PRESENTS

Fuzzing audit of Kubernetes

In collaboration with The Linux Foundation

Authors

Adam Korczynski <adam@adalogics.com>
David Korczynski <david@adalogics.com>
Date: 16th March, 2022

Executive summary

This report details the engagement where Ada Logics improved the fuzzing efforts of Kubernetes. At the beginning of this engagement Kubernetes was being fuzzed to a limited degree, and the efforts, therefore, involved exploring how this could be improved and then implementing these improvements into the existing fuzzing infrastructure.

The goal of the engagement was focused solely on the technical parts of improving the Kubernetes fuzzing efforts and during the engagement a total of 14 code bugs were discovered. This report contains a description of the fuzzers integrated, the bugs found and also a set of recommendations that we believe will help the Kubernetes team to further adopt fuzzing moving forward.

Results summarised

40 fuzzers were developed.

14 issues were found.

- 3 logic bug
- 2 unused parameters
- 4 index/slice bounds out of range
- 5 time out

All were set up to run continuously by way of OSS-fuzz.

All fuzzers were merged into the CNCF-fuzzing repository.

Engagement process and methodology

All work was done against the latest master branch and the first step was to create an extended set of fuzzers for the Kubernetes itself. Once the fuzzers had been written, the next step was to set these up to run continuously by way of OSS-fuzz and the OSS-fuzz integration was set up in such a manager that all fuzzers running continuously will test the latest master branch. All fuzzers were implemented by way of the go-fuzz engine which was the go engine supported by OSS-fuzz at the time of the engagement.

All fuzzers were set up to run continuously as soon as they were ready, and development progressed based on the feedback provided by OSS-fuzz. Because of this development process, some noise in the form of false positives was expected, but this process allowed us to use the automated features of OSS-fuzz that become increasingly helpful as the number of fuzzers grows.

In total, Ada Logics implemented 40 new fuzzers for the Kubernetes project. The efforts in this work found 14 code bugs in Kubernetes, some of which have been fixed already.

The main focus in this engagement was to test for code errors. Such errors are out of bounds, out of range, nil-pointer dereferences, faulty type assertions, out of memory, off-by-one, infinite loop, timeouts and divide by zero.

Improvement of existing Kubernetes set up
In addition to writing 40 new fuzzers for Kubernetes, we also modified and improved the existing Kubernetes fuzzing set up. Kubernetes had a fuzzing suite in place around the gofuzz engine.This fuzzing suite is built around this roundTrip test that tests that a Kubernetes object can be deep-copied, converted, marshalled and back without loss of data. The logic of the fuzzer is placed in kubernetes/staging/src/k8s.io/apimachinery/pkg/api/apitesting/roundtrip/roundtrip.go and has required little to no maintenance for a long time. The part of the fuzzer where each resource is created is then placed in each resource's respective directory, and from our initial assessment, this logic is well-maintained.

The existing setup is good in the sense that it targets a lot of code. There are, however, limitations to the infrastructure. The existing infrastructure does not utilise the coverage-guided feedback that modern fuzzing engines offer. This is a drawback that likely renders it ineffective compared to a fuzzing engine like go-fuzz. go-fuzz has an interface to the go-fuzz engine that allows a go-fuzz harness to call into a gofuzz harness. This enables the coverage-feedback and also allows OSS-fuzz to run the fuzzers since it supports the go-fuzz engine and not the gofuzz engine. As part of this engagement we wrote go-fuzz harnesses that invoke the existing gofuzz fuzzers and ran them continuously on OSS-fuzz. These fuzzers are running continuously on OSS-Fuzz now and one of these fuzzers found a serialisation bug.

Overview of fuzzers

In this section we will briefly iterate through the fuzzers that were developed and set up to run continuously.

#	Fuzzer Name	Package	Source code
1	FuzzParseQuantity	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
2	FuzzMeta1ParseToLabelSelector	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
3	FuzzParseSelector	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
4	FuzzLabelsParse	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
5	FuzzParseGroupVersion	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
6	FuzzParseResourceArg	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
7	FuzzParseVersion	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
8	FuzzParsePrivateKeyPEM	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
9	FuzzParsePublicKeysPEM	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
10	FuzzParseHostPort	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
11	FuzzUrlsMatch	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
12	FuzzParseCSR	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
13	FuzzParseEnv	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
14	FuzzParseQOSReserve	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
15	FuzzParseCPUSet	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
16	FuzzParseImageName	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
17	FuzzCreateElement	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
18	FuzzApiMarshaling	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
19	FuzzKubeRuntime	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing
20	FuzzSyncPod	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing
21	FuzzStrategicMergePatch	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing
22	FuzzconvertToAPIContainerStatuses	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing
23	FuzzHandlePodCleanups	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing
24	FuzzMakeEnvironmentVariables	k8s.io/kubernetes/pkg/kubelet	CNCF-fuzzing

25	FuzzEntireDeploymentUtil	k8s.io/kubernetes/pkg/controller/deployment/util	CNCF-fuzzing
26	FuzzDeepCopy	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
27	FuzzAesRoundtrip	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
28	FuzzLoadPolicyFromBytes	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
29	RegistryFuzzer	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
30	FuzzUnrecognized	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
31	FuzzRoundTripSpecificKind	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
32	FuzzControllerRoundtrip	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
33	FuzzKubeletSchemeRoundtrip	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
34	FuzzProxySchemeRoundtrip	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
35	FuzzRoundTripType	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
36	FuzzDecodeRemoteConfigSource	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
37	FuzzReadLogs	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
38	FuzzRoundtrip	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
39	FuzzSetDefaults_KubeSchedulerConfiguration	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing
40	FuzzAllValidation	k8s.io/kubernetes/test/fuzz/fuzzing	CNCF-fuzzing

Rundown of fuzzers

In this section we outline each of the fuzzers developed. We present a few sentences about what each fuzzer does and the code that the fuzzer targets. The fuzzers themselves are all available in: https://github.com/cncf/cncf-fuzzing/tree/main/projects/kubernetes

FuzzParseQuantity
Passes a pseudo-random string to `k8s.io/apimachinery/pkg/api/resource.ParseQuantity()`.

