Fuzzing the CNCF landscape

Adam Korczynski, Ada Logics <adam@adalogics.com>
David Korczynski, Ada Logics <david@adalogics.com>

Agenda

- Overview
- Fuzzing quick intro
- Open source fuzzing with OSS-Fuzz
- The CNCF landscape being fuzzed
- How to fuzz a CNCF project
- Fuzzing results
- Future work

Fuzzing: quick intro

Testing versus fuzzing

Test:

```
MyApi(Input1);
MyApi(Input2);
MyApi(input3);
```

Fuzzing:

```
while true {
  MyApi(Fuzzer.GenerateInput());
}
```

Slide inspired by “Fuzz or lose, CppCon 2017, Kostya Serebryany

Testing versus fuzzing

Test:

```
MyApi(Input1);
MyApi(Input2);
MyApi(input3);
```

Fuzzing:

```
while true {
  MyApi(Fuzzer.GenerateInput());
}
```

Fuzzing - actual implementation:

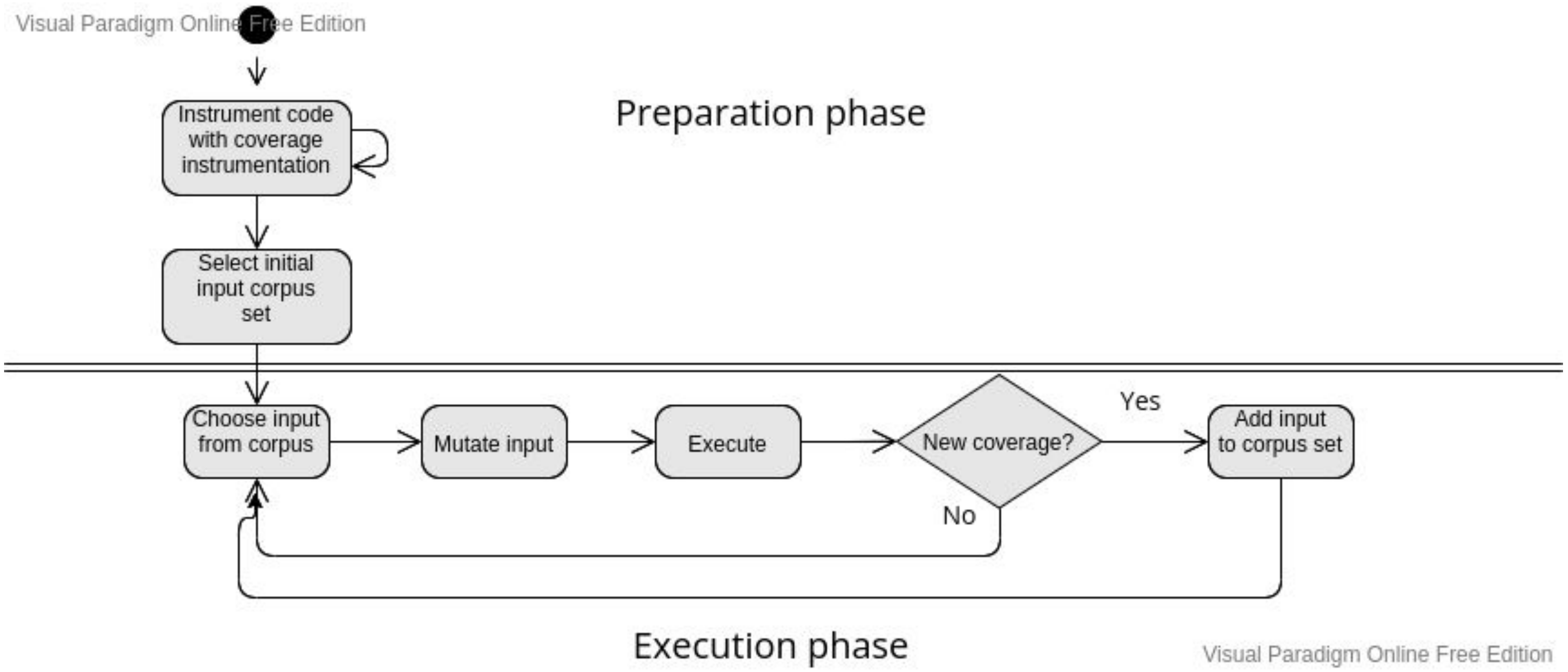
```
func Fuzz(data []byte) int {
  MyApi(string(data))
  return 1
}
```

Slide inspired by “Fuzz or lose, CppCon 2017, Kostya Serebryany

Fuzzing algorithmic underpinnings

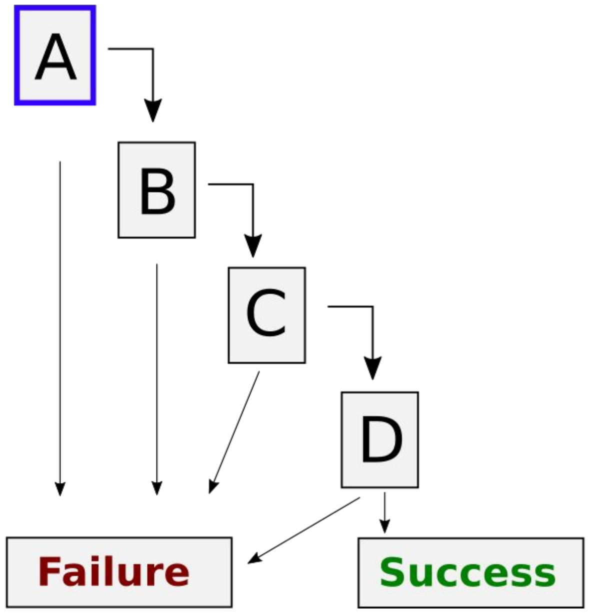
- Common myth: “Fuzzing is random testing, it will never analyse complex code”
  - **False!**
- The origins of fuzzing is in random testing
- Modern day fuzzers are complex genetic mutational algorithms
- Mutations involve a random element
- “Modern day fuzzers” in this case refer to *coverage guided fuzzers*
- Hundreds of academic papers on how to improve fuzzing engines

Coverage guided fuzzing


Visual Paradigm Online Free Edition
Instrument code with coverage instrumentation
Preparation phase
Select initial input corpus set
Choose input from corpus
Mutate input
Execute
New coverage?
Yes
No
Add input to corpus set
Execution phase
Visual Paradigm Online Free Edition

