Binary Fuzzer using Kaitai Struct

February 27, 2024

About the project

This project is in collaboration with Dr. Rahul Gopinath, Lecturer at University of Sydney. His main research area lies in the junction between Software Engineering and Cybersecurity. His research focus is on using static and dynamic program analysis to improve reliability, security, and maintainability of software systems.

Overview

- Understanding the Problem
- Project Objective
- Fuzzing
- Binary File format
- Kaitai Struct for binary files
- Binary Fuzzer
- Timeline

Understanding the problem

Rising Complexity in Software

- Software applications and binary file formats are becoming increasingly complex.
- The need for effective testing methods is crucial for ensuring software reliability and security.

Challenges in fuzzing binary input programs

- Effective fuzzing of programs that process structured binary inputs, such as multimedia files, is a challenging task, since those programs expect a very specific input format.

Limitations of Existing fuzzers

- Existing fuzzers are often format-agnostic.
- This format-agnostic approach makes them less effective when testing specific binary file formats.

Project objective:

Develop a binary fuzzer to find bugs and errors in software programs that takes a Kaitai Struct file and a random stream of bytes and produce test cases.

Background

What is Fuzzing?

Software often contains bugs, and the process of identifying and fixing these issues can be labor-intensive.

Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program.

Fuzzer

A program which injects automatically random data into a program/stack and detect bugs.

This is the python code for a simple fuzz generator.

```
import random
def fuzzer(max_length: int = 100, char_start:
int = 32, char_range: int = 32) -> str:
    """A string of up to `max_length`
characters in the range [`char_start`,
`char_start` + `char_range`)"""
    string_length = random.randrange(0,
max_length + 1)
    out = ""
    for i in range(0, string_length):
        out +=
chr(random.randrange(char_start, char_start +
char_range))
    return out
```

Binary File Format

Binary file formats are formats used to store data in a binary form. Unlike plain text files, binary files do not contain human-readable characters and are designed to store data in a more compact and efficient manner. Binary file formats are used for a wide range of purposes, including storing images, audio, video, executable programs, and more.

What is Kaitai Struct?

Kaitai Struct is a declarative language used to describe various binary data structures, laid out in files or in memory: i.e. binary file formats, network stream packet formats, etc.

About Kaitai Struct

Using KS to parse binary file

- Describe the format — i.e. create a .ksy file.
- Use a visualizer to debug the format and ensure that it parses data properly.
- Compile the .ksy file into a target language source file and include that file into your project.
- Add the KS runtime library for your particular language into your project.
- Use the generated class(es) to parse your binary file or stream and access its components.

- This is a simple .ksy format description file that describes the header of a GIF image file:

```
meta:
  id: gif
  file-extension: gif
  endian: le
seq:
  - id: header
    type: header
  - id: logical_screen
    type: logical_screen
types:
  header:
    seq:
      - id: magic
        contents: 'GIF'
      - id: version
        size: 3
  logical_screen:
    seq:
      - id: image_width
        type: u2
      - id: image_height
        type: u2
      - id: flags
        type: u1
      - id: bg_color_index
        type: u1
      - id: pixel_aspect_ratio
        type: u1
```

Kaitai Struct Language

Consider a C-struct

```c
struct {
    char uuid[16];       /* 128-bit UUID */
    char name[24];       /* Name of the animal */
    uint16_t birth_year; /* Year of birth, used to calculate the age */
    double weight;       /* Current weight in kg */
    int32_t rating;      /* Rating, can be negative */
} animal_record;
```

And here is how it would look like in .ksy:

```yaml
meta:
  id: animal_record
  endian: be
seq:
  - id: uuid
    size: 16
  - id: name
    type: str
    size: 24
    encoding: UTF-8
  - id: birth_year
    type: u2
  - id: weight
    type: f8
  - id: rating
    type: s4
```

Kaitai Struct Language

- Everything starts with a **meta** section: this is where we specify top-level info on the whole structure we describe.

