PRESENTS

Fuzzing integration of Argo Project

In collaboration with the Argo Project maintainers and The Linux Foundation

Authors

Adam Korczynski <adam@adalogics.com>
David Korczynski <david@adalogics.com>
Date: 25th February, 2022

Executive summary

This report details the engagement whereby Ada Logics made the fuzzing setup for five projects under the Argo Project ecosystem: Argo Workflows, Argo CD, Argo Events, Argo Rollouts and GitOps Engine. At the time where this engagement began, Argo was not doing any fuzzing on the five projects, and the focus was therefore to create a meaningful and long-lasting fuzzing infrastructure that the maintainers of the Argo Project can use to observe results and adopt into their development.

Argo is an incubating CNCF project with a substantial user base that includes companies like Intuit, Adobe, Nvidia, Tesla, Google, RedHat, Alibaba, BlackRock, CapitalOne, Github, HSBC, Handelsbanken, IBM and Ticketbase. Setting up a fuzzing infrastructure around the Argo ecosystem will help users rely on stable open source software in the production pipelines both now as well as long as the fuzzers run continuously.

Ada Logics developed a total of 41 fuzzers across the five projects in scope and the majority of the fuzzers target Argo CD and Argo Workflows. During the development of these fuzzers, Ada Logics was met with an engaging team from the Argo maintainers whose feedback and willingness to quickly fix the found bugs is a testament to a well-maintained open source project.

The engagement integrated continuous fuzzing by way of OSS-Fuzz and a total of 10 issues were found in production code. By the end of the engagement all issues have been fixed except for two which were found just as the report was being finalised.

Results summarised

41 fuzzers developed.

OSS-Fuzz integration for continuous fuzzing set up.

10 bugs discovered:
- 4 nil-dereferences
- 1 Slice bounds out of range
- 3 Index out of range
- 1 Interface conversion issue
- 1 Out of memory.

All issues fixed during the engagement.

All fuzzers merged into the CNCF-fuzzing repository.

Engagement process and methodology

The goal of this engagement was to write a set of fuzzers for these 5 projects in the Argo Project ecosystem:

1. Argo Workflows
2. Argo CD
3. Argo Events
4. Argo Rollouts
5. GitOps Engine

All work was done against the latest master branch and the first step was to create an extended set of fuzzers for the projects listed. After creating an initial set of fuzzers we set up the OSS-Fuzz integration such that all fuzzers will run continuously with the latest master branch. The fuzzers are implemented by way of the go-fuzz engine which was the go engine supported by OSS-fuzz at the time of the engagement.

The fuzzers were set up to run continuously as soon as they were ready, to allow development to progress based on the feedback provided by OSS-fuzz. This rapid development process can bring some noise in the form of false positives, but the process enables OSS-Fuzz to manage a lot of the tasks related to running the fuzzers which is of great assistance as the number of fuzzers grows.

The main focus in this engagement was to test for code errors that can trigger panics and crashes in the Argo projects. These types of issues include bounds out of range, index out of range, nil-pointer dereference, type confusion, out of memory, off-by-one, infinite loop, timeout and divide by zero.

