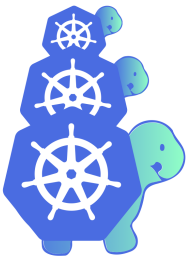
ADA LOGICS

PRESENTS

Fuzzing integration of Cluster API

In collaboration with the Cluster API project maintainers and Cloud Native Computing Foundation.



Authors

Adam Korczynski <adam@adalogics.com>
David Korczynski <david@adalogics.com>
Date: 26th May, 2022

ADA LOGICS

ADA LOGICS

Executive summary

In this engagement, Ada Logics worked on improving Cluster APIs (CAPI) fuzzing effort. The goal was to integrate CAPI into OSS-Fuzz and achieve optimal code coverage.

CAPI had previously done fuzzing with an existing fuzz test: https://github.com/kubernetes-sigs/cluster-api/blob/main/util/conversion/conversion.go#L194, however, this fuzz test is not based on a modern coverage-guided fuzzing engine and CAPI was not set up to run its fuzzer continuously.

The engagement started out by integrating CAPI into OSS-Fuzz and CNCF-fuzzing. From there, Ada Logics gradually and iteratively added more fuzzers and observed the feedback provided by OSS-fuzz. Specifically the test coverage was monitored closely so as to align the outcome with the goal of optimising code coverage. CAPI maintainers were added to the OSS-Fuzz list of recipients to receive bug reports automatically. OSS-Fuzz findings were triaged by the CAPI team and issues were open upstream to facilitate debugging.

Following this engagement, CAPI now has a solid fuzzing foundation and it is left for the CAPI maintainers to take ownership of this and apply their domain specific knowledge to write more fuzzers that test the logic and design that CAPI is meant to comply with.

PR for OSS-fuzz integration: https://github.com/google/oss-fuzz/pull/7292
PR for CNCF-fuzzing integration: https://github.com/cncf/cncf-fuzzing/pull/104

Results summarised

20 fuzzers were developed.
OSS-Fuzz integration for continuous fuzzing set up.
4 crashes were found. • 1 timeout • 2 nil-pointer dereferences • 1 type confusion
All fuzzers are merged into the CNCF-fuzzing repository.

ADA LOGICS

Overview of fuzzers

In this section we will briefly iterate through the fuzzers that were developed and set up to run continuously. All fuzzers are uploaded to the cncf-fuzzing repository. The fuzzers are being built by OSS-fuzz in the build script.

#	Fuzzer Name	Package	Uploaded to
1	FuzzClusterReconcile	https://github.com/kubernetes-sigs/cluster-api/blob/main/internal/controllers/cluster/cluster_controller.go	CNCF-fuzzing
2	FuzzClusterClassReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/clusterclass	CNCF-fuzzing
3	FuzzMachineReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machine	CNCF-fuzzing
4	FuzzMachineDeploymentReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machinedeployment	CNCF-fuzzing
5	FuzzMachineHealthCheckReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machinehealthcheck	CNCF-fuzzing
6	FuzzMachinesetReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machineset	CNCF-fuzzing
7	FuzzModifyImageRepository	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/container	CNCF-fuzzing
8	FuzzModifyImageTag	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/container	CNCF-fuzzing
9	FuzzMatchesMachineSpec	https://github.com/kubernetes-sigs/cluster-api/blob/main/controlplane/kubeadm/internal/filters.go	CNCF-fuzzing
10	FuzzKubeadmControlPlaneReconciler	https://github.com/kubernetes-sigs/cluster-api/tree/main/controlplane/kubeadm/internal/controllers	CNCF-fuzzing
11	FuzzPatch	https://github.com/kubernetes-sigs/cluster-api/blob/main/util/patch/patch.go	CNCF-fuzzing
12	FuzzPatchApply	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/conditions	CNCF-fuzzing
13	FuzzClusterReconcile	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/topology/cluster	CNCF-fuzzing
14	FuzzConversionOfAllTypes	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/conversion	CNCF-fuzzing

15	FuzzYamlParse	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/yaml	CNCF-fuzzing
16	FuzzKubeadmTypesMarshalling	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types	CNCF-fuzzing
17	FuzzUnmarshalClusterConfiguration	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types	CNCF-fuzzing
18	FuzzUnmarshalClusterStatus	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types	CNCF-fuzzing
19	FuzzV1alpha3Conversion	https://github.com/kubernetes-sigs/cluster-api/tree/main/api/v1alpha3	CNCF-fuzzing
20	FuzzWebhookValidation	https://github.com/kubernetes-sigs/cluster-api/tree/main/api/v1beta1	CNCF-fuzzing

Rundown of fuzzers

This section includes a brief description of the fuzzers developed during this engagement.

FuzzClusterReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/cluster_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/blob/main/internal/controllers/cluster/cluster_controller.go

Tests the internal Cluster controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzClusterClassReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/clusterclass_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/clusterclass

Tests the internal Clusterclass controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzMachineReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/machine_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machine

Tests the internal Machine controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzMachineDeploymentReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/machinedeployment_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machinedeployment

Tests the internal Machinedeployment controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzMachineHealthCheckReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/machinehealthcheck_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machinehealthcheck

Tests the internal Machinehealthcheck controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzMachinesetReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/machineset_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/machineset

Tests the internal Machineset controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzModifyImageRepository

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/util_container_fuzzer.go

Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/container

Tests `sigs.k8s.io/cluster-api/util/container.ModifyImageRepository()` with a pseudo-random image name and pseudo-random repository name.

FuzzModifyImageTag

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/util_container_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/container

Tests `sigs.k8s.io/cluster-api/util/container.ModifyImageTag()` with a pseudo-random image name and pseudo-random repository name.

FuzzMatchesMachineSpec

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/kubeadm_internal_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/blob/main/controlplane/kubeadm/internal/filters.go

Tests `sigs.k8s.io/cluster-api/controlplane/kubeadm/internal.MatchesMachineSpec()` with pseudo-randomized parameters.

FuzzKubeadmControlPlaneReconciler

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/internal_kubeadm_controller_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/controlplane/kubeadm/internal/controllers

Tests the unexported `(r *KubeadmControlPlaneReconciler).reconcile()` with a pseudo-random cluster and controlplane.

FuzzPatch

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/patch_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/blob/main/util/patch/patch.go

Patches a pseudo-random object.

FuzzPatchApply

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/conditions_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/conditions

Tests `sigs.k8s.io/cluster-api/util/conditions.(p Patch).Apply()` with a pseudo-random `setter` parameter, and a patch created from pseudo-random cluster objects.

FuzzClusterReconcile

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/topology_cluster_reconciler_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/internal/controllers/topology/cluster

Tests the internal Topology Cluster controller by creating a test client with a pseudo-randomized unstructured object and reconciling with the client.

FuzzConversionOfAllTypes

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/conversion_fuzzer2.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/conversion

Implements a coverage-guided fuzzer of the `FuzzTestFunc` fuzzer designed and developed by the CAPI contributors: https://github.com/kubernetes-sigs/cluster-api/blob/main/util/conversion/conversion.go#L194.

FuzzYamlParse

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/yaml_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/util/yaml

Tests the YAML parsing implemented here: https://github.com/kubernetes-sigs/cluster-api/blob/main/util/yaml/yaml.go#L103

FuzzKubeadmTypesMarshalling

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/bootstrap_kubeadm_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types

Tests marshalling routines in https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types.

FuzzUnmarshalClusterConfiguration

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/bootstrap_kubeadm_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types

Tests `sigs.k8s.io/cluster-api/bootstrap/kubeadm/types.UnmarshalClusterConfiguration()` with a pseudo-random string.

FuzzUnmarshalClusterStatus

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/bootstrap_kubeadm_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/bootstrap/kubeadm/types

Tests sigs.`k8s.io/cluster-api/bootstrap/kubeadm/types.UnmarshalClusterStatus()` with a pseudo-random string.

FuzzV1alpha3Conversion

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/conversion_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/api/v1alpha3

Performs roundtrip testing of `sigs.k8s.io/cluster-api/api/v1beta1` types.

FuzzWebhookValidation

URL	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/v1beta1_machine_webhook_fuzzer.go
Target package	https://github.com/kubernetes-sigs/cluster-api/tree/main/api/v1beta1

Tests validation of `sigs.k8s.io/cluster-api/api/v1beta1` types.

Findings

In this section we iterate through the bugs found by the fuzzers that were written in this engagement.

Throughout the engagement, a few issues were found that were triggered that would not occur in a real-world setting. An example of this is issue 44857: https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=44857. CAPI depends on Kubernetes to do sanitization of its resources, and the fuzzers were initially not considering this need for sanitization. To accommodate this hurdle, Ada Logics added the `GenerateWithCustom` API in the go-fuzz-headers library (https://github.com/AdaLogics/go-fuzz-headers/blob/main/funcs.go#L37) which allows setting values in specific structs when creating pseudo-random resources. This API is used in the `FuzzV1alpha3Conversion` fuzzer.

All issues that were found in the roundtrip fuzzer (FuzzV1alpha3Conversion) have not been included in the list below where the root cause of the issue is due to a practically impossible string values due to missing sanitisation in the fuzzer.

Overview

#	Type	ID	Fixed
1	Nil-pointer dereference	ADA-capi-22-01	Wontfx
2	Nil-pointer dereference	ADA-capi-22-02	Yes
3	Timeout in unmarshalling routine	ADA-capi-22-03	Wontfix
4	Type confusion	ADA-capi-22-04	No

Issue 1

Type	Nil-pointer dereference
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/topology_cluster_reconciler_fuzzer.go#L157
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=45066
ID	ADA-capi-22-01

A nil-dereference was found in `(r *Reconciler).reconcileControlPlane()` (https://github.com/kubernetes-sigs/cluster-api/blob/main/internal/controllers/topology/cluster/reconcile_state.go#L181). The issue was triaged by the CAPI maintainers and was found to not being possible to occur in a real-world scenario. The following comment documents why:

"*I think this a case where the nil pointer is possible in theory, but not in practice. This method is private and is only called in the overall Reconcile method in the topology controller.*

The implementation means that the ClusterBlueprint elements which are returning nil here should not ever be nil at this point in the program flow. We could resolve this (in theory) by adding nil checks for each pointer element of each struct in every function, but we purposefully decided not to do this because we know when and how these methods are called."