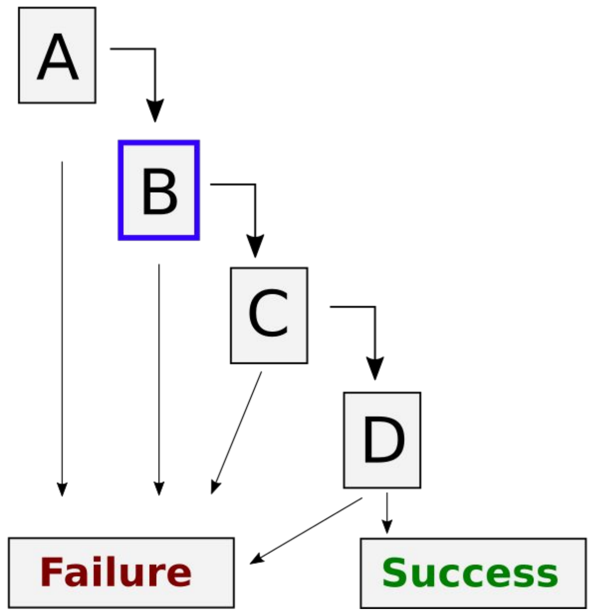
Intuition for coverage-guided fuzzing

Current seed: ____
Probability of progression: 1/256
Probability of no progression: 255/256



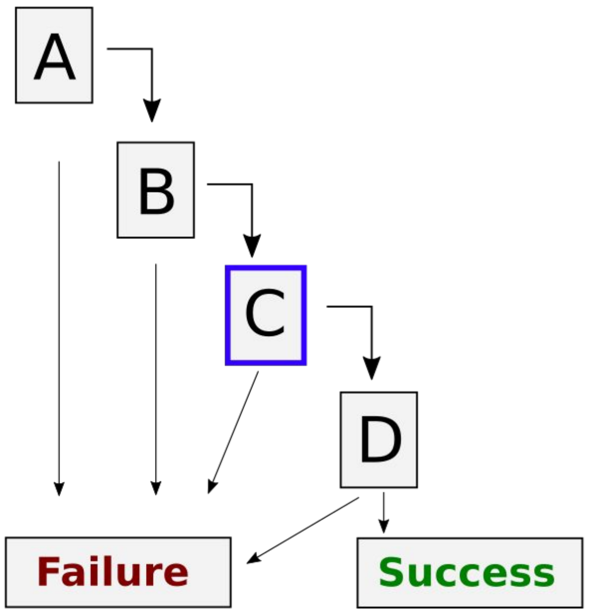
Intuition for coverage-guided fuzzing

Current seed: A___
Probability of progression: 1/256
Probability of no progression: 255/256



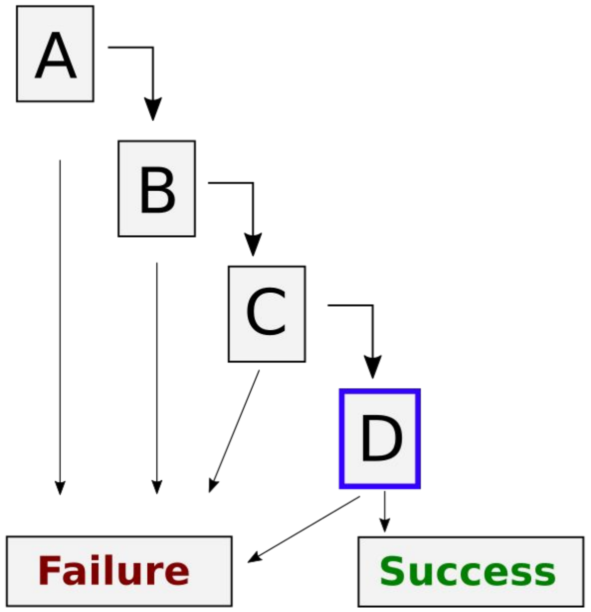
Intuition for coverage-guided fuzzing

Current seed: AB__
Probability of progression: 1/256
Probability of no progression: 255/256



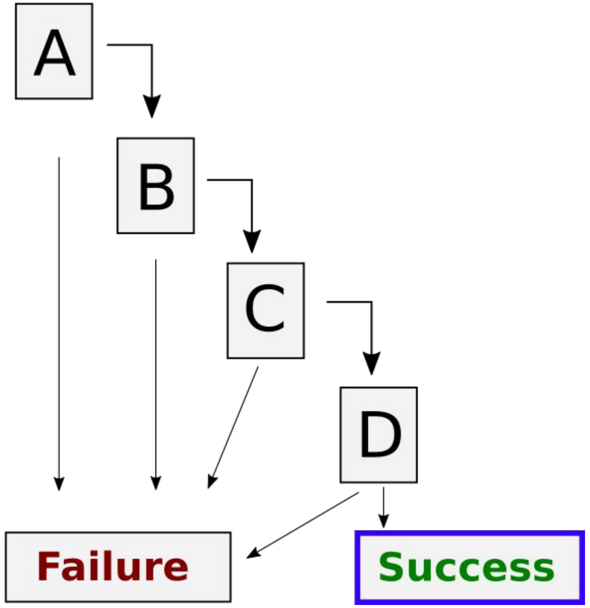
Intuition for coverage-guided fuzzing

Current seed: ABC_
Probability of progression: 1/256
Probability of no progression: 255/256



Intuition for coverage-guided fuzzing

Current seed: ABCD
Bug found after 256 * 4 tries



Which bugs can fuzzers uncover?

- Fuzzers are conceptually only test-case generators. They generate inputs that trigger code paths, i.e. they are not bug identifiers as such but rather execution path identifiers.
- **Memory unsafe languages**
    - Memory corruption.
    - Everything sanitizers tell you
    - …
- **Memory safe languages**
    - Uncaught exceptions
    - Out-of-bounds
    - Nil-pointer dereferences
    - Time outs
    - Out of memory issues
- Behavioural testing

Open source fuzzing with OSS-Fuzz

OSS-Fuzz

- The management of fuzzers is a complex task
- OSS-Fuzz is a free service managed through Github that:
  - Runs fuzzers in open source projects
  - Filters and analyses data from the fuzzers
  - Reports when issues are found to maintainers
  - Tracks when bugs are fixed
  - Suggests improvements and fuzz introspection capabilities
  - https://github.com/google/oss-fuzz
- Integration into OSS-Fuzz is easy
  - "Integrating fuzzing into your open source project with OSS-Fuzz available here