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	Uploaded to
1	FuzzGitopsDiff	github.com/argoproj/gitops-engine/pkg/diff	CNCF-fuzzing
2	FuzzEventbusReconciler	github.com/argoproj/argo-events/controllers/eventbus	CNCF-fuzzing
3	FuzzSensorController	github.com/argoproj/argo-events/controllers/sensor	CNCF-fuzzing
4	FuzzSensorControllerReconcile	github.com/argoproj/argo-events/controllers/sensor	CNCF-fuzzing
5	FuzzEventsourceReconciler	github.com/argoproj/argo-events/controllers/eventsource	CNCF-fuzzing
6	FuzzResourceReconcile	github.com/argoproj/argo-events/controllers/eventsource	CNCF-fuzzing
7	FuzzValidateEventSource	github.com/argoproj/argo-events/controllers/eventsource	CNCF-fuzzing
8	FuzzValidateProject	github.com/argoproj/argo-cd/v2/server/project	CNCF-fuzzing
9	FuzzParseUnverified	github.com/argoproj/argo-cd/v2/server/project	CNCF-fuzzing
10	FuzzCreateToken	github.com/argoproj/argo-cd/v2/server/project	CNCF-fuzzing
11	FuzzCreateRepoCertificate	github.com/argoproj/argo-cd/v2/util/db	CNCF-fuzzing
12	FuzzUserAgentUnaryServerInterceptor	github.com/argoproj/argo-cd/v2/util/grp	CNCF-fuzzing
13	FuzzuserAgentEnforcer	github.com/argoproj/argo-cd/v2/util/grpc	CNCF-fuzzing
14	FuzzLoadPolicy	github.com/argoproj/argo-cd/v2/util/rbac	CNCF-fuzzing
15	FuzzParseAppInstanceValue	github.com/argoproj/argo-cd/v2/util/argo	CNCF-fuzzing
16	FuzzGetAppName	github.com/argoproj/argo-cd/v2/util/argo	CNCF-fuzzing
17	FuzzImportPGPKeys	github.com/argoproj/argo-cd/v2/util/gpg	CNCF-fuzzing
18	FuzzValidatePGPKeysFromStrin	github.com/argoproj/argo-cd/v2/util/gp	CNCF-fuzzing

	g	g	
19	FuzzValidateAppProject	github.com/argoproj/argo-cd/v2/pkg/apis/application/v1alpha1	CNCF-fuzzing
20	FuzzStateDiff	github.com/argoproj/argo-cd/v2/util/argo/diff	CNCF-fuzzing
21	FuzzSessionmanagerParse	github.com/argoproj/argo-cd/v2/util/session	CNCF-fuzzing
22	FuzzVerifyUsernamePassword	github.com/argoproj/argo-cd/v2/util/session	CNCF-fuzzing
23	FuzzGenerateManifests	github.com/argoproj/argo-cd/v2/reposerver/repository	CNCF-fuzzing
24	FuzzNormalize	github.com/argoproj/argo-cd/v2/util/argo/normalizers	CNCF-fuzzing
25	FuzzWorkflowServer	github.com/argoproj/argo-workflows/v3/server/workflow	CNCF-fuzzing
26	FuzzGetOutputArtifact	github.com/argoproj/argo-workflows/v3/server/artifacts	CNCF-fuzzing
27	FuzzGetOutputArtifactByUID	github.com/argoproj/argo-workflows/v3/server/artifacts	CNCF-fuzzing
28	FuzzGetTaskDependencies	github.com/argoproj/argo-workflows/v3/workflow/common	CNCF-fuzzing
29	FuzzOperator	github.com/argoproj/argo-workflows/v3/workflow/controller	CNCF-fuzzing
30	FuzzWorkflowController	github.com/argoproj/argo-workflows/v3/workflow/controller	CNCF-fuzzing
31	FuzzDecodeLockName	github.com/argoproj/argo-workflows/v3/workflow/sync	CNCF-fuzzing
32	FuzzSubmitWorkflow	github.com/argoproj/argo-workflows/v3/workflow/util	CNCF-fuzzing
33	FuzzWoCRun	github.com/argoproj/argo-workflows/v3/workflow/cron	CNCF-fuzzing
34	FuzzValidateWorkflow	github.com/argoproj/argo-workflows/v3/workflow/validate	CNCF-fuzzing
35	FuzzParseObjects	github.com/argoproj/argo-workflows/v3/workflow/common	CNCF-fuzzing
36	FuzzreconcileAnalysisRun	github.com/argoproj/argo-rollouts/analysis	CNCF-fuzzing
37	FuzzNewWebMetricJsonParser	github.com/argoproj/argo-rollouts/metricproviders/webmetric	CNCF-fuzzing
38	FuzzSendNotifications	github.com/argoproj/argo-rollouts/utils/record	CNCF-fuzzing
39	FuzzPrometheusProvider	github.com/argoproj/argo-rollouts/metr	CNCF-fuzzing

		icproviders/prometheus	
40	FuzzKayenta	github.com/argoproj/argo-rollouts/metricproviders/kayenta	CNCF-fuzzing
41	FuzzSSOAuthorize	github.com/argoproj/argo-workflows/v3/server/auth/sso	CNCF-fuzzing

Rundown of fuzzers

FuzzGitopsDiff
Creates two different slices of `k8s.io/apimachinery/pkg/apis/meta/v1/unstructured.Unstructured` and passes them to `github.com/argoproj/gitops-engine/pkg/diff.Diff()` to test for crashes in the processing, decoding and normalisation routines.