FuzzMeta1ParseToLabelSelector
Passes a pseudo-random string to `k8s.io/apimachinery/pkg/apis/meta/v1.ParseToLabelSelector()`.

FuzzParseSelector
Passes a pseudo-random string to `k8s.io/apimachinery/pkg/fields.ParseSelector()`.
FuzzLabelsParse
Passes a pseudo-random string to k8s.io/apimachinery/pkg/labels.Parse().

FuzzParseGroupVersion
Passes a pseudo-random string to `k8s.io/apimachinery/pkg/runtime/schema.ParseGroupVersion()`.

FuzzParseResourceArg
Passes a pseudo-random string to `k8s.io/apimachinery/pkg/runtime/schema.ParseResourceArg()`.

FuzzParseVersion
Creates two `k8s.io/apimachinery/pkg/util/version.Version`'s and compares the two by way of `k8s.io/apimachinery/pkg/util/version.(*Version) AtLeast(*Version)`.

FuzzParsePrivateKeyPEM
Passes a pseudo-random byte slice to `k8s.io/client-go/util/keyutil.ParsePrivateKeyPEM()`.

FuzzParsePublicKeysPEM
Passes a pseudo-random byte slice to `k8s.io/client-go/util/keyutil.ParsePublicKeysPEM()`.

FuzzParseHostPort
Passes a pseudo-random string to `k8s.io/kubernetes/cmd/kubeadm/app/util.ParseHostPort()`.

FuzzUrlsMatch
Passes two pseudo-random strings to `k8s.io/kubernetes/pkg/credentialprovider.URLsMatchStr()`.

FuzzParseCSR
Passes a pseudo-random byte slice to `k8s.io/kubectl/pkg/util/certificate.ParseCSR()`.

FuzzParseEnv
Passes a pseudo-random string slice to `k8s.io/kubectl/pkg/cmd/set/env.ParseEnv()`.

FuzzParseQOSReserve
Creates a pseudo-randomized map[string]string and passes it to `k8s.io/kubernetes/pkg/kubelet/cm.ParseQOSReserved()`.

FuzzParseCPUSet
Creates a `CPUSet` by way of `k8s.io/kubernetes/pkg/kubelet/cm/cpuset.Parse()` and calls the `CPUSet`s `String()` method.

FuzzParseImageName
Passes a pseudo-random string to `k8s.io/kubernetes/pkg/util/parsers.ParseImageName()`.

FuzzCreateElement
Tests `k8s.io/kubectl/pkg/apply/parse.(*Factory).CreateElement()` with pseudo-random `recorded`, `local` and `remote` maps.

FuzzApiMarshaling
Implements a fuzzer that targets all protobuf marshalling and unmarshalling routines in Kubernetes. Some examples of these are:

- `k8s.io/kubelet/pkg/apis/podresources/v1.(*AllocatableResourcesResponse).Unmarshal()`.
- `k8s.io/api/policy/v1beta1.(*PodDisruptionBudgetList).Unmarshal()`.
- `k8s.io/api/rbac/v1beta1.(*ClusterRoleList).Marshal()`.

In total, this fuzzer targets more than 1000 marshalling and unmarshalling routines. Because of the high number of targets, the fuzzer is implemented with more than 30,000 lines of code and has lots of deduplication. This is unfeasible to maintain, and the fuzzer is auto generated at compile time.

FuzzKubeRuntime
Creates a pseudo-randomized `k8s.io/api/core/v1.Pod` and `k8s.io/kubernetes/pkg/kubelet/container.PodStatus`. The pod actions are then computed by way of `k8s.io/kubernetes/pkg/kubelet/kuberuntime.(*kubeGenericRuntimeManager).computePodActions()`.

FuzzSyncPod
Sets up a test kubelet and syncs its pods where one of the pods is pseudo-randomized.

FuzzStrategicMergePatch

Tests `k8s.io/apimachinery/pkg/util/strategicpatch.StrategicMergePatch()` with pseudo-random document and patch and a Node.

FuzzconvertToAPIContainerStatuses
Creates a pseudo-randomized `k8s.io/api/core/v1.Pod`, `k8s.io/kubernetes/pkg/kubelet/container.PodStatus`, `[]k8s.io/api/core/v1.ContainerStatus`, `[]k8s.io/api/core/v1.Container` and tests the Kubelets `convertToAPIContainerStatuses()` method.

FuzzHandlePodCleanups
Creates a test kubelet, adds a pseudo-randomized pod to its `PodContainerManager` and invokes the kubelets `HandlePodCleanups()` method.

FuzzMakeEnvironmentVariables
Sets up a test kubelet and tests its `makeEnvironmentVariables()` method with a pseudo-randomized `k8s.io/api/core/v1.Pod`, `k8s.io/api/core/v1.Container` and string slice.

FuzzEntireDeploymentUtil
Implements a fuzzer that tests the deployment utility APIs in kubernetes/pkg/controller/deployment/util/deployment_util.go. The fuzzer itself selects which API to test in the given iteration, and each API is tested with pseudo-randomized parameters.

