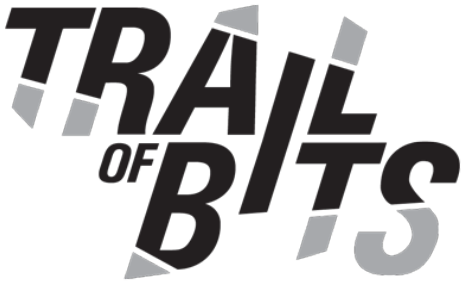


Argo

Threat Model

February 25, 2021

Prepared For:
Edward Lee | *Intuit*
edward_lee@intuit.com

Chris Aniszczyk | *Linux Foundation*
caniszczyk@linuxfoundation.org

Alex Collins | *Intuit*
alex_collins@intuit.com

Jesse Suen | *Intuit*
jesse_suen@intuit.com

Alexander Matyushentsev | *Intuit*
alexander_matyushentsev@intuit.com

Henrik Blixt | *Intuit*
henrik_blixt@intuit.com

Prepared By:
David Pokora | *Trail of Bits*
david.pokora@trailofbits.com

Brian Glas | *Trail of Bits*
brian.glas@trailofbits.com

Alex Useche | *Trail of Bits*
alex.useche@trailofbits.com

Mike Martel | *Trail of Bits*
mike.martel@trailofbits.com

Changelog:
March 19, 2021: Revised report delivered

Introduction

The Cloud Native Computing Foundation (CNCF) tasked Trail of Bits with conducting a component-focused threat model of the Argo CD, Argo Workflows, Argo Rollouts, and Argo Events systems. This threat model reviewed Argo components across six control families:

- Networking
- Cryptography
- Authentication
- Authorization
- Secrets Management
- Multi-tenancy

The Argo project consisted of four discrete services: Argo CD, Argo Workflows, Argo Rollouts, and Argo Events. While there may be optional connections between some components (e.g., Argo Events and Argo Workflows), each service operates independently and was therefore modeled and evaluated separately within the report. All four services rely on Kubernetes for their core functionality. Argo CD and Argo Workflows consist of a series of Kubernetes pods, services, and deployments that support Continuous Deployment pipelines and job orchestration workflows, respectively. Argo Rollouts extends Kubernetes deployment capabilities through a Kubernetes controller and a series of CRDs to provide functionality such as canary deployments. Argo Events leverages a series of Kubernetes resources to provide event-driven dependency management capabilities.

Rather than providing the capabilities of each project as a service, the evaluated components are deployed by their users (typically Kubernetes operators) in their own infrastructure. As such, our evaluation considered how the design of each component could introduce threats to the confidentiality, integrity, and availability of Argo customers' data, infrastructure, and reputation. We also considered the direct risk to Argo's reputation that could result from attacks on clients' Argo infrastructure.

The above control families were chosen due to system sensitivities in those areas. Trail of Bits also considered general confidentiality, integrity, and availability concerns.

Trail of Bits' assessment of the four Argo services resulted in 22 findings ranging from informational to high severity.

Key Findings

Throughout the review, Trail of Bits focused on revealing concerns in the abovementioned control families while considering confidentiality, integrity, and availability.

Generally, Trail of Bits found that while Argo services often support various risk or threat prevention methods, these methods are frequently under-documented or are provided on an opt-in basis, rather than by default.

Examples of this in Argo CD include an insufficient guide regarding secrets management (TOB-ARGO-TM03), disabled (by default) metrics/logging for some components (TOB-ARGO-TM05), and insufficient documentation regarding best practices for logging desirable information such as IP addresses in the event of an attack (TM-ARGO-TM07). Similarly, in Argo Workflows, we noted a lack of authentication by default (TM-ARGO-TM11) as well as TLS disabled by default (TM-ARGO-TM15) and similar concerns regarding IP logging documentation (TM-ARGO-TM17).

It would be beneficial to implement component isolation as an added layer of security where possible. During the remote threat modeling sessions, Trail of Bits noted that event sources and sensors in Argo Events isolate their secrets and network traffic from each other; however, Argo CD and Workflows do not enforce any ingress/egress rules for components throughout the system (TOB-ARGO-TM02, TM-ARGO-TM10). We would suggest reorienting how trust zones exist within the system.

Report Position

The Argo services provide a wide range of capabilities for end users, with various external connections and integrations. We conducted a best-effort review of each service within the given time constraints and also attempted to review all previously mentioned security controls while considering general confidentiality, availability, and integrity concerns. This report catalogs many of the key takeaways from the Rapid Risk Assessment process.

This report analyzes components, trust zones, data flows, threat actors, controls, and findings of the Argo threat model. It represents a fixed point-in-time assessment that may not account for future iterations or changes implemented during the engagement. For example, the `eventbus-default-metrics` component was removed from Argo Events in between drafts of this report but is still referenced in this report.

Trail of Bits operated under the assumption that information shared during the daily remote meetings held between February 16 and February 19, 2021 accurately described

the current state of the infrastructure. Additionally, documentation from that period was leveraged from the following pages:

- https://argo-cd.readthedocs.io/en/stable/
- https://argoproj.github.io/argo-workflows/
- https://argoproj.github.io/argo-rollouts/
- https://argoproj.github.io/argo-events/

Introduction **1**
Key Findings 1
Report Position 2
Methodology **6**
Components **7**
Argo CD 7
Argo Workflows 8
Argo Rollouts 9
Argo Events 9
Trust Zones **11**
Argo CD 11
Argo Workflows 11
Argo Rollouts 12
Argo Events 12
Trust Zone Connections **14**
Argo CD 14
Argo Workflows 15
Argo Rollouts 16
Argo Events 16
Threat Actors **18**
Argo CD 19
Argo Workflows 22
Argo Rollouts 24
Argo Events 25
Dataflow **27**
Argo CD 27
Argo Workflows 28
Argo Rollouts 29
Argo Events 30
Findings Summary **31**
Argo CD Findings **33**
1. Redis communications are unencrypted 34
2. Insufficient default network access controls between pods 35
3. Insufficient guidance on secure storage of secrets 36
4. Admin JWTs never expire 37

5. gRPC metrics are turned off by default 38
6. Manual password reset process does not enforce password complexity requirements 39
7. Documentation does not make recommendations for IP address logging, will not occur by default 40
8. JSON Web Token-based authentication 41
9. Secrets used in one component are not isolated from others 42

Argo Workflows Findings 43
10. Insufficient default network access controls between pods 44
11. Lack of authentication rate limiting 45
12. API does not require authentication by default 46
14. User authentication is not logged if user isn't mapped to RBAC rules 47
15. TLS is not enabled by default 48
16. Weak database password for base deployments of Argo Workflows 49
17. Documentation does not make recommendations for IP address logging, will not occur by default 50
18. JSON Web Encryption-based authentication 51

Argo Events Findings 52
19. Insufficient logging of dropped events 53
20. Undocumented potential race condition in Event Bus 54
21. Event Bus contains plaintext events 55
22. Events cannot be optionally rate limited 56

Argo Rollouts Findings 57

A. RRA Template 58
Overview 58
Service Notes 58
How does the service work? 58
Are there any subcomponents or shared boundaries? 58
What communication protocols does it use? 58
Where does it store data? 58
What is the most sensitive data it stores? 58
How is that data stored? 58
Data Dictionary 58
Control Families 58
Threat Scenarios 59
Networking 60
Cryptography 60
Secrets Management 60

Authentication 60
Authorization 60
Multi-tenancy Isolation 60
Summary 60
Recommendations **60**

Methodology

This document contains the results of four engineer-days of effort from members of the Intuit team and the assessment team. Each day focused on one of the following Argo services: Argo CD, Argo Rollouts, Argo Workflows, and Argo Events. The review drew from supporting documentation such as existing architectural diagrams, end user documentation, and blog posts written by the Intuit team. We provide a control-focused threat model, reviewing each component based on the controls determined by Trail of Bits and Intuit. A rudimentary Rapid Risk Assessment (RRA) template that reflects our processes is included in Appendix A: RRA Template.

