Design and Implementation of a Coverage-Guided Ruby Fuzzer

Matt Schwager, Dominik Klemba, Josiah Dykstra
Trail of Bits

What do these have in common?

About me

- **Staff Security Engineer @ Trail of Bits**
- **Background in software engineering and security research**
- **Interested in automated analyses, fuzzing, etc.**
  - “Make the computer sweat, so you don’t have to”
  - Secret ingredient: software usability and ergonomics matter

Fuzzing

- **Dynamic testing methodology to detect security issues and bugs**
  - Throw “random” data at a program and look for crashes
- **Find bugs in software that processes untrusted input**
  - Impact: denial-of-service, remote code execution, logic bugs, etc.
- **Proven track record**
  - Google’s OSS-Fuzz: over 10,000 vulnerabilities and 36,000 bugs
  - American Fuzzy Lop (AFL): thousands of vulnerabilities and bugs
  - rust-fuzz: hundreds of vulnerabilities and bugs

Ruby Fuzzing

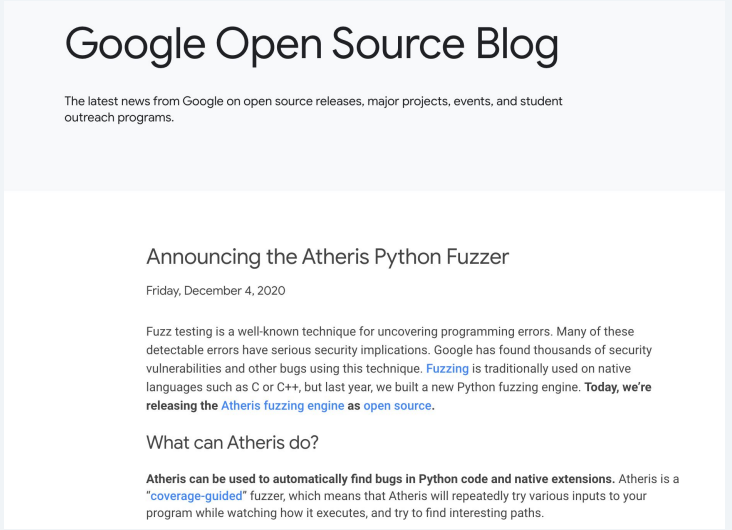
- **C/C++ have libFuzzer and AFL, Go has testing. F, Python has Atheris, Java has Jazzer, JavaScript has jazzer.js, etc.**
- **Ruby is lacking a modern, easy to use, well-integrated fuzzer**
- **The building blocks are available (libFuzzer, Ruby Coverage module)**
    - We just have to assemble them in a usable manner

Existing Ruby Fuzzers

	Fuzzer	Pure Ruby Coverage	C Extension Coverage	Coverage-Guided	Last Activity
kisaten	AFL	Yes	No	Yes	October 2018
afl-ruby	AFL	Yes	No	Yes	December 2020
FuzzBert	custom	Yes	Yes	No	August 2019
Ruzzy	libFuzzer	Yes	Yes	Yes	May 2024

Ruzzy Design Overview

- **Open source**
- **Inspired by Atheris**
- **Support for pure Ruby and Ruby C extensions**
- **Integrated with the libFuzzer ecosystem**
- **Coverage-guided fuzzing**



Google Open Source Blog

The latest news from Google on open source releases, major projects, events, and student outreach programs.

Announcing the Atheris Python Fuzzer

Friday, December 4, 2020

Fuzz testing is a well-known technique for uncovering programming errors. Many of these detectable errors have serious security implications. Google has found thousands of security vulnerabilities and other bugs using this technique. Fuzzing is traditionally used on native languages such as C or C++, but last year, we built a new Python fuzzing engine. **Today, we're releasing the Atheris fuzzing engine as open source.**

What can Atheris do?

Atheris can be used to automatically find bugs in Python code and native extensions. Atheris is a "coverage-guided" fuzzer, which means that Atheris will repeatedly try various inputs to your program while watching how it executes, and try to find interesting paths.

https://opensource.googleblog.com/2020/12/announcing-atheris-python-fuzzer.html

Pure Ruby

```ruby
puts "Hello world!"
```

Ruby C Extensions

```c
#include <ruby.h>


void Init_my_c_ext(void)
{
  ID id_puts = rb_intern("puts");
  VALUE hello_world_str = rb_str_new_cstr("Hello world!");
  rb_funcall(rb_mKernel, id_puts, 1, hello_world_str);
}
```

```ruby
require "my_c_ext"
```