FuzzDeepCopy
Implements a fuzzer for the logic defined in https://github.com/kubernetes/kubernetes/blob/master/pkg/api/testing/copy_test.go. The fuzzer is implemented by way of the go-fuzz engine instead of the gofuzz engine to enable coverage guided fuzzing and continuous fuzzing by way of OSS-fuzz.

FuzzAesRoundtrip
Creates a transformer and subsequently does the roundtrip test with pseudo-random bytes:

```
ciphertext, _ := transformer.TransformToStorage()
result, _, _ := transformer.TransformFromStorage()
```

Finally the fuzzer checks whether the initial pseudo-random bytes are equal to `result`.

FuzzLoadPolicyFromBytes
Passses pseudo-random bytes to k8s.io/apiserver/pkg/audit/policy.LoadPolicyFromBytes().

RegistryFuzzer
Implements a fuzzer for the logic defined in kubernetes/staging/src/k8s.io/apiserver/pkg/registry/generic/registry.TestNewCreateOptionsFromUpdateOptions(). The fuzzer is implemented by way of go-fuzz instead of gofuzz.

FuzzUnrecognized

Verifies that:

1. Serialized json -> object
2. Serialized json -> `map[string]interface{}` -> object

… results in the same object.

FuzzRoundTripSpecificKind
Implements a fuzzer identical to `k8s.io/apimachinery/pkg/api/apitesting/roundtrip.RoundTripSpecificKind()` with the minor distinction that it uses the byte slice provided by go-fuzz as the data source instead of `rand.Int63()`. The fuzzer uses a `DaemonSet` GVK.

FuzzControllerRoundtrip
Implements a fuzzer identical to `k8s.io/apimachinery/pkg/api/apitesting/roundtrip.RoundTripTypesWithoutProtobuf()` with the distinction that it uses the byte slice provided by go-fuzz as the data source instead of `rand.Int63()`. The fuzzer uses the `k8s.io/kubernetes/pkg/controller/apis/config/scheme` Scheme.

FuzzKubeletSchemeRoundtrip
Similar to `FuzzControllerRoundtrip` but uses the `k8s.io/kubernetes/pkg/kubelet/apis/config/scheme` Scheme instead.

FuzzProxySchemeRoundtrip
Similar to `FuzzControllerRoundtrip` but uses the `k8s.io/kubernetes/pkg/proxy/apis/config/scheme` Scheme instead.

FuzzRoundTripType
Similar to `FuzzControllerRoundtrip` but uses the `k8s.io/kubernetes/pkg/api/legacyscheme` Scheme instead.

FuzzDecodeRemoteConfigSource
Passes pseudo-random bytes to `k8s.io/kubernetes/pkg/kubelet/kubeletconfig/checkpoint.DecodeRemoteConfigSource()`.

FuzzReadLogs
Creates a logFile with pseudo-random contents and reads it by by way of `k8s.io/kubernetes/pkg/kubelet/kuberuntime/logs().ReadLogs()`.

FuzzRoundtrip
Implements a fuzzer for the roundtrip tests implemented here:

- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/apps/v1/defaults_test.go (`TestSetDefaultDeployment()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/apps/v1beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/apps/v1beta2/defaults_test.go (roundTrip())

- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/autoscaling/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/autoscaling/v2beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/autoscaling/v2beta2/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/batch/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/batch/v1beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/core/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/extensions/v1beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/networking/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/networking/v1beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/scheduling/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/scheduling/v1alpha1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/scheduling/v1beta1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/storage/v1/defaults_test.go (`roundTrip()`)
- https://github.com/kubernetes/kubernetes/blob/master/pkg/apis/storage/v1beta1/defaults_test.go (`roundTrip()`)

FuzzSetDefaults_KubeSchedulerConfiguration
Passes a pseudo-randomized `k8s.io/kubernetes/pkg/scheduler/apis/config/v1beta2.KubeSchedulerConfiguration` to `k8s.io/kubernetes/pkg/scheduler/apis/config/v1beta2.SetDefaults_KubeSchedulerConfiguration()`

FuzzAllValidation
Implements an umbrella fuzzer that selects one of 50 validation APIs and tests that. Each validation API is tested with a pseudo-randomized object corresponding to the APIs parameters.

Findings

In this section we iterate through the bugs found and present root-cause analysis.

#	Type	ID	Fixed
1	Logic bug	Ada-KUB-2022-1	No
2	Unused parameter	Ada-KUB-2022-2	No
3	Unused parameter	Ada-KUB-2022-3	No
4	Slice bounds out of range	Ada-KUB-2022-4	No
5	Slice bounds out of range	Ada-KUB-2022-5	No
6	Timeout	Ada-KUB-2022-6	No
7	Logic bug	Ada-KUB-2022-7	No
8	Logic bug	Ada-KUB-2022-8	No
9	Index out of range	Ada-KUB-2022-9	No
10	Timeout	Ada-KUB-2022-10	No
11	Timeout	Ada-KUB-2022-11	No
12	Timeout	Ada-KUB-2022-12	No
13	Timeout	Ada-KUB-2022-13	No
14	Slice bounds out of range	Ada-KUB-2022-14	No

Issue 1

Type	Logic bug
Source	`k8s.io/kubernetes/test/fuzz/fuzzing`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42286&q=kubernetes&can=1
ID	Ada-KUB-2022-1