Performing RRAs against four Argo services in a four–calendar-day work week proved to be challenging. We began with a modified version of Mozilla's RRA, including security controls we believed would effectively expose any risks in these services, given the time constraints. After agreeing upon methodology with the Intuit team, we scheduled one remote meeting for each Argo service to discuss relevant components and architectural design. During this time, we drove inquiries using agreed-upon security controls for each component, with consideration for additional security controls that could affect confidentiality, integrity, and availability. Leveraging supporting documentation and diagrams as well as relevant code repositories to address any outstanding concerns, we produced the following findings and diagrams detailing the identified risks. Code reviews were not part of this assessment, as our threat model was focused on a security control-driven evaluation of the design of all four services. However, we deployed the infrastructure for Argo CD, Argo Workflows, and Argo Events in an effort to better understand each service's components. Deploying each service locally also allowed us to confirm that our models were correct.

Components

The Argo services comprise the Argo CD, Argo Workflows, Argo Rollouts, and Argo Events projects. Each project consists of multiple components, many of which are standalone binaries written in Go running as Kubernetes services, deployments, and CRDs. The following table describes each project and its underlying components.

Argo CD

Component	Description
argocd-server	A pod that hosts the Web UI front end and gRPC API back end.
API Server (argocd-server)	A server that resides in the argocd-server pod and is responsible for authenticating operators upon initial setup of Argo CD, forwarding of authentication, enforcing RBAC rules, managing credentials stored as Kubernetes secrets, listening for and forwarding webhook events, reporting application status, and managing applications.
Web UI (argocd-server)	A component that resides in the argocd-server. It provides a web interface for interacting with the API server.
Command Line Interface (CLI)	An Argo utility that allows users to interact with the API from their local machines without using the Web UI.
argocd-dex-server	A pod running a modified version of the Dex identity service. Dex is a third-party library that allows multiplexing OICD authentication. It is responsible for translating tokens from services like OAuth into OICD tokens that Argo can understand and work with.
argocd-application-controller	A Kubernetes controller responsible for monitoring the state of CD pipelines. If configured to do so, the application controller will take corrective action whenever an application becomes out of sync with a given repository.
argocd-repo-server	A repository server running a service. The repo server maintains a local cache of repositories set up for CD pipelines and generates the Kubernetes manifest needed to spin up the application configured for CD. Additionally, it triggers webhooks for events such as pre-sync, post-sync, and sync.
argocd-redis-ha	A high-availability server running Redis. Redis is used by the argocd-repo-server, caching repositories and maintaining a

	cache of the hierarchy of Kubernetes resources. The argocd-application-controller places data in Redis, after which the data becomes available to the API.
argocd-server-metrics	A service that collects metrics from the API server.
argocd-metrics	A service that collects metrics from the application controller.
Version Control System (VCS)	The external VCS from which Argo CD pulls manifests when changes such as pull requests are made. It can be a GitHub, GitLab, or other repository.
Target Cluster	The cluster(s) to which Argo CD deploys infrastructural changes as a result of manifest changes.
Resource Hook Target	A Kubernetes resource invoked as a hook during processes such as sync operations.

Argo Workflows

Component	Description
argo-server	A server running in a pod that hosts an API and Web UI.
API Server (argo-server)	A server that provides an interface for creating and managing workflows. The workflow API provides authentication through bearer tokens and exposes endpoints for managing workflows, workflow templates, and cron jobs that kick-start workflows.
Web UI (argo-server)	A component that resides in the argo-server and provides a web interface for interacting with the API server.
workflow-controller	A controller responsible for creating workflows in a workflow-specific namespace. The controller also watches the namespace for changes and acts upon those changes. While the API creates workflows, the workflow-controller updates workflows.
workflow-controller-metrics	A service that communicates with the workflow-controller to obtain metrics data.
User Workflow	A workflow created in its own namespace separate from the Argo namespace where all other services run.

SQL Database (MySQL or PostgreSQL)	A database that holds workflow data, or structures. After a workflow has been completed, a copy is stored in SQL for retrieval by the API.
Artifact Storage	Storage for the artifacts resulting from workflows. There are numerous storage options, such asAmazon S3 and MinIO.
Command Line Interface (CLI)	An Argo utility that allows users to interact with the API.

Argo Rollouts

Component	Description
argo-rollouts	The single component deployed by Argo Rollouts. A controller that speaks to Kubernetes and monitors rollout objects that the user deploys. It is responsible for modifying Istio virtual services, executing NGINX ingress services, annotating ALB ingresses, and otherwise creating traffic splits to the appropriate canary and stable replica sets.
Service Mesh Provider	The existing service mesh resource (Istio, NGINX ingress, ALB, etc.) that Argo Rollouts will manipulate to match the intent of the rollout.
Logging Provider	argo-rollouts can optionally connect many different logging services to provide metrics for rollouts. These include DataDog, NewRelic, and Wavefront.

Argo Events

Component	Description
eventsource-controller	A controller that creates relevant event source pods to accept events from a variety of potential origins. The eventsource-controller creates a Kubernetes deployment for each event source.

Event Source (Pod)	A resource created by the eventsource-controller. It watches for events from sources defined by user-provided specs, which can include webhooks and GitHub pushes. Depending on the type of event source, it may wrap messages with structs containing metadata about the message.
Event Origin	An external component that emits events for which event source pods listen. It can take various forms, including Amazon SQS, Amazon SNS, and proprietary HTTP endpoints.
eventbus-controller	A controller that watches event bus objects, creates relevant worker pods that leverage the NATS streaming protocol, etc.
eventbus	Receives and queues messages from event sources for later retrieval by triggers.
eventbus-default-metrics	Provides metrics for the service bus.
sensor-controller	A controller that acts as an event dependency manager, creating sensor resources to handle different event sources and triggers as necessary.
Sensor (Pod)	A resource created by the sensor-controller as needed. It uses a list of event dependencies (names of event sources) to fetch events from the event bus. Events are deleted from the bus upon retrieval. After retrieving a message, the controller reads the struct data and looks for data that sensors may be interested in.
Trigger Destination	The external destination component that will be triggered by Argo Events as a result of a sensor's receiving a relevant request. It could be a webhook, AWS lambda, Argo Workflows, or another component.

