Fuzzing Programs with Structure Aware Fuzzers

DARSHANA DAS K AND KRISHNAHARI P AND
LIBNA KURIAKOSE T AND PARVATHY C M

BTech in Computer Science and Engineering

Supervisor: Dr. Rahul Gopinath
Guide: Dr. Ezudheen P

A project report submitted in fulfilment of
the requirements for Mini Project, Semester 6

Department of Computer Science and Engineering
Government Engineering College Thrissur

22 March 2024

Abstract

Fuzzing is a widely used technique for finding vulnerabilities in software. In recent years, structure aware fuzzers have gained attention due to their ability to better understand the input structure of programs and generate more effective test cases. Fuzzing programs that handle structured binary inputs, like multimedia files, presents a difficult challenge due to the specific input format these programs require. While current fuzzers are versatile in their approach, they often lack effectiveness when it comes to targeting a particular format. This work aims to develop a structure aware fuzzer to improve the efficacy of fuzzing techniques.
Our tool, Binary Fuzzer takes a binary template (using Kaitai Struct) as input and utilises the fuzzing logic of libFuzzer. Kaitai Struct's capability to define complex data structures in a declarative manner makes it well-suited for use as input to our tool. This fuzzer addresses the limitations of traditional fuzzing methods by leveraging knowledge of the program's structure to guide the generation of input data. The advantage of a structure aware fuzzer lies in its ability to explore the input space more intelligently, leading to a higher probability of discovering critical software flaws.

Acknowledgements

Thank everyone.

Contents

List of Figures

CHAPTER 1

Introduction

Fuzzing has become an important technique for identifying vulnerabilities in software programs. Existing fuzzers work by generating random or semi-random inputs to test the robustness of a program. However, these traditional fuzzers often lack the ability to understand the structure of the data they are manipulating, resulting in a limited ability to uncover certain types of bugs and vulnerabilities.

This is where structure aware fuzzing comes into play. By being aware of the structure of the input data, fuzzers can more effectively generate test cases that exercise the different parts of a program, potentially exposing more bugs. One approach to structure aware fuzzing involves the use of Kaitai Struct, a declarative language used to describe various binary data formats in a structured way. Using Kaitai Struct, fuzzers can be designed to take input as KSY files, which describe the structure of the data to be fuzzed.

Our tool, BinaryFuzzer, is designed to leverage the power of structure aware fuzzing by taking input in the form of KSY files. This approach allows for more intelligent and targeted fuzzing, improving the chances of finding vulnerabilities in software programs.

Kaitai Struct files provide a clear and concise way to describe the format of binary data, making it easier to create fuzzing test cases that cover specific parts of the data structure. Additionally, Kaitai Struct comes with a range of advantages such as portability, reusability, and maintainability.

Another important component in the context of fuzzing is libFuzzer, which is a library for in-process fuzzing. LibFuzzer is often used to perform fuzz testing on individual functions or modules within a program. By integrating libFuzzer with KSY files, we can create a powerful framework for structured fuzzing that combines the ability to understand data structures with the in-depth testing capabilities of libFuzzer. This approach opens up new possibilities for identifying and addressing security vulnerabilities in software.

CHAPTER 2

Literature review

In recent years, the field of fuzzing programs with structure-aware techniques has witnessed significant advancements, aiming to enhance the efficacy of vulnerability discovery in software systems. These developments have introduced various sophisticated methodologies, each with its own technical intricacies and contributions.

FormatFuzzer, developed by Rafael Dutra, Rahul Gopinath, and Andreas Zeller from CISPA Helmholtz Center for Information Security, introduces a format-specific approach to fuzzing, utilizing binary templates to generate C++ code for parsing, mutating, and generating inputs. It effectively handles complex formats like MP4 or ZIP files and integrates with format-agnostic fuzzers like AFL for intelligent mutations and seed evolution, enhancing fuzzing outcomes.

Another significant advancement is presented in the study "Superion: Grammar-Aware Greybox Fuzzing for Structured Inputs" by Junjie Wang, Bihuan Chen, Lei Wei, and Yang Liu. Superion enhances traditional coverage-based greybox fuzzing by incorporating grammar awareness, facilitating more effective trimming and mutation of test inputs while maintaining their validity and structure. Experimental studies on XML and JavaScript engines show Superion's improved code coverage and bug-finding capabilities compared to AFL, uncovering 31 new bugs, including 21 vulnerabilities with 16 CVEs assigned.

In their paper titled "Fast Format-Aware Fuzzing for Structured Input Applications," the authors propose an efficient approach for vulnerability discovery in applications with structured input. Their method, called FFAFuzz, leverages input-to-state (I2S) dependencies and indirect dependency analysis. FFAFuzz significantly reduces time overhead, outperforming existing techniques like Redqueen and WEIZZ. It's a valuable contribution to enhancing security testing for structured input applications.

2.1 Section

More text.

2.1.1 Subsection

Even more text.

CHAPTER 3

Methods

Text.

3.1 Section

More text.

3.1.1 Subsection

Even more text.

CHAPTER 4

Results

Text.

4.1 Section

More text.

4.1.1 Subsection

Even more text.

CHAPTER 5

Conclusion

Something concluding.

5.1 Future outlook

What next years student should do.

1 Appendix A

Something