A logic bug was found in a Kubernetes serialisation roundtrip fuzzer. Ada Logics did not write the logic for the fuzzer itself, but allowed the fuzzer to run by way of the go-fuzz engine instead of the gofuzz engine which is the engine that it was set up to use. The result of this is that the fuzzer was running as a coverage-guided fuzzer and it was able to run continuously by way of OSS-fuzz.
The bug happened in the process where an object is created, encoded and decoded which is where the roundtrip test failed. These are the encoded bytes of the object:

```
{
  "kind": "KubeControllerManagerConfiguration",
  "apiVersion": "kubecontrollermanager.config.k8s.io/v1alpha1",
  "Generic": {
    "Port": 0,
    "Address": "0.0.0.121",
    "MinResyncPeriod": "2.02116108s",
    "ClientConnection": {
      "kubeconfig": "",
      "acceptContentTypes": "",
      "contentType": "/..",
      "qps": 0.326057,
      "burst": 2021161080
    },
    "ControllerStartInterval": "2411339h5m40.56920076s",
    "LeaderElection": {
      "leaderElect": false,
      "leaseDuration": "2411339h5m40.56920076s",
      "renewDeadline": "2411339h5m40.56920076s",
      "retryPeriod": "2411339h5m40.56920076s",
      "resourceLock": "endpoints",
      "resourceName": "",
      "resourceNamespace": ""
    },
    "Controllers": [
      ""
```

```
    ],
    "Debugging": {
      "enableProfiling": false,
      "enableContentionProfiling": false
    },
    "LeaderMigrationEnabled": false,
    "LeaderMigration": {
      "leaderName": "",
      "resourceLock": "",
      "controllerLeaders": null
    }
  },
  "KubeCloudShared": {
    "CloudProvider": {
      "Name": "",
      "CloudConfigFile": ""
    },
    "ExternalCloudVolumePlugin": "",
    "UseServiceAccountCredentials": false,
    "AllowUntaggedCloud": false,
    "RouteReconciliationPeriod": "2411339h5m40.56920076s",
    "NodeMonitorPeriod": "2411339h5m40.56920076s",
    "ClusterName": "kubernetes",
    "ClusterCIDR": "",
    "AllocateNodeCIDRs": false,
    "CIDRAllocatorType": "",
    "ConfigureCloudRoutes": false,
    "NodeSyncPeriod": "2411339h5m40.56920076s"
  },
  "AttachDetachController": {
    "DisableAttachDetachReconcilerSync": false,
    "ReconcilerSyncLoopPeriod": "2411339h5m40.56920076s"
  },
  "CSRSigningController": {
    "ClusterSigningCertFile": "/",
    "ClusterSigningKeyFile": "/",
    "KubeletServingSignerConfiguration": {
      "CertFile": "",
      "KeyFile": ""
    },
    "KubeletClientSignerConfiguration": {
      "CertFile": "",
      "KeyFile": ""
    },
    "KubeAPIServerClientSignerConfiguration": {
      "CertFile": " ",
```

```
      "KeyFile": ""
    },
    "LegacyUnknownSignerConfiguration": {
      "CertFile": "",
      "KeyFile": ""
    },
    "ClusterSigningDuration": "0s"
  },
  "DaemonSetController": {
    "ConcurrentDaemonSetSyncs": 0
  },
  "DeploymentController": {
    "ConcurrentDeploymentSyncs": 0,
    "DeploymentControllerSyncPeriod": "-2562047h47m16.854775808s"
  },
  "StatefulSetController": {
    "ConcurrentStatefulSetSyncs": 0
  },
  "DeprecatedController": {
    "DeletingPodsQPS": 0,
    "DeletingPodsBurst": 0,
    "RegisterRetryCount": 0
  },
  "EndpointController": {
    "ConcurrentEndpointSyncs": 0,
    "EndpointUpdatesBatchPeriod": "5m4.942678016s"
  },
  "EndpointSliceController": {
    "ConcurrentServiceEndpointSyncs": 0,
    "MaxEndpointsPerSlice": 0,
    "EndpointUpdatesBatchPeriod": "0s"
  },
  "EndpointSliceMirroringController": {
    "MirroringConcurrentServiceEndpointSyncs": 0,
    "MirroringMaxEndpointsPerSubset": 0,
    "MirroringEndpointUpdatesBatchPeriod": "0s"
  },
  "EphemeralVolumeController": {
    "ConcurrentEphemeralVolumeSyncs": 0
  },
  "GarbageCollectorController": {
    "EnableGarbageCollector": false,
    "ConcurrentGCSyncs": 0,
    "GCIgnoredResources": null
  },
  "HPAController": {
```

```
    "HorizontalPodAutoscalerSyncPeriod": "0s",
    "HorizontalPodAutoscalerUpscaleForbiddenWindow": "0s",
    "HorizontalPodAutoscalerDownscaleStabilizationWindow": "0s",
    "HorizontalPodAutoscalerDownscaleForbiddenWindow": "0s",
    "HorizontalPodAutoscalerTolerance": 0,
    "HorizontalPodAutoscalerCPUInitializationPeriod": "0s",
    "HorizontalPodAutoscalerInitialReadinessDelay": "0s"
  },
  "JobController": {
    "ConcurrentJobSyncs": 0
  },
  "CronJobController": {
    "ConcurrentCronJobSyncs": 0
  },
  "NamespaceController": {
    "NamespaceSyncPeriod": "0s",
    "ConcurrentNamespaceSyncs": 0
  },
  "NodeIPAMController": {
    "ServiceCIDR": "",
    "SecondaryServiceCIDR": "",
    "NodeCIDRMaskSize": 0,
    "NodeCIDRMaskSizeIPv4": 0,
    "NodeCIDRMaskSizeIPv6": 0
  },
  "NodeLifecycleController": {
    "EnableTaintManager": false,
    "NodeEvictionRate": 0,
    "SecondaryNodeEvictionRate": 0,
    "NodeStartupGracePeriod": "2.175270912s",
    "NodeMonitorGracePeriod": "0s",
    "PodEvictionTimeout": "0s",
    "LargeClusterSizeThreshold": 0,
    "UnhealthyZoneThreshold": 0.99609375
  },
  "PersistentVolumeBinderController": {
    "PVClaimBinderSyncPeriod": "0s",
    "VolumeConfiguration": {
      "EnableHostPathProvisioning": false,
      "EnableDynamicProvisioning": false,
      "PersistentVolumeRecyclerConfiguration": {
        "MaximumRetry": 0,
        "MinimumTimeoutNFS": 0,
        "PodTemplateFilePathNFS": "",
        "IncrementTimeoutNFS": 0,
        "PodTemplateFilePathHostPath": "",
```

```
        "MinimumTimeoutHostPath": 0,
        "IncrementTimeoutHostPath": 0
      },
      "FlexVolumePluginDir": "/"
    },
    "VolumeHostCIDRDenylist": null,
    "VolumeHostAllowLocalLoopback": false
  },
  "PodGCController": {
    "TerminatedPodGCThreshold": 0
  },
  "ReplicaSetController": {
    "ConcurrentRSSyncs": 0
  },
  "ReplicationController": {
    "ConcurrentRCSyncs": 0
  },
  "ResourceQuotaController": {
    "ResourceQuotaSyncPeriod": "0s",
    "ConcurrentResourceQuotaSyncs": 0
  },
  "SAController": {
    "ServiceAccountKeyFile": "",
    "ConcurrentSATokenSyncs": 0,
    "RootCAFile": ""
  },
  "ServiceController": {
    "ConcurrentServiceSyncs": 0
  },
  "TTLAfterFinishedController": {
    "ConcurrentTTLSyncs": 0
  }
}
```