ClusterFuzzLite

- ClusterFuzzLite is an extension to Clusterfuzz which OSS-Fuzz uses
- Runs as part of the CI
- Catches bug early in the process
- Projects can (also) integrate this to catch bugs in the CI, and also integrate into ClusterFuzzLite without full integration into OSS-Fuzz
- https://google.github.io/clusterfuzzlite/

OSS-Fuzz project integration

1. Develop one or more fuzzers for a project
2. Develop a Dockerfile that builds the environment in which the project and fuzzers can be built
3. Develop a build.sh script that builds all the fuzzers
4. Create a project.yaml file with maintainer emails
5. Make a pull request on https://github.com/google/oss-fuzz
6. Once the PR is merged, OSS-Fuzz will run the fuzzers indefinitely

The CNCF landscape being fuzzed

CNCF projects being fuzzed

DISTRIBUTION

envoy

RUNC

fluentbit

cri-o

containerd

etcd

HELM

flux

How to fuzz a CNCF project

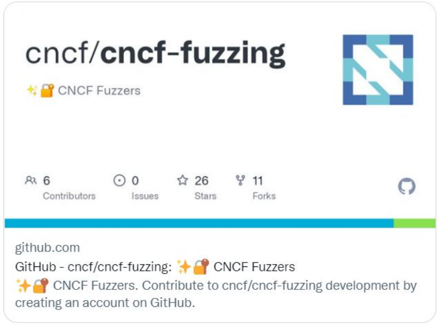
High level process when fuzzing a CNCF project

High level process when fuzzing a CNCF project

1: Initial integration

High level process when fuzzing a CNCF project

1: Initial integration



cncf/**cncf-fuzzing**

✨🔐 CNCF Fuzzers

6 Contributors · 0 Issues · 26 Stars · 11 Forks

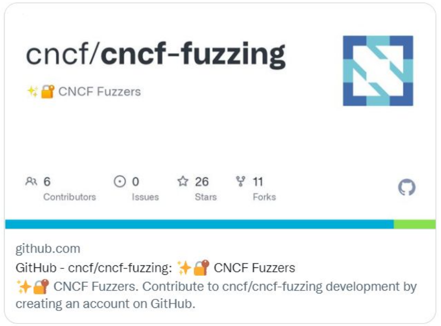
github.com

GitHub - cncf/cncf-fuzzing: ✨🔐 CNCF Fuzzers

✨🔐 CNCF Fuzzers. Contribute to cncf/cncf-fuzzing development by creating an account on GitHub.

High level process when fuzzing a CNCF project

1: Initial integration

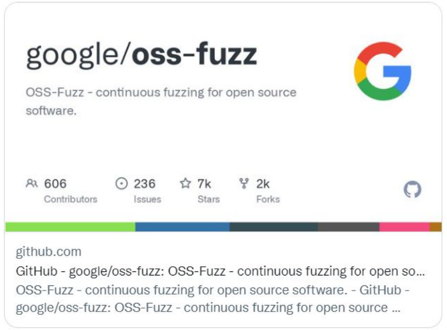


cncf/**cncf-fuzzing**

✨🔐 CNCF Fuzzers

6	0	26	11
Contributors	Issues	Stars	Forks

github.com
GitHub - cncf/cncf-fuzzing: ✨🔐 CNCF Fuzzers
✨🔐 CNCF Fuzzers. Contribute to cncf/cncf-fuzzing development by creating an account on GitHub.



google/**oss-fuzz**

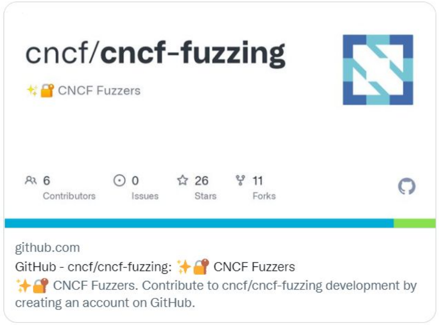
OSS-Fuzz - continuous fuzzing for open source software.

606	236	7k	2k
Contributors	Issues	Stars	Forks

github.com
GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open so...
OSS-Fuzz - continuous fuzzing for open source software. - GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open source ...

High level process when fuzzing a CNCF project

1: Initial integration



cncf/**cncf-fuzzing**

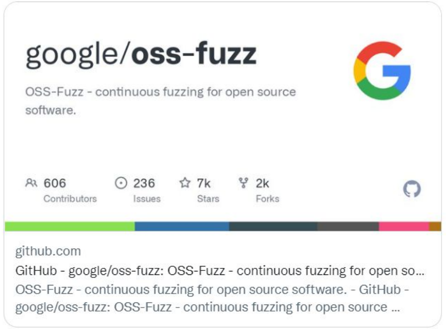
✨🔐 CNCF Fuzzers

6 Contributors · 0 Issues · 26 Stars · 11 Forks

github.com

GitHub - cncf/cncf-fuzzing: ✨🔐 CNCF Fuzzers

✨🔐 CNCF Fuzzers. Contribute to cncf/cncf-fuzzing development by creating an account on GitHub.



google/**oss-fuzz**

OSS-Fuzz - continuous fuzzing for open source software.

606 Contributors · 236 Issues · 7k Stars · 2k Forks

github.com

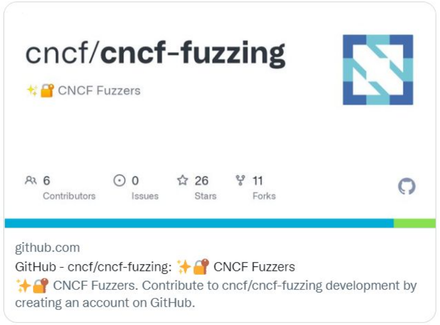
GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open so...

OSS-Fuzz - continuous fuzzing for open source software. - GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open source ...

2: Write a lot of fuzzers

High level process when fuzzing a CNCF project

1: Initial integration

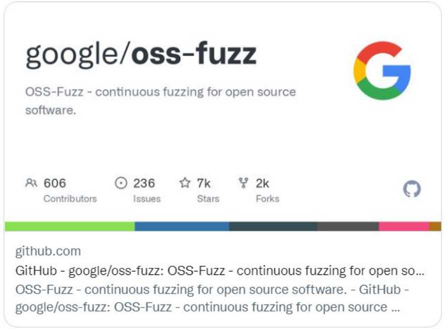


cncf/**cncf-fuzzing**

✨🔐 CNCF Fuzzers

6 Contributors · 0 Issues · 26 Stars · 11 Forks

github.com
GitHub - cncf/cncf-fuzzing: ✨🔐 CNCF Fuzzers
✨🔐 CNCF Fuzzers. Contribute to cncf/cncf-fuzzing development by creating an account on GitHub.



google/**oss-fuzz**

OSS-Fuzz - continuous fuzzing for open source software.

