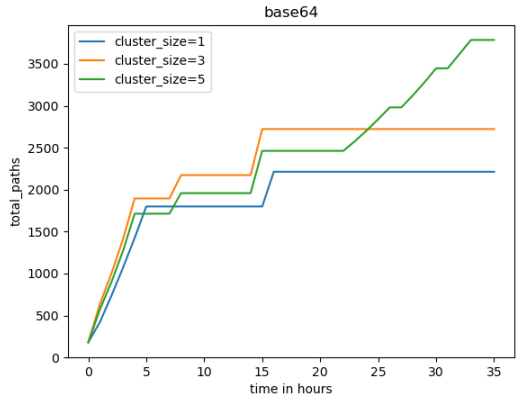

Diagram 1: Total paths of base64.

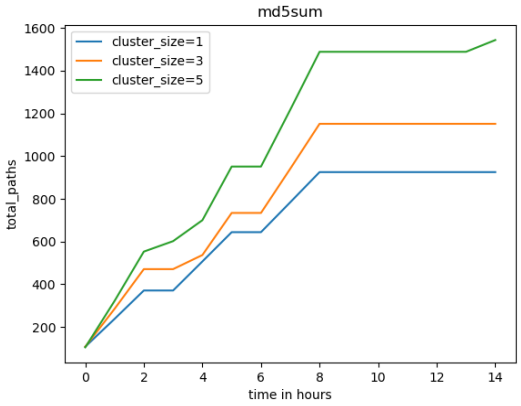

Diagram 2: Total paths of md5sum.

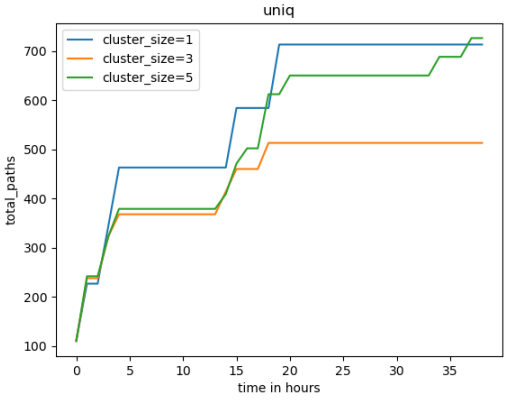

Diagram 3: Total paths of uniq.

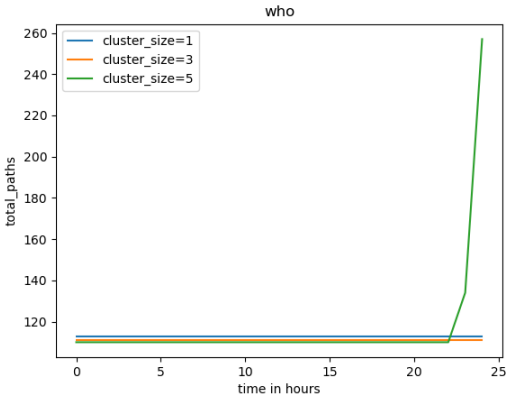

Diagram 4: Total paths of who.

The diagrams below indicate the performance of the American Fuzzy Lop fuzzer to find the total crashes and unique crashes in the input program throughout the duration of multiple hours with the input programs named base64, md5sum, uniq and who; where the total crashes and unique crashes statistic indicates the number of program paths that resulted in the program to complete unsuccessfully. The results

indicate that the American Fuzzy Lop fuzzer found more total paths in the input program when the inputs of the American Fuzzy Lop input program were placed on multiple machines.

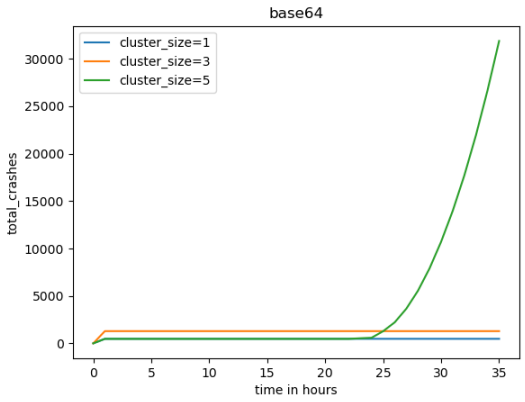

Diagram 5: Total crashes of base64.