Issue 2

Type	Unused parameter
Source	kubernetes/pkg/apis/apiserverinternal/validation.ValidateStorageVersionName()
Issue link	N/a
ID	Ada-KUB-2022-2

While reviewing the Kubernetes project, an API was found to accept a parameter that was not used in the APIs body.

```
func ValidateStorageVersionName(name string, prefix bool) []string {
	var allErrs []string
	idx := strings.LastIndex(name, ".")
	if idx < 0 {
		allErrs = append(allErrs, "name must be in the form of
<group>.<resource>")
	} else {
		for _, msg := range
utilvalidation.IsDNS1123Subdomain(name[:idx]) {
			allErrs = append(allErrs, "the group segment "+msg)
		}
		for _, msg := range
utilvalidation.IsDNS1035Label(name[idx+1:]) {
			allErrs = append(allErrs, "the resource segment "+msg)
		}
	}
	return allErrs
}
```

Issue 3

Type	Unused parameter
Source	kubernetes/pkg/apis/apiserverinternal/validation.ValidateStorageVersionUpdate()
ID	Ada-KUB-2022-3

Another API was found to accept parameters that were not used in the body of the API.

```
func ValidateStorageVersionUpdate(sv, oldSV
*apiserverinternal.StorageVersion) field.ErrorList {
	// no error since StorageVersionSpec is an empty spec
	return field.ErrorList{}
}
```

Issue 4

Type	Slice bounds out of range
Source	`k8s.io/apimachinery/pkg/api/resource.(*Quantity).Unmarshal()`
Issue link	TODO
ID	Ada-KUB-2022-4

An issue was found in the unmarshalling routine for `k8s.io/apimachinery/pkg/api/resource.Quantity`. A similar issue was reported last year: https://github.com/kubernetes/kubernetes/issues/101435 but based on the panic found here, CVE-2021-3121 has not been patched properly in Kubernetes. The crash occurs on the marked line below:

```
func (m *Quantity) Unmarshal(data []byte) error {
	l := len(data)
	iNdEx := 0
	for iNdEx < l {
		preIndex := iNdEx
		var wire uint64
		for shift := uint(0); ; shift += 7 {
			if shift >= 64 {
				return ErrIntOverflowGenerated
			}
			if iNdEx >= l {
				return io.ErrUnexpectedEOF
			}
			b := data[iNdEx]
			iNdEx++
			wire |= (uint64(b) & 0x7F) << shift
			if b < 0x80 {
				break
			}
		}
		fieldNum := int32(wire >> 3)
		wireType := int(wire & 0x7)
		if wireType == 4 {
			return fmt.Errorf("proto: Quantity: wiretype end group
for non-group")
		}
		if fieldNum <= 0 {
			return fmt.Errorf("proto: Quantity: illegal tag %d
(wire type %d)", fieldNum, wire)
```

```
            }
            switch fieldNum {
            case 1:
                  if wireType != 2 {
                        return fmt.Errorf("proto: wrong wireType = %d for
field String_", wireType)
                  }
                  var stringLen uint64
                  for shift := uint(0); ; shift += 7 {
                        if shift >= 64 {
                              return ErrIntOverflowGenerated
                        }
                        if iNdEx >= l {
                              return io.ErrUnexpectedEOF
                        }
                        b := data[iNdEx]
                        iNdEx++
                        stringLen |= (uint64(b) & 0x7F) << shift
                        if b < 0x80 {
                              break
                        }
                  }
                  intStringLen := int(stringLen)
                  if intStringLen < 0 {
                        return ErrInvalidLengthGenerated
                  }
                  postIndex := iNdEx + intStringLen
                  if postIndex > l {
                        return io.ErrUnexpectedEOF
                  }
                  s := string(data[iNdEx:postIndex])

                  // BEGIN CUSTOM DECODE
                  p, err := ParseQuantity(s)
                  if err != nil {
                        return err
                  }
                  *m = p
                  // END CUSTOM DECODE

                  iNdEx = postIndex
```

Issue 5

Type	Slice bounds out of range
Source	k8s.io/kubelet/pkg/apis/pluginregistration/v1.(*PluginInfo).Unmarshal()
Issue link	https://oss-fuzz.com/testcase-detail/5744484078452736
ID	Ada-KUB-2022-5

Similar to issue 4, an out-of-range panic was found in the unmarshalling routine of k8s.io/kubelet/pkg/apis/pluginregistration/v1.PluginInfo.