Trust Zones

Systems include logical “trust boundaries” or “zones” in which components may have different criticality or sensitivity. Therefore, to further analyze a system, we decompose components into zones based on shared criticality, rather than physical placement in the system. Trust zones capture logical boundaries where controls should or could be enforced by the system and allow designers to implement interstitial controls and policies between zones of components as needed.

Argo CD

Zone	Description	Included Components
Internet	The wider external-facing internet zone. It typically includes users and software that interface with the service but does not contain core application logic.	• VCS • CLI • Web UI (argocd-server) • Resource Hook Target
Argo Infrastructure	The infrastructure powered by the Kubernetes cluster that serves Argo CD services to the user and carries out deployments.	• API Server (argocd-server) • argocd-application-controller • argocd-repo-server • argocd-dex-server • argocd-redis-ha • argocd-metrics • argocd-server-metrics
Deployment Target Infrastructure	The target infrastructure that hosts components defined by manifests absorbed by Argo CD.	• Target Cluster

Argo Workflows

Zone	Description	Included Components
Internet	The wider external-facing internet zone. It typically includes users and software that interfaces with the service but does not contain core	• CLI • Web UI (argo-server) • Logging Provider

	application logic.	
Argo Infrastructure	The infrastructure powered by the Kubernetes cluster that oversees the workflow orchestration.	• API Server (argo-server) • workflow-controller • workflow-controller-metrics
Workflow Infrastructure	Any infrastructure created as a result of a user workflow submission.	• User Workflow
Workflow Storage	Any file storage, repositories, databases, or VCSes that house user workflow artifacts.	• Artifact Storage • SQL Database

Argo Rollouts

Zone	Description	Included Components
Internet	The wider external-facing internet zone. It typically includes users and software that interface with the service but does not contain core application logic.	• Logging Provider
Argo Infrastructure	The infrastructure powered by the Kubernetes cluster that oversees Argo Rollouts-type deployments for the user.	• argo-rollouts
Service Mesh	The service mesh boundaries highlight user-supplied service meshes that will be modified to direct traffic for the desired rollout strategy.	• Service Mesh Provider (Istio, NGINX ingress, ALB, etc).

Argo Events

Zone	Description	Included Components
Internet	The wider external-facing internet zone. It excludes components that	• Event Origin • Trigger Destination

	better fit into other trust zones, often due to notions of ownership.	• Logging Provider
Argo Infrastructure	The infrastructure powered by the Kubernetes cluster that oversees Argo Events coordination efforts.	• eventsource-controller • Event Source (Pod) • eventbus-controller • eventbus • Sensor-controller • Sensor (Pod) • eventbus-default-metrics

Trust Zone Connections

Trust zones are useful to enumerate attack scenarios when we understand what data flows between zones and why.

Argo CD

Originating Zone	Destination Zone	Data Description	Connection Type	Authentication Type
Internet	Argo Infrastructure	The Web UI and CLI tools that live on the client's browser or machine speak to the API server.	gRPC	Pre-auth: Bearer Token Post-auth: JWT
Internet	Argo Infrastructure	The VCS used to host a given repository can optionally leverage webhooks to call the API server and force a sync immediately.	Webhook	TLS Certificate Verification + Secret
Argo Infrastructure	Internet	The Web UI is a client-side application served to the user by argocd-server.	HTTPS	TLS
Argo Infrastructure	Internet	argocd-repo-server polls VCS repositories for changes every three minutes.	HTTPS/SSH	TLS Certificate / SSH Key Pinning
Argo Infrastructure	Internet	Resource hooks can be configured to reach out to arbitrary external components upon execution of various operations, such as sync.	Arbitrary	Arbitrary
Argo Infrastructure	Deployment Target Infrastructure	The argocd-application-controller performs a sync and ensures that the deployment	Kubernetes	Kubernetes

		target infrastructure matches the specification to which it was synced.		

Argo Workflows

Originating Zone	Destination Zone	Data Description	Connection Type	Authentication Type
Internet	Argo Infrastructure	The Web UI and CLI tools that live on the client's browser or machine speak to the API server.	gRPC	Pre-auth: Bearer Token Post-auth: JWT
Argo Infrastructure	Workflow Storage	Users may wish to retain workflows for longer periods of time. Argo optionally saves completed workflows to an SQL database. The API server retrieves workflows, while the controller saves them to the database.	PostgreSQL / MySQL	Username + Password
Argo Infrastructure	Workflow Storage	Workflows often pass data between each other as inputs and outputs. These inputs and outputs can be taken as artifacts and stored in a number of providers. These providers, such as Amazon S3, essentially serve as intermediaries. The API server retrieves artifacts, and the workflow-controller saves them to the artifact repository.	Various	Cloud Providers: Access Key + Secret Key / Git: Username + Password + SSH Key

Internet	Argo Infrastructure	Event sources can submit events via the `events` API endpoint by submitting an event template.	gRPC	Client Access Tokens

Argo Rollouts

Originating Zone	Destination Zone	Data Description	Connection Type	Authentication Type
Argo Infrastructure	Service Mesh Provider	The argo-rollouts controller modifies existing service mesh providers, NGINX ingress services, or ALB ingresses for purposes such as enabling canary deployments. It splits traffic accordingly.	Various	Various
Argo Infrastructure	Internet	Argo infrastructure can push logs to a variety of logging providers to provide metrics for rollouts. These include DataDog, NewRelic, and Wavefront.	Various	Various

Argo Events

Originating Zone	Destination Zone	Data Description	Connection Type	Authentication Type
Internet	Argo Infrastructure	An event origin can push events to an event source pod for processing by Argo Events. This is done	HTTPS	Authentication Token

		via webhooks.		
Argo Infrastructure	Internet	An event source pod can pull events from various types of event origins.	Various	Various
Argo Infrastructure	Internet	A sensor pod will attempt to deliver an event it pulls from the event bus to the trigger destination. Depending on the trigger destination type, the method for pushing this data may vary.	Various	Various
Argo Infrastructure	Internet	External logging providers are sent data from Argo infrastructure.	Various	Various

Threat Actors

Similarly to establishing trust zones, defining malicious actors before conducting a threat model is useful in determining which protections, if any, are necessary to mitigate or remediate a vulnerability. We will use these actors in all subsequent findings from the threat model. Additionally, we define other "users" of the system who may be impacted by, or induced to undertake, an attack. For example, in a confused deputy attack such as cross-site request forgery, a normal user would be both the victim and the potential direct attacker, even though that user would be induced to undertake the action by a secondary attacker.

Actor	Description
Malicious Internal User	A user such as an administrator or developer who uses stolen credentials or a privileged position maliciously against the system.
Internal Attacker	An attacker who has transited one or more trust boundaries, such as an attacker with container access.
External Attacker	An attacker who is external to the cluster and unauthenticated—for example, a developer who did not contribute to Argo products directly but worked on dependencies included in Argo services or end user repositories.
Administrator	An actual administrator of the Argo service tasked with operating and maintaining the internal cluster as a whole.
End User	The end user of the Argo service.