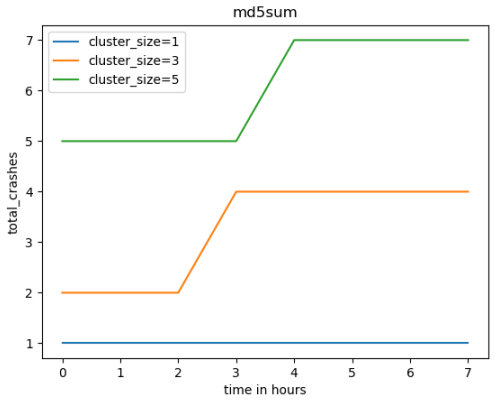

Diagram 6: Total crashes of md5sum.

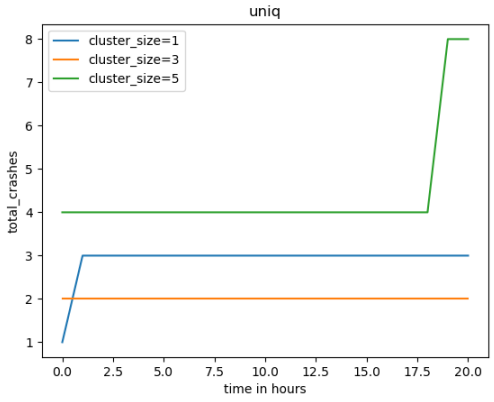

Diagram 7: Total crashes of uniq.

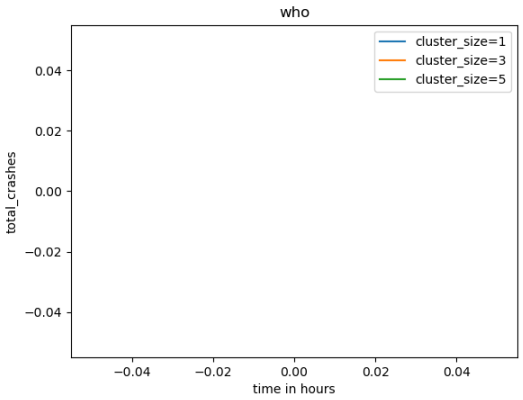

Diagram 8: Total crashes of who.

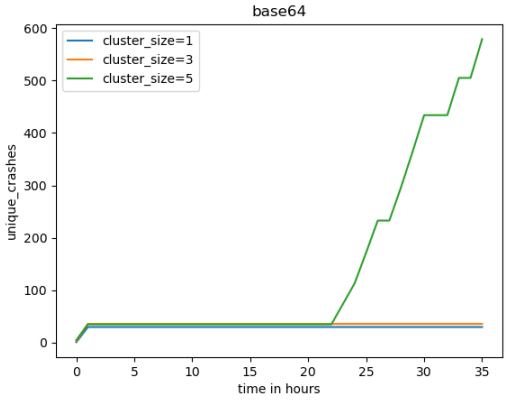

Diagram 9: Unique crashes of base64.

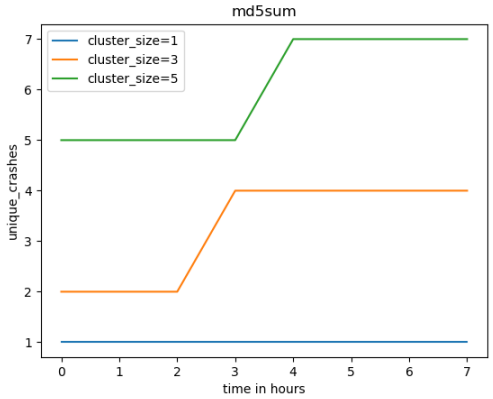

Diagram 10: Unique crashes of md5sum.

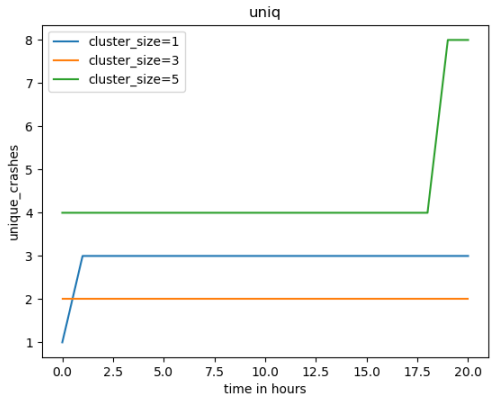

Diagram 11: Unique crashes of uniq.

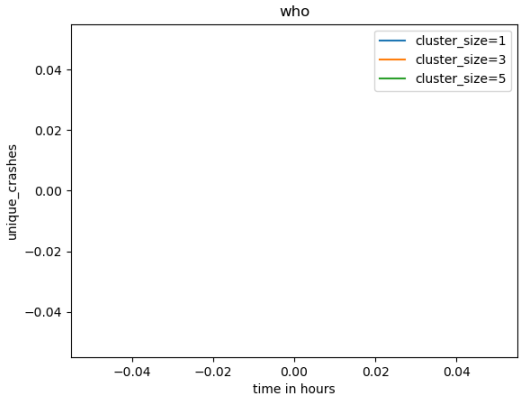

Diagram 12: Unique crashes of who.

