Fuzzing тестирование

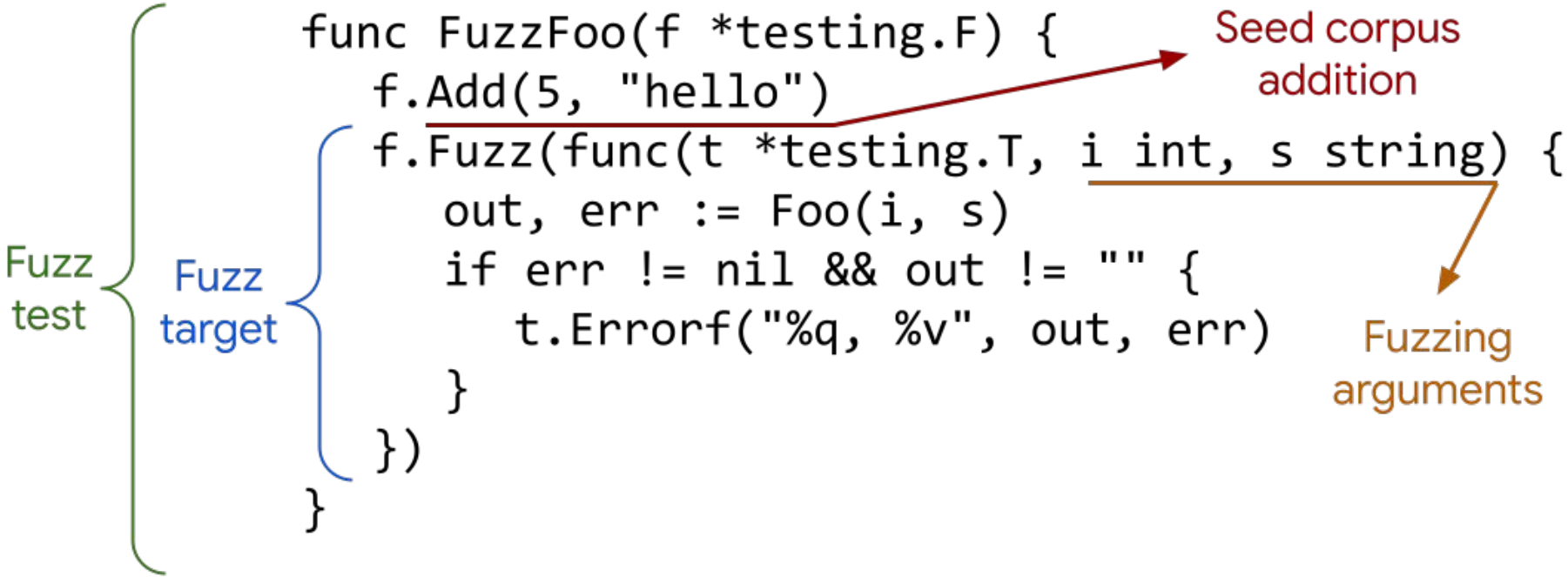
Автор: Денис Лимарев

Введение

Версия Go - 1.18.1beta

Материалы доклада: https://github.com/peakle/meetup/tree/master/go-fuzzing-testing

Что это такое?



Чем может быть полезно?

- Проверка крайних случаев
- Проверка кода для работы с пользовательскими данными
- Поиск новых тестовых случаев

Как использовать?

Обязательно:

- Название теста должно содержать суффикс Fuzz
- Тест должен находиться в *_test.go файле
- Тестовый метод должен содержать параметр *testing.F
- В тестовом методе может быть определен только один fuzzing тест
- Типы параметров источников должны совпадать с типами параметров переданными в fuzz тест
- go test -fuzz=FuzzTestName

Желательно:

- Тестируемая функция не должна долго выполняться
- Тестируемая функция не должна использовать глобальные переменные
- Тестируемая функция должна иметь детерминированное поведение

Ограничения типов на параметры источники

- `string`, `[]byte`
- `int`, `int8`, `int16`, `int32/rune`, `int64`
- `uint`, `uint8/byte`, `uint16`, `uint32`, `uint64`
- `float32`, `float64`
- `bool`

```go
func Reverse(s string) string {
    b := []byte(s)
    for i, j := 0, len(b)-1; i < len(b)/2; i, j = i+1, j-1 {
        b[i], b[j] = b[j], b[i]
    }
    return string(b)
}
```