606 Contributors · 236 Issues · 7k Stars · 2k Forks

github.com
GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open so...
OSS-Fuzz - continuous fuzzing for open source software. - GitHub - google/oss-fuzz: OSS-Fuzz - continuous fuzzing for open source ...

2: Write a lot of fuzzers

3: Additional goals depending on project

Integrate as a third-party

- As a third party we bring a lot of knowledge about fuzzing, but have little knowledge of the target project
- Maintainers have a lot of knowledge about target project, but little knowledge of fuzzing
- Maintainers have little time available

Encourage and help maintainers as third parties

- Integrate the fuzzers into the upstream repository
- Do root-cause analysis of bugs reported
- Write more fuzzers
- Inspect the state of the fuzzing to identify limitations
- Take complete control of the fuzzing process

Go-fuzz-headers

https://github.com/AdaLogics/go-fuzz-headers

```
import (
    fuzz “github.com/AdaLogics/go-fuzz-headers”
)
func Fuzz(data []byte) int {
    f := fuzz.NewConsumer(data)

    myStruct := &CustomStruct{}
    err := f.GenerateStruct(myStruct)

    myMap := make(map[string]string)
    err := f.FuzzMap(&myMap)

    var mySlice []string
    err := f.CreateSlice(&mySlice)
    return 1
}
```

Go-fuzz-headers

https://github.com/AdaLogics/go-fuzz-headers

```
import (
    fuzz “github.com/AdaLogics/go-fuzz-headers”
)
func Fuzz(data []byte) int {
    f := fuzz.NewConsumer(data)

    myStruct := &CustomStruct{}
    err := f.GenerateStruct(myStruct)

    myMap := make(map[string]string)
    err := f.FuzzMap(&myMap)

    var mySlice []string
    err := f.CreateSlice(&mySlice)
    return 1
}
```

Go-fuzz-headers

https://github.com/AdaLogics/go-fuzz-headers

```
import (
    fuzz “github.com/AdaLogics/go-fuzz-headers”
)
func Fuzz(data []byte) int {
    f := fuzz.NewConsumer(data)

    myStruct := &CustomStruct{}
    err := f.GenerateStruct(myStruct)

    myMap := make(map[string]string)
    err := f.FuzzMap(&myMap)

    var mySlice []string
    err := f.CreateSlice(&mySlice)
    return 1
}
```

Go-fuzz-headers

https://github.com/AdaLogics/go-fuzz-headers

```
import (
    fuzz “github.com/AdaLogics/go-fuzz-headers”
)
func Fuzz(data []byte) int {
    f := fuzz.NewConsumer(data)

    myStruct := &CustomStruct{}
    err := f.GenerateStruct(myStruct)

    myMap := make(map[string]string)
    err := f.FuzzMap(&myMap)

    var mySlice []string
    err := f.CreateSlice(&mySlice)
    return 1
}
```

Go-fuzz-headers

https://github.com/AdaLogics/go-fuzz-headers

```
import (
    fuzz “github.com/AdaLogics/go-fuzz-headers”
)
func Fuzz(data []byte) int {
    f := fuzz.NewConsumer(data)

    myStruct := &CustomStruct{}
    err := f.GenerateStruct(myStruct)

    myMap := make(map[string]string)
    err := f.FuzzMap(&myMap)

    var mySlice []string
    err := f.CreateSlice(&mySlice)
    return 1
}
```

The spectrum of fuzzing CNCF projects

- There are CNCF projects in all sorts of languages
- Fuzzing by way of OSS-Fuzz can be done in the languages
  - Golang
  - C/C++
  - Rust
  - Python
  - Java

Bugs to find in CNCF projects

- The bugs you find largely depends on the language of the project
- Threat models are not always present
- CNCF projects are often written in memory safe languages
- Dependencies

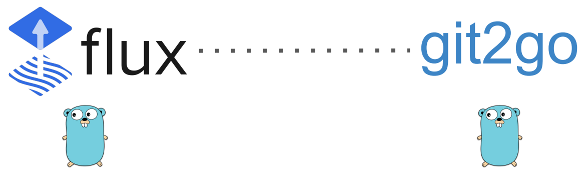
Bugs to find in CNCF projects

- The bugs you find largely depends on the language of the project
- Threat models are not always present
- CNCF projects are often written in memory safe languages
- Dependencies

flux

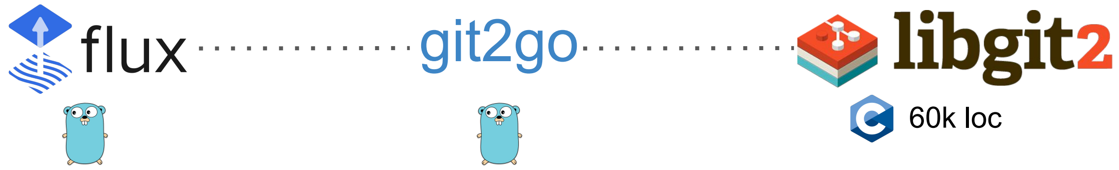
Bugs to find in CNCF projects

- The bugs you find largely depends on the language of the project
- Threat models are not always present
- CNCF projects are often written in memory safe languages
- Dependencies



Bugs to find in CNCF projects

- The bugs you find largely depends on the language of the project
- Threat models are not always present
- CNCF projects are often written in memory safe languages
- Dependencies



Fuzzing results

Querying monorail for issues

- URL: https://bugs.chromium.org/p/oss-fuzz/issues/
    - Select “All issues” in drop-down list.
- Query: “proj=PROJ_NAME Type=Bug-Security,Bug label:Reproducible”
- This count gives over-approximations on what is considered real bugs.