FuzzEventbusReconciler
Creates a pseudo-random `github.com/argoproj/argo-events/pkg/apis/eventbus/v1alpha1.EventBus` and tests its handling in `github.com/argoproj/argo-events/controllers/eventbus.(*reconciler).reconcile()`.

FuzzSensorController
Creates a pseudo-random `github.com/argoproj/argo-events/pkg/apis/sensor/v1alpha1.Sensor` and tests its handling in `github.com/argoproj/argo-events/controllers/sensor.(*reconciler).reconcile()`.

FuzzSensorControllerReconcile
Tests `github.com/argoproj/argo-events/controllers/sensor.Reconcile()` for crashes when handling the incoming `Sensor`.

FuzzEventsourceReconciler
Creates a pseudo-random github.com/argoproj/argo-events/pkg/apis/eventsource/v1alpha1.EventSource and tests its handling in `github.com/argoproj/argo-events/controllers/eventsource.(*reconciler).reconcile()`.

FuzzResourceReconcile
Tests `github.com/argoproj/argo-events/controllers/eventsource.Reconcile()` for crashes when handling the incoming `EventSource`.

FuzzValidateEventSource
Tests for crashes when validating a `github.com/argoproj/argo-events/pkg/apis/eventsource/v1alpha1.EventSource`.

FuzzValidateProject
Tests the validation of `github.com/argoproj/argo-cd/v2/pkg/apis/application/v1alpha1.AppProject`.

FuzzParseUnverified
Tests a dependency of Argo CD that is used for jwt parsing.

FuzzCreateToken
Sets up a project server and creates a token with a randomized `github.com/argoproj/argo-cd/v2/pkg/apiclient/project.ProjectTokenCreateRequest`.

FuzzCreateRepoCertificate
Sets up an `ArgoDB` and creates a repository certificate based on a randomized `github.com/argoproj/argo-cd/v2/pkg/apis/application/v1alpha1.RepositoryCertificateList`.

FuzzUserAgentUnaryServerInterceptor
Tests `github.com/argoproj/argo-cd/v2/util/grpc.UserAgentUnaryServerInterceptor()` with a pseudo-random client name and constraint string.

FuzzuserAgentEnforcer
Similar to `FuzzUserAgentUnaryServerInterceptor` but is more focused on `github.com/argoproj/argo-cd/v2/util/grpc.userAgentEnforcer()`.

FuzzLoadPolicy
Creates a `github.com/argoproj/argo-cd/v2/util/rbac.argocdAdapter` with a randomized `builtinPolicy`, `userDefinedPolicy` and `runtimePolicy` and loads them by calling `github.com/argoproj/argo-cd/v2/util/rbac.LoadPolicy()`.

FuzzParseAppInstanceValue
Tests the resource tracking id parsing with pseudo-random strings.

FuzzGetAppName
Tests the application name retrieval with a randomized unstructured object.

FuzzImportPGPKeys
Tests the importing of PGP keys with pseudo-random file contents.

FuzzValidatePGPKeysFromString
Tests the validation of PGP keys.

FuzzValidateAppProject
Tests the validation of a `github.com/argoproj/argo-cd/v2/pkg/apis/application/v1alpha1.AppProject`.

FuzzStateDiff
Tests the normalization of two different randomized unstructured objects.

FuzzSessionmanagerParse
Tests token verification with a pseudo-random token.

FuzzVerifyUsernamePassword
Tests if a username or password can crash Argo-CD.

FuzzGenerateManifests
Creates a path, creates a set of pseudo-random files and directories and finally generates a manifest from that path.