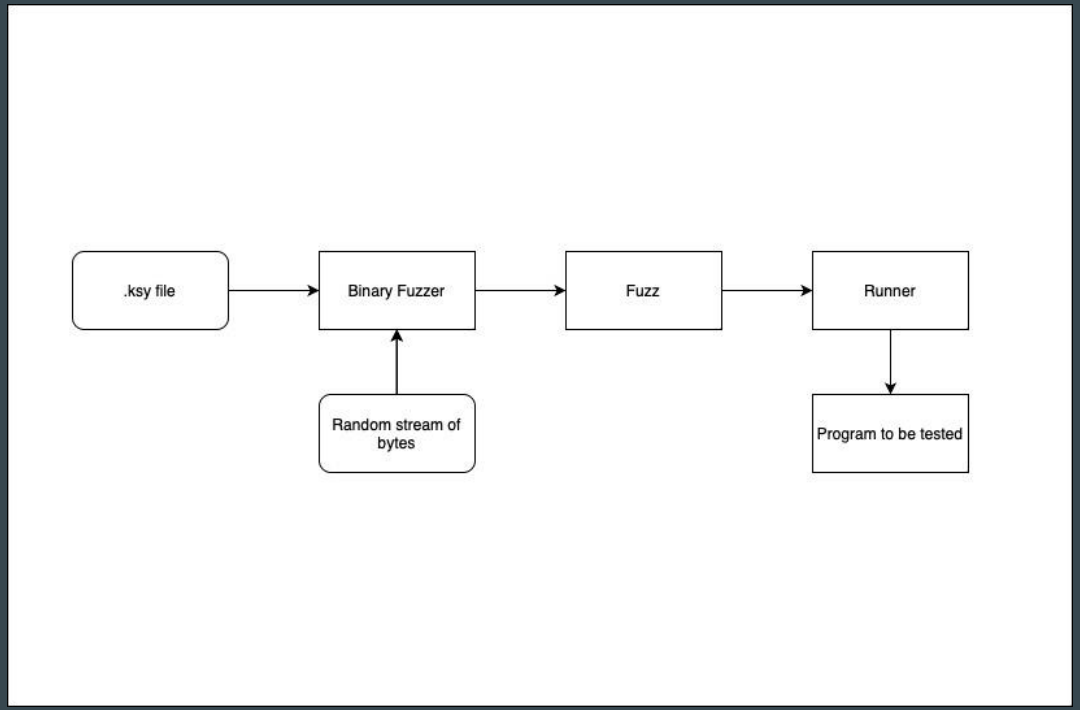
- Then, we use a **seq** element with an array (ordered sequence of elements) in it to describe which attributes this structure consists of. Every attribute includes several keys.

```
# This is a generated file! Please edit source .ksy file and use kaitai-struct-compiler
to rebuild

import kaitaistruct
from kaitaistruct import KaitaiStruct, KaitaiStream, BytesIO


if getattr(kaitaistruct, 'API_VERSION', (0, 9)) < (0, 9):
    raise Exception("Incompatible Kaitai Struct Python API: 0.9 or later is required,
    but you have %s" % (kaitaistruct.__version__))

class AnimalRecord(KaitaiStruct):
    def __init__(self, _io, _parent=None, _root=None):
        self._io = _io
        self._parent = _parent
        self._root = _root if _root else self
        self._read()

    def _read(self):
        self.uuid = self._io.read_bytes(16)
        self.name = (self._io.read_bytes(24)).decode(u"UTF-8")
        self.birth_year = self._io.read_u2be()
        self.weight = self._io.read_f8be()
        self.rating = self._io.read_s4be()
```

Binary Fuzzer

Our project focuses on developing a fuzzer that takes a .ksy file and a random stream of bytes as input and generate test cases in the format specified.

Workflow diagram



Timeline

Week 0

- We emailed Dr. Rahul Gopinath, expressing our interest in contributing to his research works. He responded by agreeing to have a talk with us.
- We familiarized ourselves with the fuzzingbook and his research papers.

Week 1

- A meeting was held on 9th February. We discussed about our project objective which was to develop the Binary Fuzzer.
- We reviewed the user guide of Kaitai Struct to understand the Kaitai Struct Language. We experimented by writing some binary formats in KS language and tested the working of the generated parser with binary data. Also analyzed ksy file and parsers of some formats.

Week 2

- Our second meeting was held on 17th February.
- We identified the file formats of our interest and created .ksy files for some of the formats.

Week 3

- We are asked to build a basic fuzzer with minimal functionality.