Fuzzing results

Project	**Approx bugs found**	**Language**
Argo	25	Go
Cluster API	4	Go
Containerd	4	Go
CoreDNS	9	Go
CRI-O	4	Go
Distribution	4	Go
Envoy	869	C++
Etcd	15	Go
Fluent-Bit	222	C
Helm	2	Go
Kubernetes	54	Go
Linkerd2 + (-proxy)	7 + (14)	Go + Rust
Runc	0	Go
Vitess	45	Go

Fuzzing results

Project	**Approx bugs found**	**Language**
Argo	25	Go
Cluster API	4	Go
Containerd	4	Go
CoreDNS	9	Go
CRI-O	4	Go
Distribution	4	Go
Envoy	869	C++
Etcd	15	Go
Fluent-Bit	222	C
Helm	2	Go
Kubernetes	54	Go
Linkerd2 + (-proxy)	7 + (14)	Go + Rust
Runc	0	Go
Vitess	45	Go

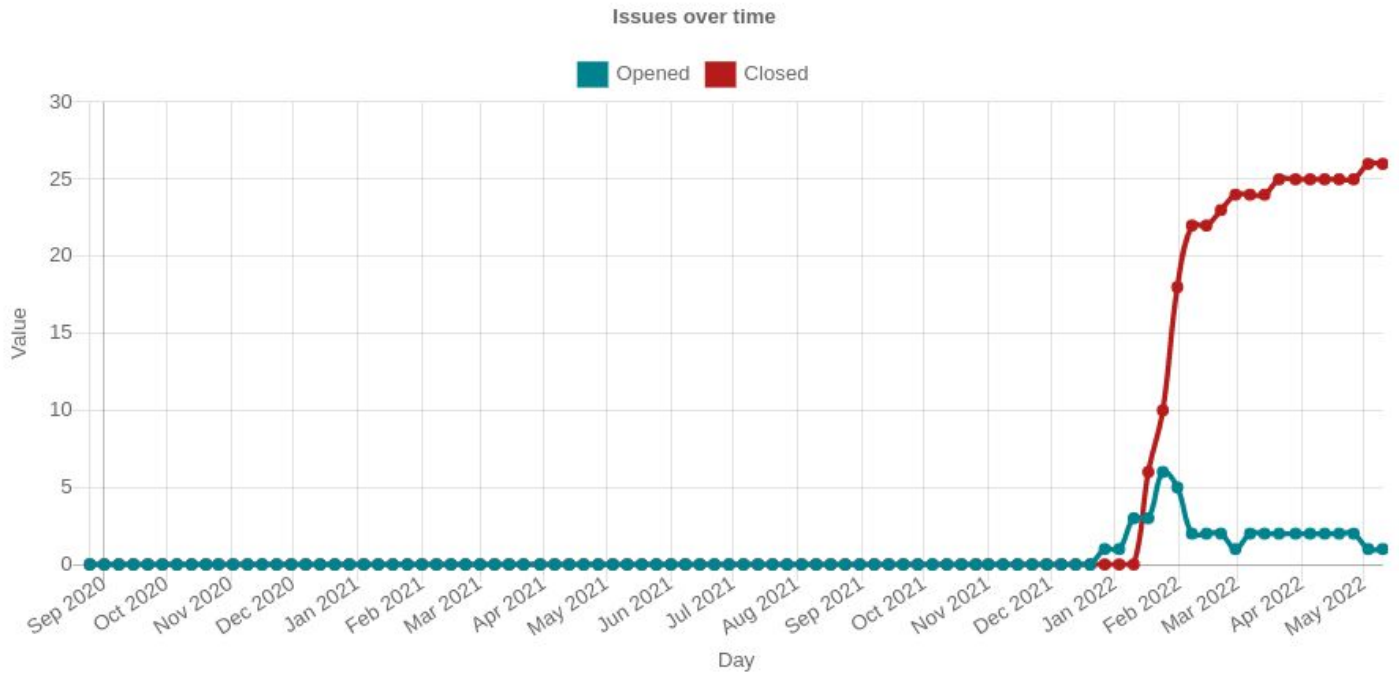
Fuzzing results

Project	**Approx bugs found**	**Language**
Argo	25	Go
Cluster API	4	Go
Containerd	4	Go
CoreDNS	9	Go
CRI-O	4	Go
Distribution	4	Go
Envoy	869	C++
Etcd	15	Go
Fluent-Bit	222	C
Helm	2	Go
Kubernetes	54	Go
Linkerd2 + (-proxy)	7 + (14)	Go + Rust
Runc	0	Go
Vitess	45	Go

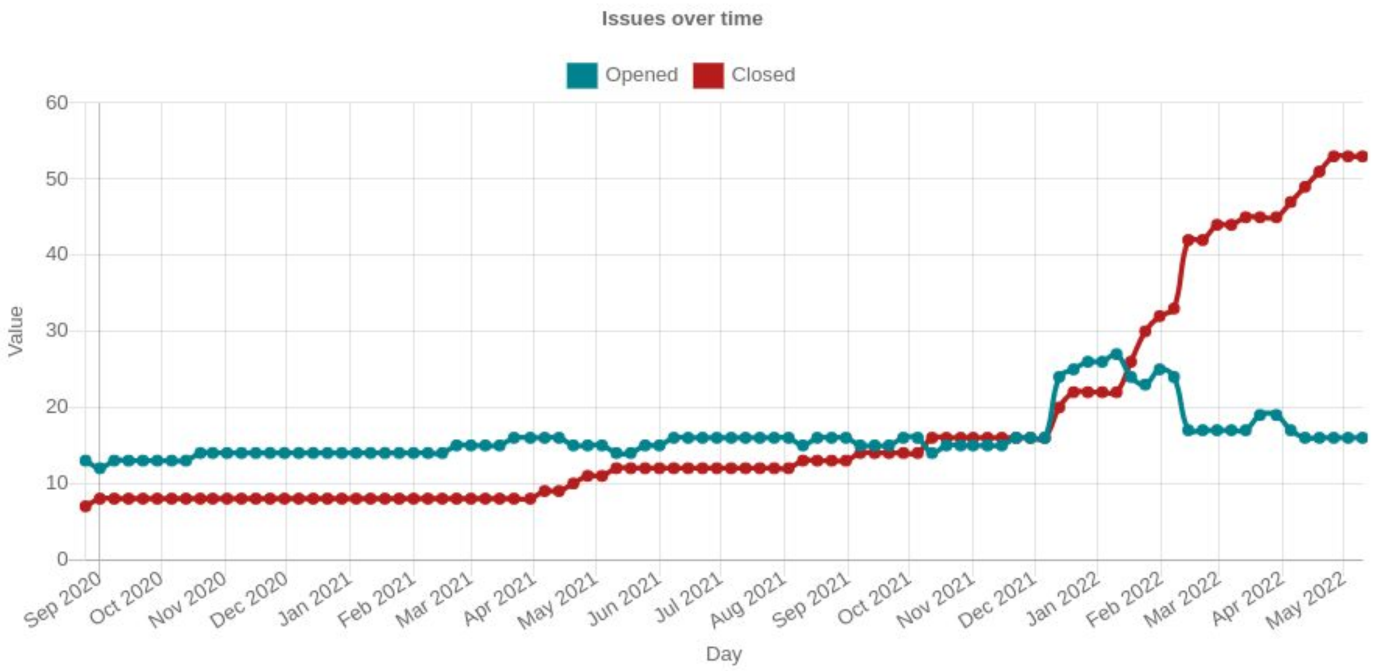
Plotting issues

- Plotting issues from monorail show macro effects of fuzzing
    - Use the “chart” feature on monorail in the previous queries
- Closed issues: bugs reported by OSS-Fuzz that are now closed
- Open issues: bugs reported by OSS-Fuzz that are not resolved.