Additionally, defining attackers' paths through the various zones is useful when analyzing potential controls, remediations, and mitigations that exist in the current architecture.

Argo CD

Actor	Originating Zone	Destination Zone(s)	Description
Malicious Internal User	Any	Any	Malicious internal users are often privileged and have access to a wide range of resources, such as the Kubernetes cluster that hosts Argo Infrastructure. Therefore, for strong non-repudiation of actions, controls must be in place to ensure that users are authorized to undertake an action, as well as to log all actions within the system.
Administrator	Any	Any	Administrators may misconfigure Argo services such that they do not allow users to deploy their target infrastructure as expected. To mitigate such concerns, errors in the system should be logged accordingly to provide an audit trail alongside extensive administrator documentation.
End User	Any	Any	End users may misconfigure their repositories in a way that prevents their target infrastructure from deploying as expected. To mitigate such concerns, errors in the system should be logged accordingly to provide an audit trail alongside extensive user documentation.
Internal Attacker	Argo Infrastructure	Internet	An internal attacker could modify the Web UI code served to the end user by argocd-server and force the end user's browser to execute a malicious payload.

Internal Attacker	Argo Infrastructure	Argo Infrastructure	An internal attacker could leverage access to a component in Argo infrastructure to move laterally and gain access to other components. Such an attacker may also be able to extract Kubernetes secrets meant for other components due to a lack of isolation.
Internal Attacker	Internet	Argo Infrastructure	An internal attacker could gain access to the VCS and change underlying infrastructure managed by Argo CD via pull requests/commits that modify relevant manifests.
Internal Attacker	Argo Infrastructure	Internet	An internal attacker could leverage access to Argo infrastructure to clone repositories that would otherwise be inaccessible to the attacker. The attacker may be able to extract Kubernetes secrets such as repository SSH keys, allowing the attacker greater access to underlying repositories.
Internal Attacker	Argo Infrastructure	Target Infrastructure	An internal attacker could leverage access to the Argo infrastructure to control target infrastructure deployments; the attacker could also potentially change policies for users or allow himself or herself to override syncs from their local machine instead of the VCS.
External Attacker	Internet	Argo Infrastructure	An external attacker may try to leverage API calls to look for vulnerabilities within the system. Appropriate auditing and logging procedures should be implemented to ensure that administrators can follow an audit trail to understand and mitigate attempted attacks.
External Attacker	Internet	Argo Infrastructure / Target Infrastructure	An external attacker may be a developer of dependencies used by Argo CD or user repositories. The attacker may execute a supply chain

			attack that gains control of Argo infrastructure or may target the end user's target infrastructure.

Argo Workflows

Actor	Originating Zone	Destination Zone(s)	Description
Malicious Internal User	Any	Any	Malicious internal users are often privileged and have access to a wide range of resources, such as the Kubernetes cluster that hosts Argo Infrastructure. Therefore, for strong non-repudiation of actions, controls must be in place to ensure that users are authorized to undertake an action, as well as to log all actions within the system.
Administrator	Any	Any	Administrators may misconfigure Argo services such that they do not allow users to deploy their workflow infrastructure as expected. To mitigate such concerns, errors in the system should be logged accordingly to provide an audit trail alongside extensive administrator documentation.
End User	Any	Any	End users may misconfigure their repositories in a way that prevents their workflow infrastructure from deploying as expected. To mitigate such concerns, errors in the system should be logged accordingly to provide an audit trail alongside extensive user documentation.
Internal Attacker	Argo Infrastructure	Argo Infrastructure	An internal attacker could leverage access to a component in Argo infrastructure to move laterally and gain access to other components. Such an attacker may also be able to extract Kubernetes secrets meant for

			other components due to a lack of isolation.
Internal Attacker	Argo Infrastructure	Internet	An internal attacker could extract Kubernetes secrets to connect to external logging providers, affecting the integrity or availability of logs.
Internal Attacker	Argo Infrastructure	Workflow Infrastructure / Workflow Storage	An internal attacker could extract Kubernetes secrets or otherwise access the end user's workflow infrastructure or the workflow storage that houses workflow infrastructure artifacts.
External Attacker	Internet	Argo Infrastructure	An external attacker may try to leverage API calls to look for vulnerabilities within the system. Appropriate auditing and logging procedures should be implemented to ensure that administrators can follow an audit trail to understand and mitigate attempted attacks.
External Attacker	Internet	Argo Infrastructure / Workflow Infrastructure / Workflow Storage	An external attacker may be a developer of dependencies used by Argo Workflows or user repositories. The attacker may execute a supply chain attack that gains control of Argo infrastructure or may target the end user's workflow infrastructure.