ADA LOGICS

Issue 2

Type	Nil-pointer dereference
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/conditions_fuzzer.go#L23
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=45211
ID	ADA-capi-22-02

A nil-pointer dereference was found by the FuzzPatchApply fuzzer. The issue was triaged by the CAPI maintainers and was fixed by implementing nil-checks in a general manner throughout the CAPI codebase.

The issue was fixed in https://github.com/kubernetes-sigs/cluster-api/pull/6401

Issue 3

Type	Timeout in unmarshalling routine
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/bootstrap_kubeadm_fuzzer.go#L124
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=46455
ID	ADA-capi-22-03

The FuzzUnmarshalClusterStatus fuzzer found an issue, whereby a well-crafted string could be passed to sigs.k8s.io/cluster-api/bootstrap/kubeadm/types.UnmarshalClusterStatus() and make CAPI spend excessive cycles on a single request. The CAPI team triaged the crash and found that there was an issue. See discussion in https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=46455 for more info.

However, the part of CAPI in which this issue was found is in the process of being phased out. Upon further discussion, the CAPI team concluded not to fix the issue but wait until the unstable areas of the code can be removed.

ADA LOGICS

Issue 4

Type	Type confusion
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/cluster-api/conversion_fuzzer2.go#L163
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=47602
ID	ADA-capi-22-04

A type confusion was found in (src *MachineHealthCheckList) ConvertTo(dstRaw conversion.Hub) here: https://github.com/kubernetes-sigs/cluster-api/blob/main/api/v1alpha3/conversion.go#L265:

```
func (src *MachineHealthCheckList) ConvertTo(dstRaw conversion.Hub) error {
	dst := dstRaw.(*clusterv1.MachineHealthCheckList)

	return Convert_v1alpha3_MachineHealthCheckList_To_v1beta1_MachineHealthCheckList(src, dst, nil)
}
```

If dstRaw above is not of the type `*clusterv1.MachineHealthCheckList`, CAPI will panic with “panic: interface conversion: conversion.Hub is OTHER_TYPE, not *v1beta1.MachineHealthCheckList”

The fuzzer that finds this issue is stuck moving forward because it hits the issue. However, from skimming through the code we found that this issue exists across several of the conversion routines of CAPIs custom resources, for example:

https://github.com/kubernetes-sigs/cluster-api/blob/main/api/v1alpha3/conversion.go#L217

```
func (src *MachineDeploymentList) ConvertTo(dstRaw conversion.Hub) error {
	dst := dstRaw.(*clusterv1.MachineDeploymentList)

	return Convert_v1alpha3_MachineDeploymentList_To_v1beta1_MachineDeploymentList(src, dst, nil)
}
```

https://github.com/kubernetes-sigs/cluster-api/blob/main/api/v1alpha3/conversion.go#L223

```
func (dst *MachineDeploymentList) ConvertFrom(srcRaw conversion.Hub) error {
```

```
	src := srcRaw.(*clusterv1.MachineDeploymentList)

	return
Convert_v1beta1_MachineDeploymentList_To_v1alpha3_MachineDeploymentList(
src, dst, nil)
}
```

These last two issues have not been reported by OSS-Fuzz but will be so when the first instance of this issue is fixed.

Advice following engagement

Short-term advice

1. Create a strategy for where the fuzzers should be maintained. They are now hosted at the cncf-fuzzing repository, however it is recommended for the CAPI maintainers to move the fuzzers upstream.
2. Fuzzing will be natively supported in Go 1.18 and it may be worthwhile to rewrite the fuzzers to native Go fuzzers and place them in their respective directories similar to how unit tests are managed. OSS-Fuzz is able to handle native Go fuzzers as of a recent PR, so continuous fuzzing will remain supported.
3. Run the fuzzers in the CI with either CIFuzz or as native Go fuzzers when Go 1.18 is released.
4. Improve the procedures and expectations among the maintainers to respond to reports by OSS-Fuzz.

Long-term advice

1. Assess which parts of the CAPI ecosystem are missing coverage and write fuzzers to cover the missing parts. These fuzzers should run continuously on OSS-Fuzz, and if any bugs are found, they should be triaged and fixed within OSS-fuzz's 90 days disclosure policy.
2. When new code is submitted to CAPI that will not be covered by existing fuzzers, make it a routine to include fuzzers that cover this code.

Conclusions and future work

In this engagement we, Ada Logics, developed an extensive fuzzing suite for the CAPI project. We integrated the fuzzing suite into the OSS-Fuzz fuzzing service such that all fuzzers are now running continuously by OSS-Fuzz indefinitely. A total of 20 fuzzers were developed and a total of 4 crashes were found.

This work was commissioned by the Cloud Native Computing Foundation (CNCF).