Argo

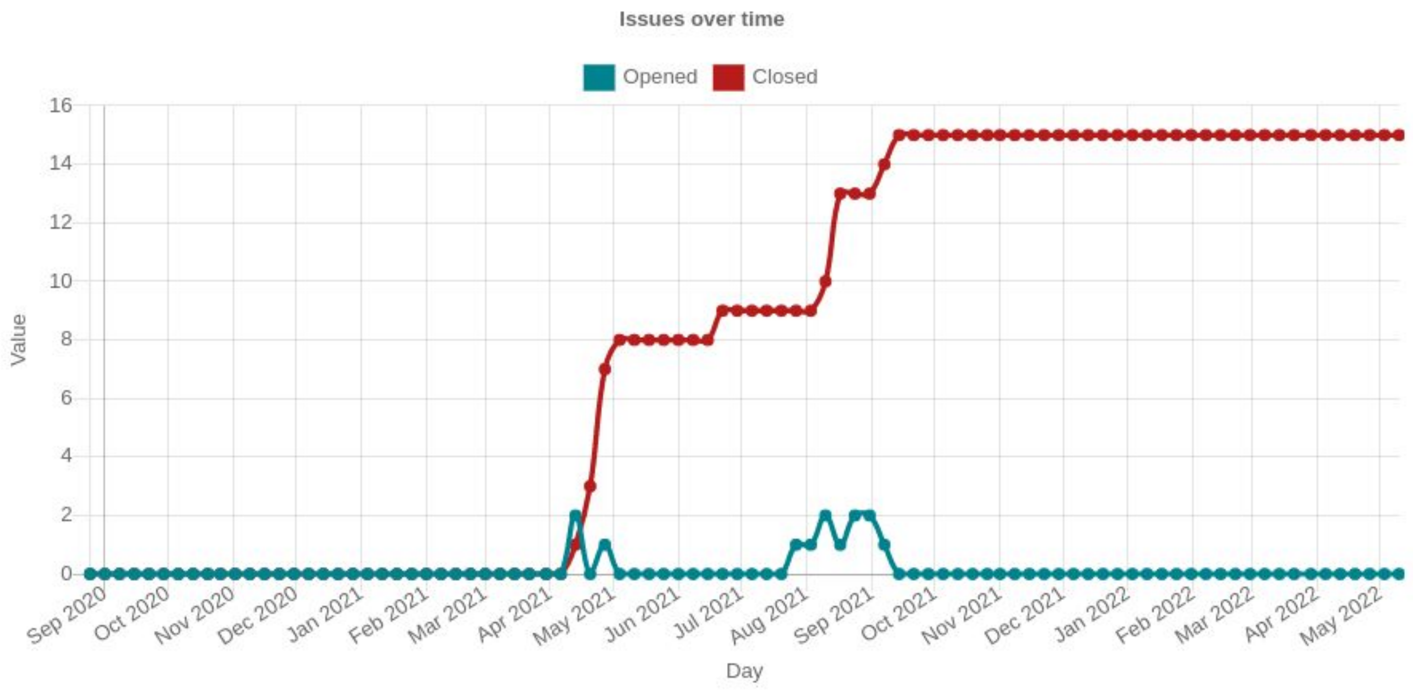

Issues over time
Opened
Closed
Value
Day
30
25
20
15
10
5
0
Sep 2020
Oct 2020
Nov 2020
Dec 2020
Jan 2021
Feb 2021
Mar 2021
Apr 2021
May 2021
Jun 2021
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022

Kubernetes

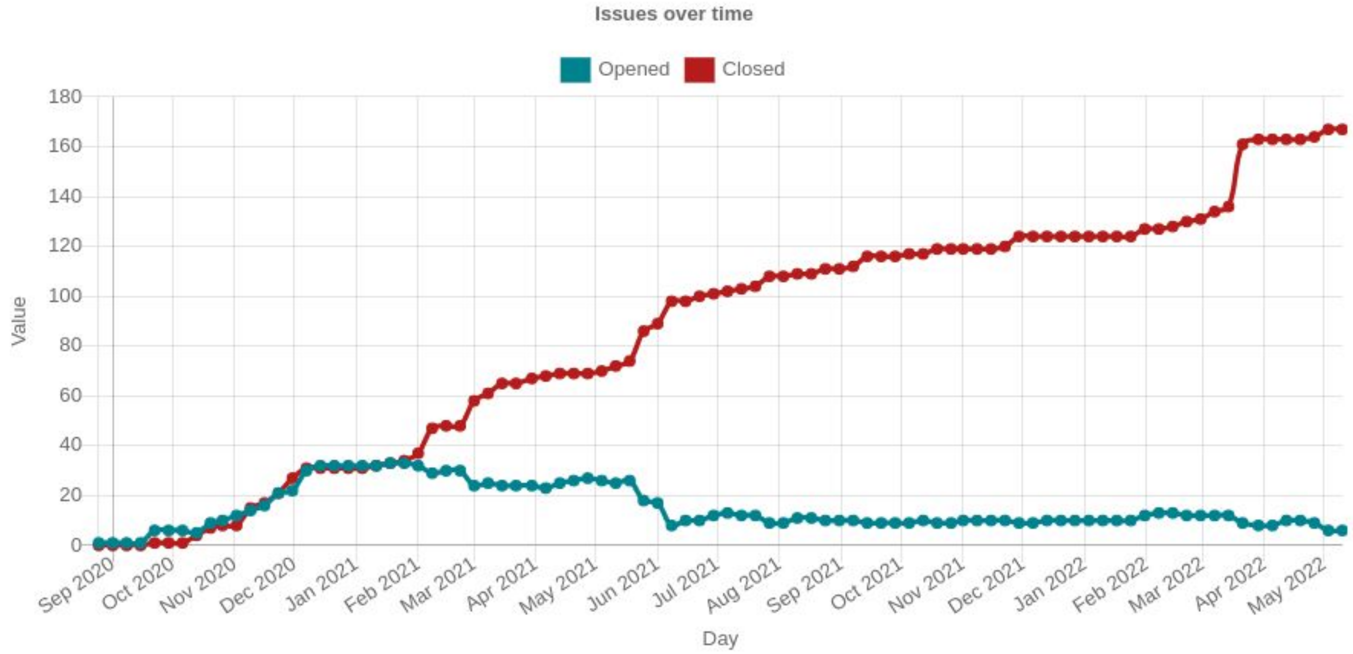


Issues over time
Opened
Closed
60
50
40
30
20
10
0
Value
Sep 2020
Oct 2020
Nov 2020
Dec 2020
Jan 2021
Feb 2021
Mar 2021
Apr 2021
May 2021
Jun 2021
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
Day