The crash occurs on the marked line below:

```
func (m *PluginInfo) Unmarshal(dAtA []byte) error {
	l := len(dAtA)
	iNdEx := 0
	for iNdEx < l {
		preIndex := iNdEx
		var wire uint64
		for shift := uint(0); ; shift += 7 {
			if shift >= 64 {
				return ErrIntOverflowApi
			}
			if iNdEx >= l {
				return io.ErrUnexpectedEOF
			}
			b := dAtA[iNdEx]
			iNdEx++
			wire |= (uint64(b) & 0x7F) << shift
			if b < 0x80 {
				break
			}
		}
		fieldNum := int32(wire >> 3)
		wireType := int(wire & 0x7)
		if wireType == 4 {
			return fmt.Errorf("proto: PluginInfo: wiretype end
group for non-group")
		}
		if fieldNum <= 0 {
			return fmt.Errorf("proto: PluginInfo: illegal tag %d
(wire type %d)", fieldNum, wire)
		}
```

```
            switch fieldNum {
            case 1:
                  if wireType != 2 {
                        return fmt.Errorf("proto: wrong wireType = %d for
field Type", wireType)
                  }
                  var stringLen uint64
                  for shift := uint(0); ; shift += 7 {
                        if shift >= 64 {
                              return ErrIntOverflowApi
                        }
                        if iNdEx >= l {
                              return io.ErrUnexpectedEOF
                        }
                        b := dAtA[iNdEx]
                        iNdEx++
                        stringLen |= (uint64(b) & 0x7F) << shift
                        if b < 0x80 {
                              break
                        }
                  }
                  intStringLen := int(stringLen)
                  if intStringLen < 0 {
                        return ErrInvalidLengthApi
                  }
                  postIndex := iNdEx + intStringLen
                  if postIndex > l {
                        return io.ErrUnexpectedEOF
                  }
                  m.Type = string(dAtA[iNdEx:postIndex])
                  iNdEx = postIndex
```

Issue 6

Type	Timeout
Source	ParseQuantity
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42288
ID	Ada-KUB-2022-6

A fuzzer found that a well-crafted string could be passed to ParseQuantity to cause a timeout.

The payload passed to ParseQuantity was: “011E011110011110”.

Issue 7

Type	Logic bug
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/kubernetes/apiserver_fuzzer.go
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42289&q=kubernetes&can=1
ID	Ada-KUB-2022-7

The fuzzer that found this issue implements a roundtrip test similar to https://github.com/kubernetes/kubernetes/blob/master/staging/src/k8s.io/apiserver/pkg/registry/generic/registry/store_test.go#L386 (`TestNewCreateOptionsFromUpdateOptions()`). An edge case was found where the roundtrip failed. The stacktrace of the issue:

```
panic: output != input:
  want:
{"apiVersion":"meta.k8s.io/v1","fieldValidation":"\u0000","kind":"CreateOptions"
}
   got: {"apiVersion":"meta.k8s.io/v1","kind":"CreateOptions"}
goroutine 17 [running, locked to thread]:
k8s.io/kubernetes/test/fuzz/fuzzing.RegistryFuzzer({0x6020000000b0, 0x0, 0x1})
  k8s.io/kubernetes/test/fuzz/fuzzing/apiserver_fuzzer.go:70 +0x7bb
main.LLVMFuzzerTestOneInput(...)
  ./main.2709281834.go:21
```

The fuzzer has sinced been modified, and the crashing line is now 90 instead of 70.

Issue 8

Type	Logic bug
Source	https://github.com/cncf/cncf-fuzzing/blob/main/projects/kubernetes/converter_fuzzer.go#L48
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42300
ID	Ada-KUB-2022-8

The fuzzer that found this crash implements a roundtrip test that verifies that:

1. serialized json -> object
2. serialized json -> map[string]interface{} -> object

… produces the same object.

The fuzzer found an issue where this was not the case. Stacktrace:

```
  &fuzzing.F{
    ... // 2 identical fields
    C: nil,
    D: 0,
-   E: 5.0000092e+07,
+   E: 5.0000088e+07,
    F: nil,
    G: nil,
    ... // 2 identical fields
  }
panic: DeepEqual failed
goroutine 17 [running, locked to thread]:
k8s.io/kubernetes/test/fuzz/fuzzing.doUnrecognized({0x10c0006d9dd8, 0x1f},
{0x26c86a0, 0x10c0004a9b80})
  k8s.io/kubernetes/test/fuzz/fuzzing/converter_fuzzer.go:83 +0x3f9
k8s.io/kubernetes/test/fuzz/fuzzing.FuzzUnrecognized(...)
  k8s.io/kubernetes/test/fuzz/fuzzing/converter_fuzzer.go:49
main.LLVMFuzzerTestOneInput(0x603000001900, 0x1f)
  ./main.2041483530.go:21 +0x76
```

Issue 9

Type	Index out of range
Source	`k8s.io/apimachinery/pkg/api/resource.skipGenerated()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42320
ID	Ada-KUB-2022-9