Argo Rollouts

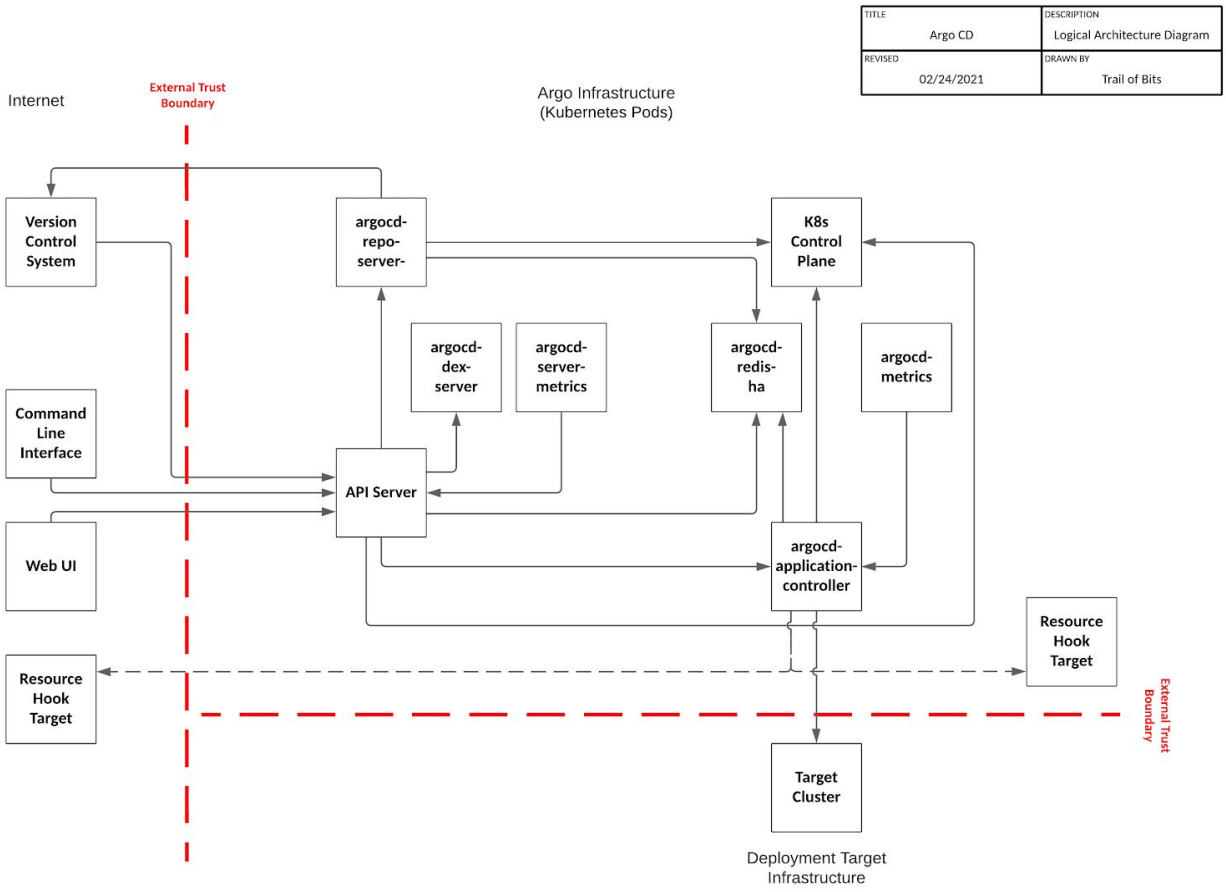
Actor	Originating Zone	Destination Zone(s)	Description
Malicious Internal User	Any	Any	Malicious internal users are often privileged and have access to a wide range of resources, such as the Kubernetes cluster that hosts Argo infrastructure. Therefore, for strong non-repudiation of actions, controls must be in place to ensure that users are authorized to undertake an action, as well as to log all actions within the system.
Administrator / End User	Argo Infrastructure	Service Mesh	End users in this system are also administrators. In this case, the Argo infrastructure modifies the underlying service mesh. End users may misconfigure their service meshes, preventing their rollouts from working as expected. To mitigate such concerns, errors in the system should be logged accordingly to provide an audit trail alongside extensive user documentation.
Administrator / End User	Argo Infrastructure	Internet	End users in this system are also administrators. In this case, the Argo infrastructure logs to a number of potential logging providers. End users may misconfigure their logging providers. End users should be provided extensive user documentation, which may help alleviate such concerns.

Argo Events

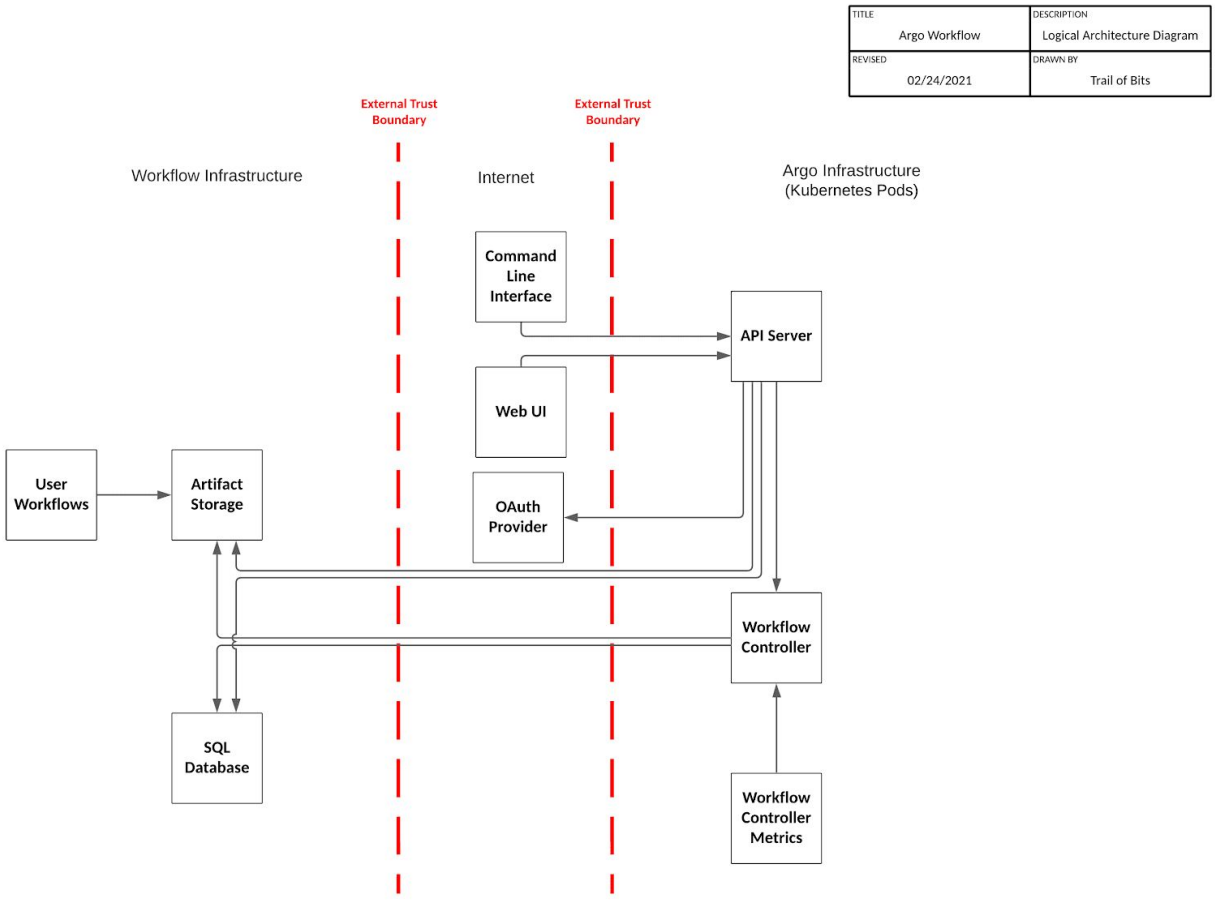
Actor	Originating Zone	Destination Zone(s)	Description
Malicious Internal User	Any	Any	Malicious internal users are often privileged and have access to a wide range of resources, such as the Kubernetes cluster that hosts Argo infrastructure. Therefore, for strong non-repudiation of actions, controls must be in place to ensure that users are authorized to undertake an action, as well as to log all actions within the system.
External Attacker	Argo Infrastructure	Internet	An external attacker may be able to attack a trigger destination to cause a message delivery failure. If the trigger destination endpoint is unavailable, the event will fail to send to the endpoint. Based on user configuration, it may retry sending a number of times, but the event may ultimately be lost permanently.
External Attacker	Internet	Argo Infrastructure	An external attacker may be able to control an event origin and affect availability, causing event source pods to fail to fetch events. Additionally, the attacker may be able to control or generate event data that can exhaust the resources of Argo Events, resulting in a denial of service.
End User	Argo Infrastructure	Argo Infrastructure / Internet	Argo infrastructure may log to a number of external logging providers. End users may misconfigure their logging providers. End users should be provided extensive user documentation, which may help alleviate such concerns.