Conclusion

The actual result of the project corresponds with the expected result of the project, where the expected result of the projected indicated that the performance of the American Fuzzy Lop fuzzer would improve when the inputs of the American Fuzzy Lop fuzzer input program are placed on multiple machines to complete the fuzzing of the input program. The results of the American Fuzzy Lop fuzzer after fuzzing the input program in every machine were obtained after logging the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes during the runtime of the American Fuzzy Lop fuzzer and when the attributes of the American Fuzzy Lop fuzzer coverage was modified during the runtime of the American Fuzzy Lop fuzzer.

Future work

The first next step of the project could be to obtain the results of the American Fuzzy Lop fuzzer after running multiple instances of the fuzzer on one machine. The efficiency of running the American Fuzzy Lop fuzzer on multiple machines could be determined by dividing the results of the American Fuzzy Lop fuzzer on multiple machines with the results of the American Fuzzy Lop fuzzer with multiple instances on one machine.

The second next step of the project could be to filter out the duplicates in the total paths attribute of the American Fuzzy Lop coverage. The method to filter out the duplicates in the total paths attribute of the American Fuzzy Lop coverage could be to obtain an encoding of the control flow graph path that is associated with every instance of the total path attribute of the American Fuzzy Lop coverage from the bitmap of the American Fuzzy Lop fuzzer, and to add the encoding of the control flow graph path that is associated with every instance of the total path attribute of the American Fuzzy Lop coverage to the log file. The machine that reads the results of the American Fuzzy Lop fuzzer to indicate the combined results of the American Fuzzy Lop fuzzer would not count the total paths attribute with the same control flow graph path.

The third next step of the project could be to adjust the primary Google Cloud Platform to combine the results of the American Fuzzy Lop fuzzer that are on multiple machines to run the program named afl-cov [9]. The input of the program named afl-cov is the output of the American Fuzzy Lop fuzzer, and the output of the program named afl-cov provides details about the coverage of the American Fuzzy Lop coverage of the input program that includes the percentage of the lines, the functions, and the branches that were covered during the fuzzing of the input program.

Code

The code of the project is located at https://github.com/singh264/ece1724/tree/project_final_report. The GitHub repository includes the primary script, the worker script, the data of the experiments of the project, the code to obtain the results of the project, and the results of the project.

The adjustment to the American Fuzzy Lop fuzzer is located at https://github.com/singh264/AFL/tree/ece1724_project_final_report. The GitHub repository includes the

American Fuzzy Lop fuzzer with the adjustment of logging the attributes of the American Fuzzy Lop coverage that includes the total paths, the total crashes, and the unique crashes during the runtime of the American Fuzzy Lop fuzzer and when the attributes of the American Fuzzy Lop fuzzer coverage is modified during the runtime of the American Fuzzy Lop fuzzer.

References

[1] “american fuzzy lop,” GitHub, Nov. 30, 2021. https://github.com/google/AFL (accessed Feb. 10, 2023).

[2] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107–113. https://doi.org/10.1145/1327452.1327492 (accessed Feb. 10, 2023).

[3] “GNU Coreutils 9.1,” www.gnu.org. https://www.gnu.org/software/coreutils/manual/coreutils.html#who-invocation (accessed Feb. 10, 2023).

[4] “Parallel Fuzzing,” AFLplusplus. https://aflplus.plus/docs/parallel_fuzzing/ (accessed Mar. 15, 2023).

[5] richö butts, “Roving,” GitHub, Dec. 06, 2022. https://github.com/richo/roving (accessed Mar. 15, 2023).

[6] M. Bogaard, “Distributed Fuzzing for afl,” GitHub, Mar. 13, 2023. https://github.com/MartijnB/disfuzz-afl (accessed Mar. 15, 2023).

[7] “Free trial and free tier | google cloud,” Google. [Online]. Available: https://cloud.google.com/free/ (accessed Mar. 15, 2023).

[8] “gcloud CLI overview | Google Cloud CLI Documentation,” Google Cloud. https://cloud.google.com/sdk/gcloud (accessed Mar. 15, 2023).

[9] Mrash, “Mrash/AFL-COV: Produce code coverage results with gcov from AFL-fuzz test cases,” GitHub. [Online]. Available: https://github.com/mrash/afl-cov. (accessed: APr. 9, 2023).