Another out of range panic in protobuf marshalling. The panic happens on the marked line below.

```
func skipGenerated(data []byte) (n int, err error) {
	l := len(data)
	iNdEx := 0
	for iNdEx < l {
		var wire uint64
		for shift := uint(0); ; shift += 7 {
			if shift >= 64 {
				return 0, ErrIntOverflowGenerated
			}
			if iNdEx >= l {
				return 0, io.ErrUnexpectedEOF
			}
			b := data[iNdEx]
			iNdEx++
			wire |= (uint64(b) & 0x7F) << shift
			if b < 0x80 {
				break
			}
		}
		wireType := int(wire & 0x7)
		switch wireType {
		case 0:
			for shift := uint(0); ; shift += 7 {
				if shift >= 64 {
					return 0, ErrIntOverflowGenerated
				}
				if iNdEx >= l {
					return 0, io.ErrUnexpectedEOF
				}
				iNdEx++
				if data[iNdEx-1] < 0x80 {
					break
				}
```

```
                }
                return iNdEx, nil
            case 1:
                iNdEx += 8
                return iNdEx, nil
            case 2:
                var length int
                for shift := uint(0); ; shift += 7 {
                    if shift >= 64 {
                        return 0, ErrIntOverflowGenerated
                    }
                    if iNdEx >= l {
                        return 0, io.ErrUnexpectedEOF
                    }
                    b := data[iNdEx]
                    iNdEx++
                    length |= (int(b) & 0x7F) << shift
                    if b < 0x80 {
                        break
                    }
                }
                iNdEx += length
                if length < 0 {
                    return 0, ErrInvalidLengthGenerated
                }
                return iNdEx, nil
            case 3:
                for {
                    var innerWire uint64
                    var start int = iNdEx
                    for shift := uint(0); ; shift += 7 {
                        if shift >= 64 {
                            return 0, ErrIntOverflowGenerated
                        }
                        if iNdEx >= l {
                            return 0, io.ErrUnexpectedEOF
                        }
                        b := data[iNdEx]
                        iNdEx++
                        innerWire |= (uint64(b) & 0x7F) << shift
                        if b < 0x80 {
                            break
                        }
                    }
                    innerWireType := int(innerWire & 0x7)
                    if innerWireType == 4 {
```

```
                        break
                    }
                    next, err := skipGenerated(data[start:])
                    if err != nil {
                        return 0, err
                    }
                    iNdEx = start + next
                }
                return iNdEx, nil
```

Issue 10

Type	Timeout
Source	`k8s.io/client-go/util/keyutil.ParsePrivateKeyPEM()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42320
ID	Ada-KUB-2022-10

A timeout panic was found in `k8s.io/client-go/util/keyutil.ParsePrivateKeyPEM()`. The issue has subsequently been fixed.

The minimized testcase for this crash is:

```
[214 26 149 0 67 226 32 255 255 255 255 255 255 255 255 255 32 32 32 32
32 255 255 255 255]
```

Issue 11

Type	Timeout
Source	API marshalling fuzzer
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42690
ID	Ada-KUB-2022-11

A timeout panic was found by the api marshalling fuzzer. The issue has subsequently been fixed.

The minimized testcase for this crash is:

```
[128 176 18 18 18 18 18 72 10 48 48 53 101 48 48 48 48 48 48 48 48 48 48
48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48
50 49 51 55 57 49 57 48 54 55 48 50 49 54 55 48 52 54 49 187 187 187 187
187 187 187 187 187 1 0 0 0 0 187 187 59 48 56 54 48 49 53 54 187 187
187 187 187 187 187 255 255 255 1 187 187 187 1 0 187 0 115 0 40 35]
```

Issue 12

Type	Timeout
Source	`k8s.io/kubectl/pkg/util/certificate.ParseCSR()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42692
ID	Ada-KUB-2022-12

A timeout panic was found in `k8s.io/kubectl/pkg/util/certificate.ParseCSR()`. The issue has subsequently been fixed.

The minimized testcase for this crash is too large to include in the report but can be downloaded here: https://oss-fuzz.com/testcase-detail/5730114665578496

Issue 13

Type	Timeout
Source	`k8s.io/client-go/util/keyutil.ParsePublicKeysPEM()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43323
ID	Ada-KUB-2022-13

A timeout panic was found in `k8s.io/client-go/util/keyutil.ParsePublicKeysPEM()`. The issue has subsequently been fixed.

The minimized testcase for this crash is too large to include in the report but can be downloaded here: https://oss-fuzz.com/testcase-detail/6436325601968128

Issue 14

Type	Slice bounds out of range
Source	`github.com/robfig/cron/v3.Parser.Parse()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43912

ID	Ada-KUB-2022-14

A fuzzer found that a k8s.io/kubernetes/pkg/apis/batch/validation.ValidateCronJob() could accept an object that could lead to an out of bounds panic in the github.com/robfig/cron/v3 3rd party dependency.