Dataflow

Argo CD

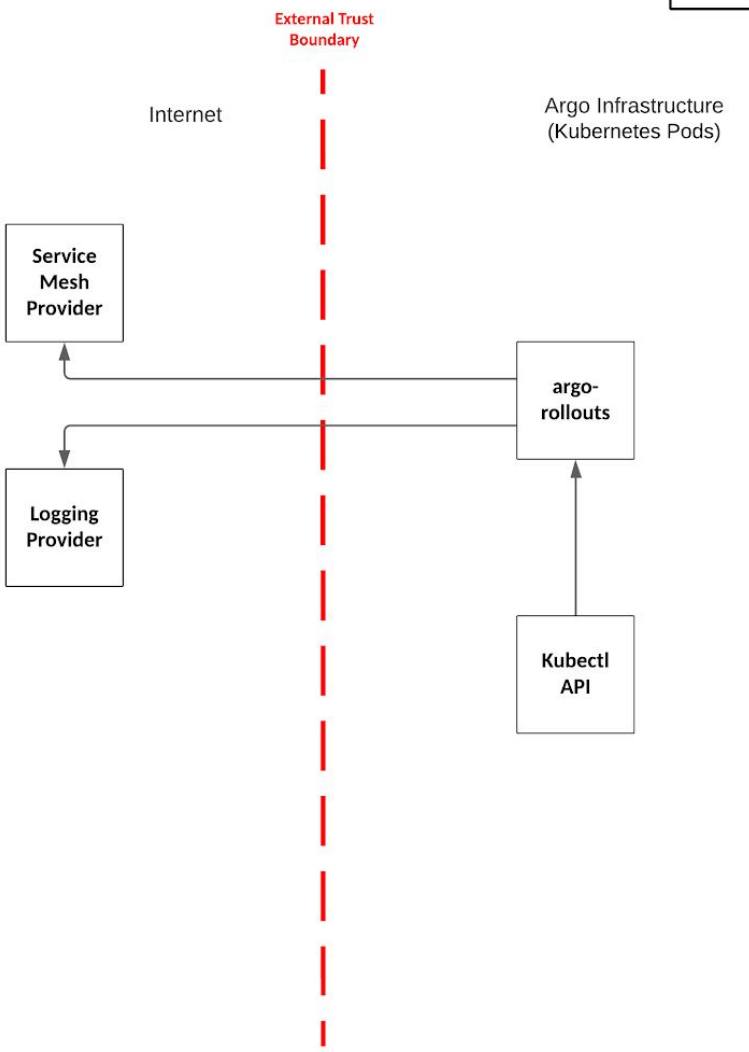


Argo Workflows

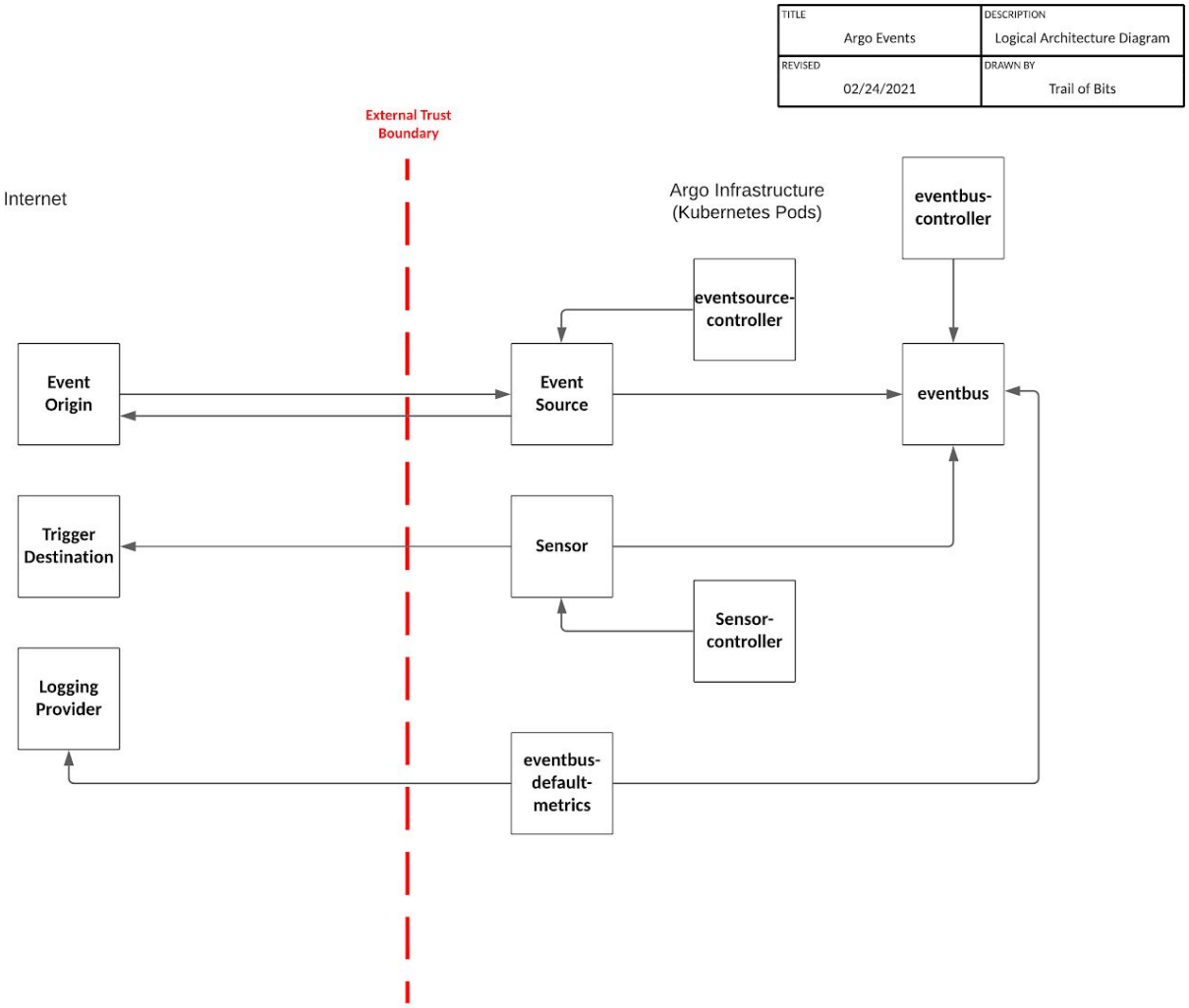


Argo Rollouts

TITLE	DESCRIPTION
Argo Rollouts	Logical Architecture Diagram
REVISED	DRAWN BY
02/24/2021	Trail of Bits



Argo Events


TITLE
Argo Events
DESCRIPTION
Logical Architecture Diagram
REVISED
02/24/2021
DRAWN BY
Trail of Bits
External Trust Boundary
Internet
Argo Infrastructure (Kubernetes Pods)
eventbus-controller
eventsource-controller
Event Origin
Event Source
eventbus
Trigger Destination
Sensor
Sensor-controller
Logging Provider
eventbus-default-metrics