FuzzNormalize
Tests the normalization of a randomized unstructured object.

FuzzWorkflowServer
Sets up a `github.com/argoproj/argo-workflows/v3/pkg/apiclient/workflow.WorkflowServiceServer` and pseudo-randomly creates, gets, lists, deletes, retries, resubmits, resumes, suspends, terminates, stops, sets, lints and submits workflows against that server.

FuzzGetOutputArtifact
Sets up an artifact server and tests `GetOutputArtifact` with a randomized http request.

FuzzGetOutputArtifactByUID
Sets up an artifact server and tests `GetOutputArtifactByUID` with a randomized http request.

FuzzGetTaskDependencies
Gets the dependencies of a randomized `github.com/argoproj/argo-workflows/v3/pkg/apis/workflow/v1alpha1.DAGTask`.

FuzzOperator
Tests the main operator logic with a randomized workflow.

FuzzWorkflowController
Similar to `FuzzOperator` but unmarshals the workflow from a json or yaml string by way of a helper function.

FuzzDecodeLockName
Tests decoding of pseudo-random lock names.

FuzzSubmitWorkflow
Tests workflow validation and submission of a randomized workflow.

FuzzWoCRun

Runs a `github.com/argoproj/argo-workflows/v3/workflow/cron.cronWfOperationCtx` with a randomized `github.com/argoproj/argo-workflows/v3/workflow/cron.cronWf`.

FuzzValidateWorkflow
Tests validation of a randomized workflow.

FuzzParseObjects
Tests an API for parsing a byte slice into a workflow object.

FuzzreconcileAnalysisRun
Performs a reconcile analysis run with a pseudo-randomized `github.com/argoproj/argo-rollouts/pkg/apis/rollouts/v1alpha1.AnalysisRun`.

FuzzNewWebMetricJsonParser
Tests the creation of a new Webmetric JSON parser with a pseudo-randomized `github.com/argoproj/argo-rollouts/pkg/apis/rollouts/v1alpha1.Metric`.

FuzzSendNotifications
Sends a notification with a pseudo-randomized `github.com/argoproj/argo-rollouts/pkg/apis/rollouts/v1alpha1.Rollout`.

FuzzPrometheusProvider
Creates and runs a Prometheus provider with a pseudo-randomized `github.com/argoproj/argo-rollouts/pkg/apis/rollouts/v1alpha1.Metric`.

FuzzKayenta
Creates and runs a Kayenta provider with a pseudo-randomized `github.com/argoproj/argo-rollouts/pkg/apis/rollouts/v1alpha1.Metric`.

FuzzSSOAuthorize
Tests `github.com/argoproj/argo-workflows/v3/server/auth/sso.(*sso).Authorize()` with a pseudo-random string.

Issues found

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

#	Type	Fixed	ID
1	Nil-dereference in `github.com/argoproj/argo-cd/v2/util/argo/normalizers.(*ignoreNormalizer).Normalize()`.	Yes	ADA-Argo-22-01
2	Nil-dereference in `github.com/argoproj/argo-workflows/v3/workflow/util.ApplySubmitOpts`	Yes	ADA-Argo-22-02
3	Nil-dereference in `github.com/dgrijalva/jwt-go/v4.(*Time).UnmarshalJSON`	Yes	ADA-Argo-22-03
4	“interface conversion: interface {} is nil, not string” in `github.com/argoproj/argo-workflows/v3/util/unstructured/workflow.GetConditions`	Yes	ADA-Argo-22-04
5	Index out of range in `github.com/argoproj/argo-workflows/v3/server/artifacts.(*ArtifactServer).getArtifact`	Yes	ADA-Argo-22-05
6	Slice bounds out of range in `github.com/argoproj/argo-cd/v2/util/rbac.(*argocdAdapter).LoadPolicy`	Yes	ADA-Argo-22-06
7	Nil-dereference in `github.com/argoproj/argo-workflows/v3/server/workflow.(*workflowServer).LintWorkflow`	Yes	ADA-Argo-22-07
8	Index out of range in `github.com/argoproj/argo-workflows/v3/server/artifacts.(*ArtifactServer).GetInputArtifactByUID`	Yes	ADA-Argo-22-08
9	Out of memory panic	No	ADA-Argo-22-09
10	Index out of range	No	ADA-Argo-22-10