Stacktrace:

```
panic: runtime error: slice bounds out of range [:-1]
goroutine 17 [running, locked to thread]:
github.com/robfig/cron/v3.Parser.Parse({0x5218828}, {0x10c000361ac0, 0x9})
  github.com/robfig/cron/v3@v3.0.1/parser.go:99 +0xa45
github.com/robfig/cron/v3.ParseStandard(...)
  github.com/robfig/cron/v3@v3.0.1/parser.go:230
k8s.io/kubernetes/pkg/apis/batch/validation.validateScheduleFormat({0x10c000361a
c0, 0x9}, 0x30)
  k8s.io/kubernetes/pkg/apis/batch/validation/validation.go:348 +0x57
k8s.io/kubernetes/pkg/apis/batch/validation.ValidateCronJobSpec(0x10c000203298,
0x4, {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0})
  k8s.io/kubernetes/pkg/apis/batch/validation/validation.go:311 +0x208
k8s.io/kubernetes/pkg/apis/batch/validation.ValidateCronJob(0x10c000203180,
{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0})
  k8s.io/kubernetes/pkg/apis/batch/validation/validation.go:284 +0x12d
k8s.io/kubernetes/test/fuzz/fuzzing.FuzzValidateCronJob({0x60b0000001a1,
0x10c000000680, 0x10c000000708})
  k8s.io/kubernetes/test/fuzz/fuzzing/validation_fuzzer.go:492 +0xbd
k8s.io/kubernetes/test/fuzz/fuzzing.FuzzAllValidation({0x60b0000001a0, 0x61,
0x61})
  k8s.io/kubernetes/test/fuzz/fuzzing/validation_fuzzer.go:94 +0xb73
main.LLVMFuzzerTestOneInput(...)
  ./main.2957556078.go:21
```

Crash:

```go
func (p Parser) Parse(spec string) (Schedule, error) {
	if len(spec) == 0 {
		return nil, fmt.Errorf("empty spec string")
	}

	// Extract timezone if present
	var loc = time.Local
	if strings.HasPrefix(spec, "TZ=") || strings.HasPrefix(spec,
"CRON_TZ=") {
		var err error
		i := strings.Index(spec, " ")
		eq := strings.Index(spec, "=")
		if loc, err = time.LoadLocation(spec[eq+1 : i]); err != nil
{
			return nil, fmt.Errorf("provided bad location %s: %v",
spec[eq+1:i], err)
		}
```

```
            spec = strings.TrimSpace(spec[i:])
        }
```

Advice following engagement

Continuous efforts

Prior to this engagement, Kubernetes had limited exposure to fuzzing. One such exposure is that Kubernetes had integrated into OSS-fuzz with 7 initial fuzzers in 2019. The initial fuzzers were testing the yaml and json parsing, and the work had produced some good results, for example CVE-2019-11254. However, since this initial integration into OSS-fuzz, no further work had been done on the OSS-fuzz side. Considering the size, importance and complexity of the Kubernetes project, this is insufficient. From our point of view, Kubernetes should have constant hands on improving the fuzz coverage and infrastructure around fuzzing. Looking forward, this includes:

- Encourage the community to write fuzzers for contributions.
- Work together across interest groups to create new logic for fuzzers.
- Constantly and continuously improve fuzz coverage.

We hope that this report can give the Kubernetes community a sense of the benefits that fuzzing can have for the project both short- and long-term.

Improve Kubernetes maintainer response to fuzz findings

Ada Logics had difficulties getting response from maintainers on the issues found. This may be related to Kubernetes only having a single maintainer email for receiving bug reports on OSS-Fuzz since October 2019. We advise the community to add as many as possible on the OSS-Fuzz email list for Kubernetes bug reports (https://github.com/google/oss-fuzz/blob/master/projects/kubernetes/project.yaml).

We note furthermore that OSS-fuzz has reported a number of crashes throughout 2020 and 2021 none of which have had any response from the Kubernetes side.

Location of fuzzers

As part of this engagement, Ada Logics merged all fuzzers into the official CNCF fuzzing repo as this is the repo intended for fuzzer development. Kubernetes may want to consider moving the fuzzers into the Kubernetes repository itself for easier maintenance.

Consider the fuzzing engine

As of now, all fuzzers are running well in OSS-fuzz, but in line with moving the fuzzers upstream, it may be beneficial to rewrite the fuzzers to native Go fuzzers as supported in Go 1.18. OSS-fuzz supports continuous fuzzing of native Go fuzzers, and on the Kubernetes side it might help improve further development and lower friction for testing for breakages in the fuzzers.

Run the fuzzers in the CI

Running the fuzzers in Kubernetes' CI will help catch easy-to-find bugs and regressions. The fuzzers can run in Kubernetes' CI by way of OSS-fuzz's CIFuzz, and it could also be made

possible by rewriting the fuzzers to native Go fuzzers, where they can then be run as unit tests.

It is worth noting here that this will add significant time to the CI tests, since 40+ fuzzers need to be built and run. We recommend each fuzzer would run for at least 1 minute during the CI.

Improve the existing fuzzing suite

As mentioned in “Engagement process and methodology” in this report, Kubernetes has an extensive fuzzing suite that is built around the gofuzz engine. A lot of work has been put into it, and Ada Logics did use it in this engagement to find serialisation issues in Kubernetes. However, the fuzzing suite is written in a way that makes it highly inefficient. Here are our recommendations regarding optimization:

1. The fuzzing engine must be switched to a coverage-guided engine.
2. The tests are very slow which will impede the effectiveness of a coverage-guided fuzzer. At the end of this engagement, Ada Logics found that all the gofuzz harnesses that were rewritten to go-fuzz harnesses execute at less than 50 executions per second with one of the fuzzers running at less than 1 execution per second. This should be increased by finding the bottlenecks in the fuzzing suite and unblocking these for faster execution.

Conclusions

Ada Logics engaged with a fuzzing audit of Kubernetes at the end of 2021 and beginning of 2022. The focus of the audit was on improving continuous fuzzing of Kubernetes with OSS-Fuzz, implementing new fuzzers to achieve more code coverage and also optimising the existing fuzzing set up in Kubernetes. A total of 40 new fuzzers were developed, 14 bugs were found, improvements were made to the existing fuzzing infrastructure to make it coverage-based and all of the new fuzzers are running on the OSS-Fuzz fuzzing infrastructure.

We would like to thank the Linux Foundation for funding this audit and also the Kubernetes maintainers for engaging during the audit.