Findings Summary

#	Title	Type	Severity
1	Redis communications are unencrypted	Cryptography	Medium
2	Insufficient default network access controls between pods	Networking	Medium
3	Insufficient guidance on secure storage of secrets	Secrets Management	Informational
4	Admin JWTs never expire	Authentication	Medium
5	gRPC metrics are turned off by default	Authentication	Low
6	Manual password reset process does not enforce password complexity requirements	Authentication	Medium
7	Documentation does not make recommendations for IP address logging, will not occur by default	Networking	Low
8	JSON Web Token-based authentication	Authentication	Low
9	Secrets used in one component are not isolated from others	Secrets Management	Undetermined
10	Insufficient default network access controls between pods	Networking	High
11	Lack of authentication rate limiting	Authentication	Medium
12	API does not require authentication by default	Authentication	Medium
14	User authentication is not logged if user isn't mapped to RBAC rules	Authentication	Low

15	TLS is not enabled by default	Authentication	Medium
16	Weak database password for base deployments of Argo Workflows	Authentication	Medium
17	Documentation does not make recommendations for IP address logging, will not occur by default	Networking	Low
18	JSON Web Encryption-based authentication	Authentication	Low
19	Insufficient logging of dropped events	Networking	Low
20	Undocumented potential race condition in Event Bus	Networking	Medium
21	Event Bus contains plaintext events	Cryptography	Low
22	Events cannot be optionally rate limited	Networking	Medium

Argo CD Findings

Argo CD is the most complex of all Argo services. It is a Kubernetes-based GitOps delivery tool that allows developers to create continuous delivery pipelines for deploying applications in their own clusters.

Argo CD deployments consist of docker images and installation files orchestrated with Kubernetes. It relies on a REST API that provides authentication, authorization, and deployment management capabilities. Operators authenticate using the API and can set up single sign-on (SSO) authentication, which the API handles by forwarding authentication requests to the argocd-dex-server. Users interact with the API via Web UI or the `argo` CLI utility. A repository server monitors the state of VCS repositories and synchronizes CD tasks through an application controller. Repositories are cached using high-availability Redis servers. Webhooks also communicate with the repository server to trigger deployment synchronization tasks.

1. Redis communications are unencrypted

Severity: Medium
Type: Cryptography
Difficulty: High
Finding ID: TOB-ARGO-TM01

Description
Argo CD leverages a Redis instance when performing operations in the API server, `argocd-application-controller`, and `argocd-repo-server`. However, connections to this Redis instance are not secured with a password via a `redis.conf` configuration file.

If a user were able to speak to the Redis instance, the lack of password authentication would allow the user access to data cached by the abovementioned Argo architectural components.

Justification
The difficulty is high for the following reasons:

- An attacker must transit trust boundaries to be able to speak to the Redis instance.
- Alternatively, an end user would need to publicly expose the Redis instance.

The severity is medium for the following reason:

- The Redis instance caches artifacts from the API server, `argocd-application-controller`, and `argocd-repo-server`. These may contain sensitive information about some repositories.

Recommendation
Short term, consider creating a Redis configuration with a password such that an internal attacker or publicly exposed Redis instance does not lead to compromise of sensitive cached data.

Long term, review all Argo infrastructure components to ensure that connections between components are secured using appropriate cryptographic and authentication standards.

2. Insufficient default network access controls between pods

Severity: High
Type: Networking
Difficulty: High
Finding ID: TOB-ARGO-TM02

Description
The namespace `argocd`, where pods and services reside, does not enforce any ingress or egress restrictions. As a result, all pods can communicate with each other, which is unnecessary in many cases.

For example, `argocd-repo-server` does not need to communicate with `argocd-server`, yet communication between the two components is not restricted. This could facilitate lateral movement should an attacker gain access to a pod within this namespace.

Justification
The difficulty is high for the following reason:

- An attacker must first transit trust boundaries into Argo Infrastructure to be able to speak to other non–external-facing components.

The severity is high for the following reason:

- This allows lateral movement, which may enable an attacker to further compromise the system and underlying repository credentials.

Recommendation
Short term, consider enforcing ingress and egress restrictions to allow only resources that need to speak to each other to do so. Leverage allow-lists instead of deny-lists to ensure that only expected components can establish these connections.

Long term, review exposure of sensitive secrets and services within the network. Ensure the use of appropriate methods of isolation to prevent lateral movement.

3. Insufficient guidance on secure storage of secrets

Severity: Informational
Type: Secrets Management
Difficulty: Low
Finding ID: TOB-ARGO-TM03

Description
Currently, Argo CD documentation provides some documentation regarding how to use secret management for private repository credentials. However, this documentation is quite vague and may lead users to mistakenly store sensitive data in plaintext.

Similarly, each link in other secret management guides refer the user to the respective secret management vendor without providing guidance on how or why to use those systems for secrets such as private repository credentials.

Justification
The difficulty is low for the following reason:

- It is not difficult for users to misunderstand the motivations and use cases of the suggestions in these guidelines.

The severity is informational for the following reasons:

- The documentation refers the user to the correct secret management vendor, which will likely provide similar documentation.
- However, more context would help prevent user error.

Recommendation
Short term, consider iterating on guides related to secret management.

Long term, consider reviewing the documentation. Look for any deficiencies in explanations of the purpose or methodology behind the suggestions.

4. Admin JWTs never expire

Severity: Medium
Type: Authentication
Difficulty: Medium
Finding ID: TOB-ARGO-TM04

Description
Argo CD leverages JSON Web Tokens (JWTs) for authentication purposes. JWTs typically expire within 24 hours for SSO users. However, for administrators, JWTs never expire.

This could pose a problem if an attacker who compromises an administrator's JWT leverages it for persistent access within the system.

Justification
The difficulty is high for the following reason:

- It may be difficult to capture an administrator's JWT, as administrators are discouraged from using these accounts for anything but initial setup.

The severity is medium for the following reasons:

- A compromised administrator JWT may be used to gain persistent access in the system.
- An operator may notice signs of an intrusion and invalidate all tokens.
- However, the attacker may perform destructive actions before the operator notices the intrusion. Alternatively, the attacker may avoid performing actions that raise suspicion.

Recommendation
Short term, consider setting an expiration date for administrator JWTs to ensure that an attacker does not gain persistent access if an administrator JWT is leaked.

5. gRPC metrics are turned off by default

Severity: Low
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM05

Description
Argo CD's gRPC metrics are turned off by default for its API server. This means that an audit trail for the API server may not be available to operators in the event of an attack.

An upgrade guide indicates that the metrics were disabled because they were too expensive. However, it is likely that logging can be fine-tuned, a different provider can be used, or users can be led to opt out. It is often best to balance with secure-by-default.

Remember that an operator could naively accept the default configuration and be left vulnerable.

Justification
The difficulty is low for the following reason:

- It would not be difficult for an external attacker to target the API server in a way that could have been identified and prevented by the use of gRPC metrics.

The severity is low for the following reasons:

- The lack of an audit trail may hinder an operator's ability to understand behavior within a system and reduce future risk.
- However, a lack of logging does not directly compromise the system in a severe way.