Issue 1

Type	Nil-dereference
Source	`github.com/argoproj/argo-cd/v2/util/argo/normalizers.(*ignoreNormalizer).Normalize()`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=42919
Fix	https://github.com/argoproj/argo-cd/pull/8185
ID	ADA-Argo-22-01

If the argument to `github.com/argoproj/argo-cd/v2/util/argo/normalizers.(*ignoreNormalizer).Normalize()` is nil (which is perfectly valid Go code) a nil-dereference would happen when calling `un.GroupVersionKind()`:

```
func (n *ignoreNormalizer) Normalize(un *unstructured.Unstructured)
error {
	matched := make([]normalizerPatch, 0)
	for _, patch := range n.patches {
		groupKind := un.GroupVersionKind().GroupKind()
```

The crash was fixed with a simple check:

```
func (n *ignoreNormalizer) Normalize(un *unstructured.Unstructured)
error {
	if un == nil {
		return fmt.Errorf("invalid argument: unstructured is nil")
	}
	matched := make([]normalizerPatch, 0)
	for _, patch := range n.patches {
		groupKind := un.GroupVersionKind().GroupKind()
```

Issue 2

Type	Nil-dereference
Source	`github.com/argoproj/argo-workflows/v3/workflow/util.ApplySubmitOpts`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43395
Fix	https://github.com/argoproj/argo-workflows/pull/7529
ID	ADA-Argo-22-02

If `ApplySubmitOpts`'s first argument is nil, a nil-dereference crash would occur on the marked lines:

```
func ApplySubmitOpts(wf *wfv1.Workflow, opts *wfv1.SubmitOpts) error {
	if opts == nil {
		opts = &wfv1.SubmitOpts{}
	}
	if opts.Entrypoint != "" {
		wf.Spec.Entrypoint = opts.Entrypoint
	}
	if opts.ServiceAccount != "" {
		wf.Spec.ServiceAccountName = opts.ServiceAccount
	}
	if opts.PodPriorityClassName != "" {
		wf.Spec.PodPriorityClassName = opts.PodPriorityClassName
	}
```

A simple check was enough to fix the crash:

```
func ApplySubmitOpts(wf *wfv1.Workflow, opts *wfv1.SubmitOpts) error {
	if wf == nil {
		return fmt.Errorf("workflow cannot be nil")
	}
	if opts == nil {
		opts = &wfv1.SubmitOpts{}
	}
	if opts.Entrypoint != "" {
		wf.Spec.Entrypoint = opts.Entrypoint
	}
	if opts.ServiceAccount != "" {
		wf.Spec.ServiceAccountName = opts.ServiceAccount
	}
	if opts.PodPriorityClassName != "" {
```

```
            wf.Spec.PodPriorityClassName = opts.PodPriorityClassName
        }
```

Issue 3

Type	Nil-dereference
Source	`github.com/dgrijalva/jwt-go/v4.(*Time).UnmarshalJSON`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43448
Fix	https://github.com/argoproj/argo-cd/pull/8136
ID	ADA-Argo-22-03

Argo-CD was using the archived and unmaintained https://github.com/dgrijalva/jwt-go to handle parsing of jwt tokens. A fuzzer was written to test the parsing routine used by Argo-CD, and a nil-dereference was discovered.

To fix this Argo-CD replaced https://github.com/dgrijalva/jwt-go with the maintained https://github.com/golang-jwt/jwt.