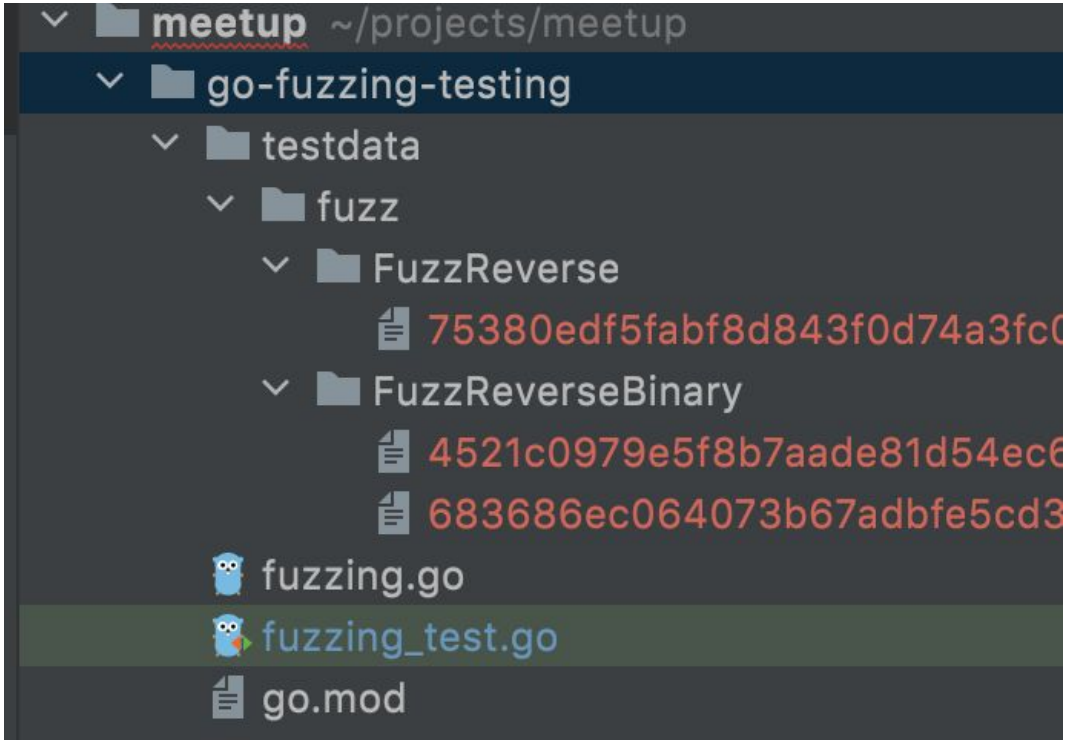
```go
func FuzzReverse(f *testing.F) {
    testcases := []string{"Hello, world", " ", "!12345"}
    for _, tc := range testcases {
        f.Add(tc)  // Use f.Add to provide a seed corpus
    }
    f.Fuzz(func(t *testing.T, orig string) {
        rev := Reverse(orig)
        doubleRev := Reverse(rev)
        if orig != doubleRev {
            t.Errorf( format: "Before: %q, after: %q", orig, doubleRev)
        }
        if utf8.ValidString(orig) && !utf8.ValidString(rev) {
            t.Errorf( format: "Reverse produced invalid UTF-8 string %q", rev)
        }
    })
}
```

```
d.limarev@d-limarev go-fuzzing-testing % go test -fuzz=FuzzReverse
fuzz: elapsed: 0s, gathering baseline coverage: 0/9 completed
failure while testing seed corpus entry: FuzzReverse/75380edf5fabf8d843f0d74a3fc0fad02dc6c12d5a34b44989f874fd86ed3dc9
fuzz: elapsed: 0s, gathering baseline coverage: 3/9 completed
--- FAIL: FuzzReverse (0.02s)
    --- FAIL: FuzzReverse (0.00s)
        fuzzing_test.go:20: Reverse produced invalid UTF-8 string "\xbe\xcb"
```

Какие еще есть возможности/особенности?

- Вынесение своих тестовых случаев в файлы вместо add метода testdata/fuzz/{FuzzTestName}
- При единоразовой неудаче fuzz теста, тест будет каждый раз запускаться с тестовыми данными последней неудачи, до успешного завершения
- Добавлен новый инструмент file2fuzz

```
d.limarev@d-limarev go-fuzzing-testing % echo "foo bar kekekekekekeke" > 1.txt
d.limarev@d-limarev go-fuzzing-testing % file2fuzz -o testdata/fuzz/FuzzReverseBinary 1.txt
```



Примеры фаззинг данных

- `string("İ")`
- `string("Ɣ")`
- `string("")`
- `string("Ā")`

Выводы

- Отлично подходит для проверки кода работающего с пользовательскими данными
- Улучшает понимание своего кода
- Уменьшение кодовой базы тестов
- Документация по fuzzing тестированию https://go.dev/doc/fuzz/
- Туториал по fuzzing тестированию https://go.dev/doc/tutorial/fuzz

Минусы:

- Нет возможности настройки тестовых данных