# Fuzzer Research

Jack Foley

October 7, 2024

# Contents



# List of Figures

# 1 Introduction

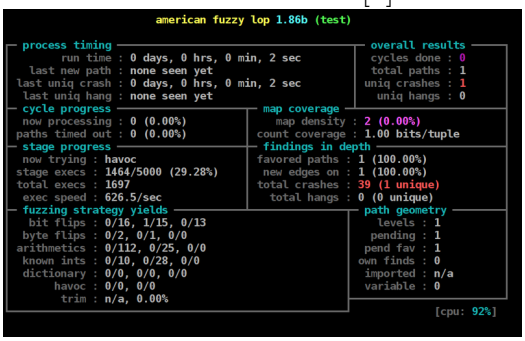
This document will outline the research undertaken for the 4th Software Development Final Year Project (FYP). This project was proposed by Dr. Chris Meudec and focuses developing a fuzzer for the C programming language.

# 2 Fuzzing

Fuzzing is a method of testing software by using broken, random or unusual data as an input into the software which is being tested. The idea of fuzzing is that it will find bugs and other issues, including memory spikes and leaks (temporary denial-of-service), buffer overruns (remote code execution), unhandled exceptions, read access violations (AVs), and thread hangs (permanent denial-of-service). These are issues that traditional software testing methods, such as unit testing, will not find as easily. There are some different types of fuzzing, such as White-Box, grey-box and black-box fuzzing [3].

During typical use of a fuzzer, the fuzzer will use structured data as its input. The structure that the fuzzer will use is normally already defined via "a file format or protocol" [5] which in turn can be used to determine a correct input from an incorrect input. The fuzzer will create cases that are considered "correct enough" so that the software will not reject it, but also "incorrect enough" so that "corner cases" [5] can be found.

Figure 1: American Fuzzy Lop screenshot. Version used: 1.86b [5]




## 2.1 White-Box Fuzzing

White-Box fuzzing, also known as smart fuzzing, is a technique that is used where the fuzzer is fully aware of the code structure and input variables. White-Box fuzzing often leads to discovering bugs more quickly compared to grey-box and black-box fuzzing, but it can also be more computationally expensive as it needs to do an analysis of the codebase before running.

A case study done during development of ISA Server 2006 showed that one defect was found per 17 KLOC (thousand lines of code), a similar black-box fuzzer only found 30% of the defects that the White-Box fuzzer found

[3].

### 2.2 Black-Box Fuzzing

Black-Box fuzzing is a technique used to test software, analyzing the software by sending random data to the software to discover an application's bugs and vulnerabilities. The black-box fuzzer does not have any information about the inner-workings of the software, it only knows the input and output of the software.

It is a sought-after testing technique as it will work it applications regardless of the programming language or the platform that the software is running on [1].

### 2.3 Grey-Box Fuzzing

Grey-Box fuzzing is a well-known and commonly used fuzzing technique that is used for testing software and finding vulnerabilities. Differing from White-Box fuzzing, which can suffer from high computational needs since source code analysis is required, grey-box fuzzing is a very good middle-ground between White-Box and black-box fuzzing.[4]

Grey-Box fuzzing can also receive coverage feedback from the software, which can then be used to more efficiently traverse the software's codebase to find bugs and vulnerabilities [2].

## 3 Techniques Deep Dive

### 3.1 Random Input Fuzzing

The simplest implementation of a Fuzzer is a Random Fuzzer. This type of Fuzzer will generate a random string at a fixed or variable length which will then be used as the input for the software which we are testing. This method works well at producing errors, but may struggle at producing inputs that **do not** cause errors [6].

Examples of random fuzzing would be: `*&322h2k,b&(Gb2\|q&@ih`

### 3.2 Mutation Based Fuzzing

Instead of generating completely random strings, we can use mutation based fuzzing. Most randomly generated inputs are always invalid, which is not ideal. Mutation Fuzzing will take a valid input at first, then with each

subsequent execution, it will change, or mutate, the string slightly. This mutation is usually done by modifying one random character in the input. This approach is popular with fuzzing as it may cause the program to crash while only changing the input slightly, which is difficult to achieve with traditional testing [7].

Examples of mutation fuzzing would be:

- **Original Input:** `Hello World`
- **Mutated Input 1:** `Hello Wzrld`
- **Mutated Input 2:** `Hell1 Wzrld`
- **Mutated Input 3:** `H;ll1 Wzrld`
- ...
- **Mutated Input N:** `Fr'1?.tOP4+`

# 4 Benchmarking Overview

## 4.1 Competitions

### 4.1.1 Test-Comp

Test-Comp ... TODO

### 4.1.2 SV-Comp

SV-Comp (SVC) is a software verification benchmark website. It runs an annual competition to test various different software verification tools which can be used in the software development lifecycle. It is mostly used to prove the correctness of a software verification tool while following formal specifications, but it seems that there is some fuzzers used in the competition.

SVC does not seem like it will be a good candidate for testing the fuzzer as it is mostly used for the testing of verification tools, not fuzzers. A better alternative is Test-Comp, a software testing competition that is run by the same people who run SVC, but has a higher focus on software testing tools, including fuzzers, rather than software verification tools.

# References


[1] Aseel Alsaedi, Abeer Alhuzali, and Omaimah Bamasag. "Effective and scalable black-box fuzzing approach for modern web applications". In: *Journal of King Saud University - Computer and Information Sciences* 34.10, Part B (2022), pp. 10068–10078. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2022.10.006. URL: https://www.sciencedirect.com/science/article/pii/S1319157822003573.

[2] Daniel Blackwell and David Clark. "PrescientFuzz: A more effective exploration approach for grey-box fuzzing". In: (Apr. 2024). arXiv: 2404.18887 [cs.SE].

[3] J. Neystadt. *Automated penetration testing with white-box fuzzing*. Microsoft Learn. Available at: https://learn.microsoft.com/en-us/previous-versions/software-testing/cc162782(v=msdn.10)?redirectedfrom=MSDN (Accessed: 28 September 2024). 2009.

[4] Van-Thuan Pham et al. "Smart Greybox Fuzzing". In: *IEEE Transactions on Software Engineering* 47.9 (2021), pp. 1980–1997. DOI: 10.1109/TSE.2019.2941681.

[5] Wikipedia contributors. *Fuzzing — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-October-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Fuzzing&oldid=1249540069.

[6] Andreas Zeller et al. "Fuzzing: Breaking Things with Random Inputs". In: *The Fuzzing Book*. Retrieved 2024-06-29 17:55:20+02:00. CISPA Helmholtz Center for Information Security, 2024. URL: https://www.fuzzingbook.org/html/Fuzzer.html.

[7] Andreas Zeller et al. "Mutation-Based Fuzzing". In: *The Fuzzing Book*. Retrieved 2023-11-11 18:18:06+01:00. CISPA Helmholtz Center for Information Security, 2023. URL: https://www.fuzzingbook.org/html/MutationFuzzer.html.