Issue 4

Type	interface conversion: interface {} is nil, not string
Source	`github.com/argoproj/argo-workflows/v3/util/unstructured/workflow.GetConditions`.
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43581
Fix	https://github.com/argoproj/argo-workflows/pull/7556
ID	ADA-Argo-22-04

An unchecked type assertion could be crashed by one of the fuzzers in case either `m["type"]` or `m["status"]` was `nil`:

```
func GetConditions(un *unstructured.Unstructured) wfv1.Conditions {
	if un == nil {
		return nil
	}
	items, _, _ := unstructured.NestedSlice(un.Object, "status",
"conditions")
	var x wfv1.Conditions
	for _, item := range items {
		m, ok := item.(map[string]interface{})
		if !ok {
			return nil
		}
		x = append(x, wfv1.Condition{
			Type:   wfv1.ConditionType(m["type"].(string)),
			Status: metav1.ConditionStatus(m["status"].(string)),
		})
	}
	return x
}
```

The fix was a simple check for `m["type"]` and `m["status"]`:

```
func GetConditions(un *unstructured.Unstructured) wfv1.Conditions {
	if un == nil {
		return nil
	}
	items, _, _ := unstructured.NestedSlice(un.Object, "status",
"conditions")
```

```
	var x wfv1.Conditions
	for _, item := range items {
		m, ok := item.(map[string]interface{})
		if !ok {
			return nil
		}
		_, ok = m["type"].(string)
		if !ok {
			return nil
		}
		_, ok = m["status"].(string)
		if !ok {
			return nil
		}
		x = append(x, wfv1.Condition{
			Type:   wfv1.ConditionType(m["type"].(string)),
			Status: metav1.ConditionStatus(m["status"].(string)),
		})
	}
	return x
}
```

Issue 5

Type	Index out of range
Source	`github.com/argoproj/argo-workflows/v3/server/artifacts.(*ArtifactServer).getArtifact`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43583
Fix	https://github.com/argoproj/argo-workflows/pull/7648
ID	ADA-Argo-22-05

An out of range was observed, where the URL.Path value of an *http.Request would be split by the “/” char. The resulting slice was assumed to have 6 elements. The crash would happen for cases where the resulting slice would have less than 6 elements:

```
func (a *ArtifactServer) getArtifact(w http.ResponseWriter, r
*http.Request, isInput bool) {
	requestPath := strings.SplitN(r.URL.Path, "/", 6)
	namespace := requestPath[2]
	workflowName := requestPath[3]
	nodeId := requestPath[4]
	artifactName := requestPath[5]
```

The crash was fixed by checking that requestPath had 6 elements right after performing the split:

```
func (a *ArtifactServer) getArtifact(w http.ResponseWriter, r
*http.Request, isInput bool) {
	requestPath := strings.SplitN(r.URL.Path, "/", 6)
	if len(requestPath) != 6 {
		a.serverInternalError(errors.New("request path is not
valid"), w)
		return
	}
	namespace := requestPath[2]
	workflowName := requestPath[3]
	nodeId := requestPath[4]
	artifactName := requestPath[5]
```

Issue 6

Type	Slice bounds out of range
Source	`github.com/argoproj/argo-cd/v2/util/rbac.(*argocdAdapter).LoadPolicy`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43657
Fix	https://github.com/argoproj/argo-cd/pull/8186
ID	ADA-Argo-22-06

The crash was observed in `github.com/argoproj/argo-cd/v2/util/rbac.loadPolicyLine` where no range check existed for the `tokens` variable. In the case of zero-length, an out of range panic would happen:

```
func loadPolicyLine(line string, model model.Model) error {
	if line == "" || strings.HasPrefix(line, "#") {
		return nil
	}

	reader := csv.NewReader(strings.NewReader(line))
	reader.TrimLeadingSpace = true
	tokens, err := reader.Read()
	if err != nil {
		return err
	}

	key := tokens[0]
	sec := key[:1]
```

The fix was a simple range check:

```
func loadPolicyLine(line string, model model.Model) error {
	if line == "" || strings.HasPrefix(line, "#") {
		return nil
	}

	reader := csv.NewReader(strings.NewReader(line))
	reader.TrimLeadingSpace = true
	tokens, err := reader.Read()
	if err != nil {
		return err
	}
```

```
        if len(tokens) < 2 || len(tokens[0]) < 1 {
                return fmt.Errorf("invalid RBAC policy: %s", line)
        }

        key := tokens[0]
        sec := key[:1]
```