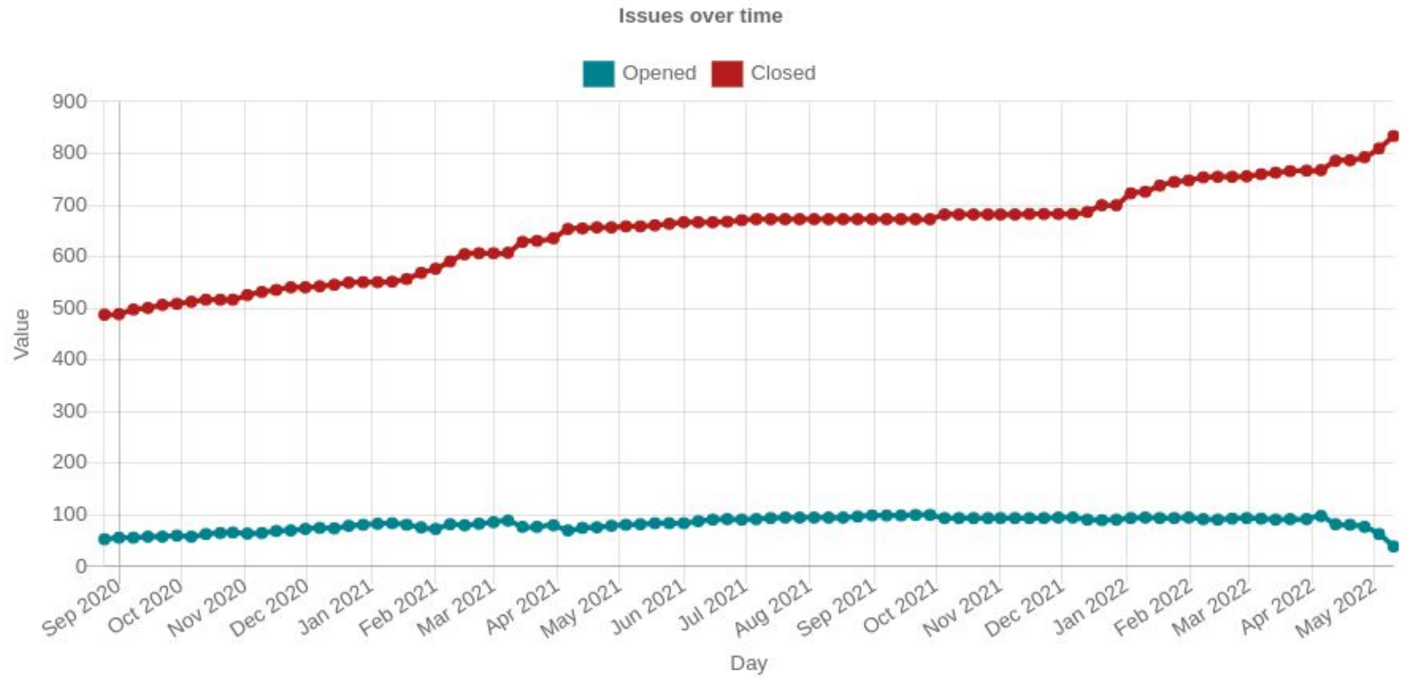
Linkerd2-proxy



Fluent Bit

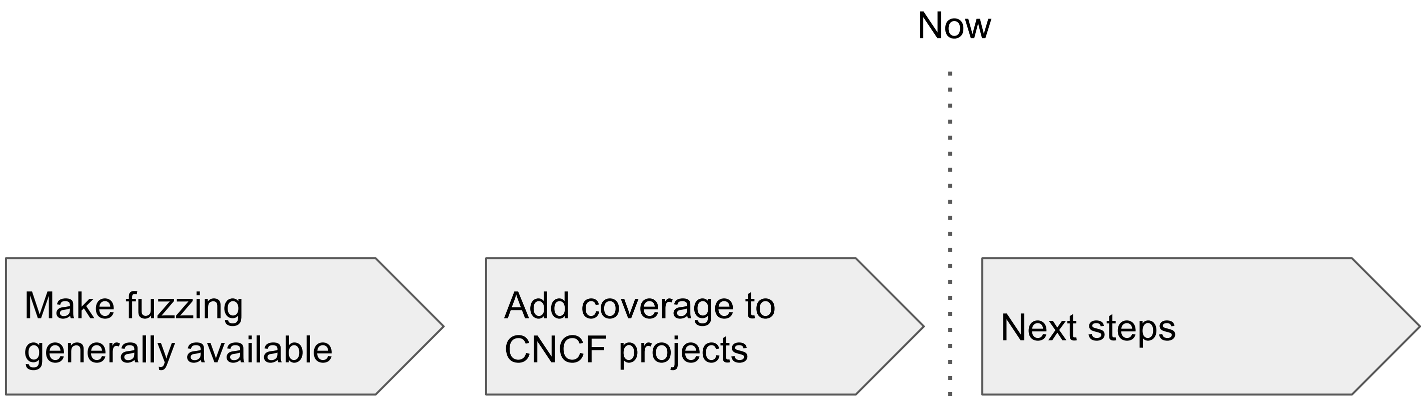

Issues over time
Opened
Closed
Value
180
160
140
120
100
80
60
40
20
0
Sep 2020
Oct 2020
Nov 2020
Dec 2020
Jan 2021
Feb 2021
Mar 2021
Apr 2021
May 2021
Jun 2021
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
Day