Pure Ruby Fuzzing Target

```ruby
require 'ruzzy'

Ruzzy.trace('test_harness.rb')
```

test_tracer.rb

```ruby
require 'ruzzy'

def fuzzing_target(input)
  if input.length == 4
    if input[0] == 'F'
      if input[1] == 'U'
        if input[2] == 'Z'
          if input[3] == 'Z'
            raise
          end
        end
      end
    end
  end
end

test_one_input = lambda do |data|
  fuzzing_target(data)
  return 0
end

Ruzzy.fuzz(test_one_input)
```

test_harness.rb

```
$ LD_PRELOAD=$(ruby -e 'require "ruzzy"; print Ruzzy::ASAN_PATH') ruby test_tracer.rb

INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 2311041000
...
/app/ruzzy/bin/test_harness.rb:12:in `block in ': unhandled exception
    from /var/lib/gems/3.1.0/gems/ruzzy-0.7.0/lib/ruzzy.rb:15:in `c_fuzz'
    from /var/lib/gems/3.1.0/gems/ruzzy-0.7.0/lib/ruzzy.rb:15:in `fuzz'
    from /app/ruzzy/bin/test_harness.rb:35:in `'
    from bin/test_tracer.rb:7:in `require_relative'
    from bin/test_tracer.rb:7:in `
'
...
SUMMARY: libFuzzer: fuzz target exited
MS: 1 CopyPart-; base unit: 24b4b428cf94c21616893d6f94b30398a49d27cc
0x46,0x55,0x5a,0x5a,
FUZZ
artifact_prefix='./'; Test unit written to ./crash-aea2e3923af219a8956f626558ef32f30a914ebc
Base64: RlVaWg==
```

Ruby C Extension Fuzzing Target

```ruby
def dummy_test_one_input(data)

  require 'dummy/dummy'

  c_dummy_test_one_input(data)

end



def dummy

  fuzz(->(data) { dummy_test_one_input(data) })

end
```

ruzzy.rb

```c
static int _c_dummy_test_one_input(const uint8_t *data, size_t size)
{
    volatile char boom = 'x';

    if (size == 2) {
        if (data[0] == 'H') {
            if (data[1] == 'I') {
                char * volatile ptr = malloc(128);
                ptr[0] = 'x';
                free(ptr);
                boom = ptr[0];
                (void) boom;
            }
        }
    }

    return 0;
}
```

dummy/dummy.c

```
$ export ASAN_OPTIONS="allocator_may_return_null=1:detect_leaks=0:use_sigaltstack=0"

$ LD_PRELOAD=$(ruby -e 'require "ruzzy"; print Ruzzy::ASAN_PATH') \
    ruby -e 'require "ruzzy"; Ruzzy.dummy'

INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 2527961537
...
==45==ERROR: AddressSanitizer: heap-use-after-free on address 0x50c0009bab80 at pc 0xffff99ea1b44 bp
0xffffce8a67d0 sp 0xffffce8a67c8
...
SUMMARY: AddressSanitizer: heap-use-after-free /var/lib/gems/3.1.0/gems/ruzzy-0.7.0/ext/dummy/dummy.c:18:24
in _c_dummy_test_one_input
...
==45==ABORTING
MS: 4 EraseBytes-CopyPart-CopyPart-ChangeBit-; base unit: 410e5346bca8ee150ffd507311dd85789f2e171e
0x48,0x49,
HI
artifact_prefix='./'; Test unit written to ./crash-253420c1158bc6382093d409ce2e9cff5806e980
Base64: SEk=
```


Evaluation and Use Cases

- **Evaluation and Preliminary Results**
  - Evaluated against 15 Ruby libraries
  - Found 4 bugs and unexpected crashes
- **Potential Use Cases**
  - Strong: parsers, decoders, deserializers, etc.
  - Weak: stateful applications, network services, UIs

Ruby libraries tested

GitHub search query: “lang:ruby path:extconf.rb NOT is:archived”

https://github.com/cabo/cbor-ruby
https://github.com/emancu/toml-rb
https://github.com/flori/json
https://github.com/jgarber/redcloth
https://github.com/k0kubun/hamlit
https://github.com/kgiszczak/tomlib
https://github.com/mongodb/bson-ruby
https://github.com/msgpack/msgpack-ruby
https://github.com/ohler55/ox
https://github.com/puma/puma/tree/master/ext/puma_http11
https://github.com/ruby/erb
https://github.com/ruby/openssl
https://github.com/ruby/rdoc
https://github.com/ruby/zlib
https://github.com/slim-template/slim

Future Work

- **Currently integrating with Google's OSS-Fuzz**
- **Improve Ruby interpreter coverage-guidance support**
- **Share work with the Ruby community and fuzz prominent Ruby gems**
- **Fix bugs and support additional functionality**
    - Fuzzed data providers for supporting custom protocols (JSON, YAML, etc.)

Design and Implementation of a Coverage-Guided Ruby Fuzzer

Matt Schwager
matt.schwager@trailofbits.com

TRAIL OF BITS