Issue 7

Type	Nil-dereference
Source	`github.com/argoproj/argo-workflows/v3/server/workflow.(*workflowServer).LintWorkflow`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=43808
Fix	https://github.com/argoproj/argo-workflows/pull/7769
ID	ADA-Argo-22-07

A nil-dereference would happen if the `req` parameter to `github.com/argoproj/argo-workflows/v3/server/workflow.(*workflowServer).LintWorkflow` was `nil`. `LintWorkflow` would pass `nil` onto `s.instanceIDService.Label(req.Workflow)` which would pass nil onto `github.com/argoproj/argo-workflows/v3/util/labels.Label` which would call `obj.GetLabels()` where a nil-dereference would happen.

The fix was to make an early nil-check in LintWorkflow:

```
func (s *workflowServer) LintWorkflow(ctx context.Context, req
*workflowpkg.WorkflowLintRequest) (*wfv1.Workflow, error) {
	if req.Workflow == nil {
		return nil, fmt.Errorf("unable to get a workflow")
	}
	wfClient := auth.GetWfClient(ctx)
	wftmplGetter :=
templateresolution.WrapWorkflowTemplateInterface(wfClient.ArgoprojV1alph
a1().WorkflowTemplates(req.Namespace))
	cwftmplGetter :=
templateresolution.WrapClusterWorkflowTemplateInterface(wfClient.Argopro
jV1alpha1().ClusterWorkflowTemplates())
	s.instanceIDService.Label(req.Workflow)
	creator.Label(ctx, req.Workflow)
```

Issue 8

Type	Index out of range
Source	`github.com/argoproj/argo-workflows/v3/server/artifacts.(*ArtifactServer).GetInputArtifactByUID.`
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=44174
Fix	https://github.com/argoproj/argo-workflows/pull/7648
ID	ADA-Argo-22-08

The http.Request parameter to (*ArtifactServer)GetInputArtifactByUID is getting split by “/”. The resulting slice was being looked up by index without sufficient range checks:

```
func (a *ArtifactServer) GetInputArtifactByUID(w http.ResponseWriter, r
*http.Request) {
	a.getArtifactByUID(w, r, true)
}

func (a *ArtifactServer) getArtifactByUID(w http.ResponseWriter, r
*http.Request, isInput bool) {
	requestPath := strings.SplitN(r.URL.Path, "/", 5)

	uid := requestPath[2]
	nodeId := requestPath[3]
	artifactName := requestPath[4]
```

The fix was a simple check to ensure the slice has a length of 5:

```
func (a *ArtifactServer) GetInputArtifactByUID(w http.ResponseWriter, r
*http.Request) {
	a.getArtifactByUID(w, r, true)
}

func (a *ArtifactServer) getArtifactByUID(w http.ResponseWriter, r
*http.Request, isInput bool) {
	requestPath := strings.SplitN(r.URL.Path, "/", 5)
	if len(requestPath) != 5 {
		a.serverInternalError(errors.New("request path is not
valid"), w)
		return
	}
	uid := requestPath[2]
	nodeId := requestPath[3]
```

```
    artifactName := requestPath[4]
```

Issue 9

Type	Out of memory
Source	N/a
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=44912
Fix	N/a
ID	ADA-Argo-22-09

This issue was found in the final moments of the engagement, and a fix has not yet been committed.

Issue 10

Type	Index out of range
Source	N/a
Issue link	https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=45026
Fix	N/a
ID	ADA-Argo-22-10

This issue was found in the final moments of the engagement, and a fix has not yet been committed.

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 Argo 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.

Long-term advice

1. Assess which parts of the argo 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 fixed in line with how the bugs in this engagement were fixed.
2. When new code is submitted to Argo 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 Ada Logics integrated fuzzing into the Argo Project. The focus was on five repositories: Argo Workflows, Argo CD, Argo Events, Argo Rollouts and GitOps Engine. Ada Logics created a total of 41 fuzzers and 10 bugs were found. All fuzzers were set up to run continuously by way of OSS-fuzz which will allow them to continue to look for bugs. At the end of the engagement, all bugs have been fixed.