Envoy

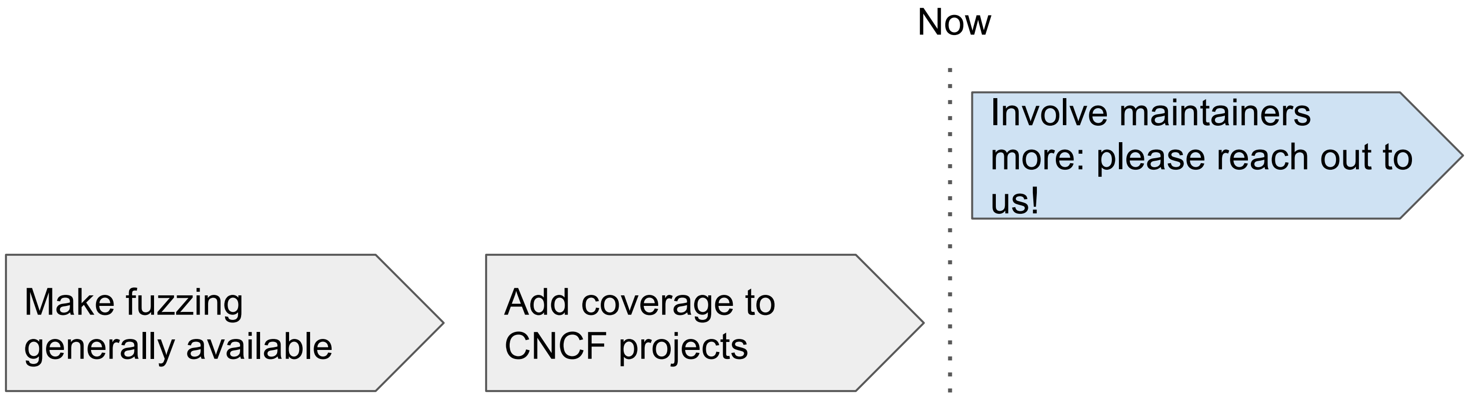

Issues over time
Opened
Closed
Value
900
800
700
600
500
400
300
200
100
0
Sep 2020
Oct 2020
Nov 2020
Dec 2020
Jan 2021
Feb 2021
Mar 2021
Apr 2021
May 2021
Jun 2021
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021
Jan 2022
Feb 2022
Mar 2022
Apr 2022
May 2022
Day

Future work

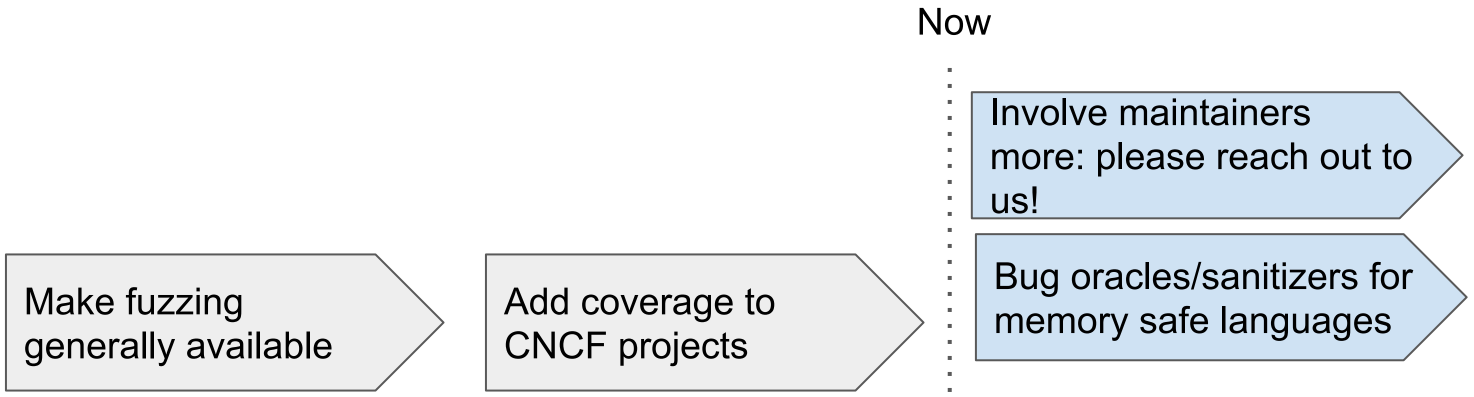
Looking forward


Now
Make fuzzing generally available
Add coverage to CNCF projects
Next steps

Looking forward


Now
Involve maintainers more: please reach out to us!
Make fuzzing generally available
Add coverage to CNCF projects

Looking forward


Now
Make fuzzing generally available
Add coverage to CNCF projects
Involve maintainers more: please reach out to us!
Bug oracles/sanitizers for memory safe languages

Looking forward

Make fuzzing generally available

Add coverage to CNCF projects

Now

Involve maintainers more: please reach out to us!

Bug oracles/sanitizers for memory safe languages

Ensure completeness of projects being fuzzed

Increase maintainer involvement

- Showcase importance of fuzzing through results
- Assist maintainers in fuzzing development/integration
- Provide documentation and training material
- Please reach out via https://github.com/cncf/cncf-fuzzing if you’d like your project fuzzed

Better bug detection

- Applied in a general manner.

Better bug detection

- Applied in a general manner.
- Go fuzzing:
    - Out of range
    - Out of memory
    - Time outs

Better bug detection

- Applied in a general manner.
- Go fuzzing:
  - Out of range
  - Out of memory
  - Time outs
  - **Race conditions**

Better bug detection

- Applied in a general manner.
- Go fuzzing:
    - Out of range
    - Out of memory
    - Time outs
    - **Race conditions**
    - **Logging**

Better bug detection

- Applied in a general manner.
- Go fuzzing:
  - Out of range
  - Out of memory
  - Time outs
  - **Race conditions**
  - **Logging**
  - **File handling**

Better bug detection

- Applied in a general manner.
- Go fuzzing:
    - Out of range
    - Out of memory
    - Time outs
    - **Race conditions**
    - **Logging**
    - **File handling**
    - **Command injections**

Better bug detection

- Applied in a general manner.
- Go fuzzing:
    - Out of range
    - Out of memory
    - Time outs
    - **Race conditions**
    - **Logging**
    - **File handling**
    - **Command injections**
- Massive impact

envoy
RUNC
fluentbit
cri-o
etcd
flux
DISTRIBUTION
HELM
containerd

Conclusions

Conclusions

- 15+ CNCF projects has integrated fuzzing over the last two years
- Integrating with OSS-Fuzz provides a continuous security monitoring on each CNCF project
- Many bugs have been discovered
- Security and reliability bugs
- The fuzzing work has several future works:
    - Extend fuzzers
    - New bug oracles for memory safe languages
    - Involve maintainers more