Recommendation
Short term, consider logging API operations by default to ensure that users will have an audit trail unless they opt out.

Long term, review all areas within the system where critical operations are performed to ensure that there would be an appropriate audit trail in the event of an attack or system deficiency.

6. Manual password reset process does not enforce password complexity requirements

Severity: Medium
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM06

Description
Argo CD requires that username/password authentication, such as that for the default administrator account, be set via kubectl. This account can then be logged in to with the CLI or Web UI. However, due to password reset conventions, there are no password complexity requirements.

If a user selects a password that is insufficient by today's standards, an attacker may be able to gain access to the user's account. Implementation of first-party password setting/resetting methodology would easily prevent this scenario.

Justification
The difficulty is low for the following reason:

- It would not be difficult for an attacker to discern a weak password, especially a single-character password.

The severity is medium for the following reason:

- Allowing users to set insecure passwords may lead to account/infrastructure compromise.

Recommendation
Short term, consider implementing first-party password-setting functionality that enforces reasonable password complexity requirements.

7. Documentation does not make recommendations for IP address logging, will not occur by default

Severity: Low
Type: Networking
Difficulty: Low
Finding ID: TOB-ARGO-TM07

Description
While Argo CD offers logging services for various cases, it does not log the IP address of any users interacting with the API. In the event of an attack against the system, operators may be unable to discern the attacker.

During the remote threat modeling sessions, it was noted that production infrastructure should have a load balancer placed in front of Argo that logs IP addresses for audit trail purposes. However, the documentation does not appear to make recommendations regarding this issue.

Justification
The difficulty is low for the following reason:

- It would not be difficult for an attacker to evade identification by operators, as operators may be unable to capture the attacker's IP address, depending on the Argo CD configuration.

The severity is low for the following reasons:

- The lack of an audit trail may hinder an operator's ability to understand behavior within a system and reduce future risk.
- However, the lack of logging does not directly compromise the system in a severe way.

Recommendation
Short term, consider documenting best practices for ensuring that users can log the IP addresses that access their API server instance.

Long term, review all areas within the system where critical operations are performed to ensure that there would be an appropriate audit trail in the event of an attack or system deficiency.

8. JSON Web Token-based authentication

Severity: Low
Type: Authentication
Difficulty: Medium
Finding ID: TOB-ARGO-TM08

Description
Argo CD relies on JWTs for authentication. While using a JWT is a convenient method of providing authentication to multiple clients that share access to a service, JWTs cannot be invalidated upon logout or when new sessions are created before they expire.

Justification
The difficulty is medium for the following reasons:

- Attackers would need to obtain valid JWTs through other vulnerabilities.
- Users cannot take proactive steps to invalidate tokens.

The severity is medium for the following reason:

- Attackers who obtain valid JWTs may be able to perform authenticated tasks via the API until the tokens expire.

Recommendation
Short term, given that JWTs are necessary in this type of architecture and intended usage, consider mitigating the associated risks by setting sensible expiration dates for tokens and issuing refresh tokens so that users can maintain a valid session through the Web UI. Additionally, leverage cookies to store JWTs to the server and avoid using LocalStorage to store JWTs. (As of the time threat modelling investigations were conducted, JWTs were stored in cookies).

9. Secrets used in one component are not isolated from others

Severity: Undetermined
Type: Secrets Management
Difficulty: Medium
Finding ID: TOB-ARGO-TM09

Description
Remote threat modeling sessions indicated Argo CD utilizes secrets throughout the system to store sensitive data. However, these secrets are not isolated from other components in the same cluster/namespace.

As a result, a compromised component such as Redis may be able to acquire Kubernetes secrets containing sensitive user data.

Justification
The difficulty is medium for the following reason:

- An attacker must first transit trust boundaries into Argo infrastructure to be able to retrieve these secrets.

The severity is undetermined for the following reason:

- Because an exhaustive list of potential secrets was not enumerated during this assessment, the severity could not be determined.

Recommendation
Short term, consider isolating sensitive user data within a trust zone that includes only components that need it.

Long term, review exposure of sensitive secrets and services within the network. Ensure the use of appropriate methods of isolation to prevent lateral movement or exposure of sensitive user data.

Argo Workflows Findings

Argo Workflows is an engine for creating and managing multi-step workflows. Workflow steps are defined in their own containers. Like Argo CD, Argo Workflows depends on an API server that users access via a Web UI or CLI utility for creating and managing workflows. The API also serves as an authentication endpoint.

Within Kubernetes, Argo Workflows is defined as a set of Kubernetes CRDs, and each workflow is a custom Kubernetes resource that resides in the user namespace. Workflows can be started via API requests, webhooks, or cron jobs. Requests are then forwarded to the workflow-controller, which monitors the workflow namespace for changes and executes tasks accordingly. After a workflow has been completed, a copy of the workflow is made and stored in an SQL database. Produced artifacts are stored in services like s3 and MinIO.

Note: Finding No. 13 was removed from this report as it was found to be a false-positive. To maintain integrity against relevant issues opened on GitHub in response to this document, we did not change finding numbers or identifiers to reflect the removal.

10. Insufficient default network access controls between pods

Severity: High
Type: Networking

Difficulty: High
Finding ID: TOB-ARGO-TM10

Description
The namespace `argo` where pods and services reside does not enforce any ingress or egress restrictions. As a result, all pods can communicate with each other, which is unnecessary in many cases.

For example, a new component added to Argo Workflows would be able to communicate with all other components within the same cluster/namespace even if communication were unnecessary. This could facilitate lateral movement should an attacker gain access to a pod within this namespace.

Justification
The difficulty is high for the following reason:

- An attacker must first transit trust boundaries into Argo infrastructure to be able to speak to other non–external-facing components.

The severity is high for the following reason:

- This allows lateral movement, which may enable an attacker to further compromise the system and underlying repository credentials.

Recommendation
Short term, consider enforcing ingress and egress restrictions to allow only resources that need to speak to each other to do so. Leverage allow-lists instead of deny-lists to ensure that only expected components can establish these connections. See the Kubernetes documentation for guidance on network policies.

Long term, review exposure of sensitive secrets and services within the network. Ensure the use of appropriate methods of isolation to prevent lateral movement.

11. Lack of authentication rate limiting

Severity: Medium
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM11

Description
Although Argo CD leverages authentication rate limiting, Argo Workflows does not.

If an external attacker attempted to gain access to the system, rate limiting would prevent an attack against the authentication system and could also prevent resource exhaustion.

Justification
The difficulty is low for the following reason:

- An external attacker could access the external-facing API server with minimal effort and begin to send authentication requests.

The severity is medium for the following reasons:

- It is unlikely that an external attacker would be able to easily gain access to an arbitrary account.
- Nonetheless, an attacker could leverage a flaw in the authentication system in conjunction with this issue to gain further access.

Recommendation
Short term, consider implementing rate-limiting mechanisms for Argo Workflows' authentication methods.

Long term, review all critical operations that authenticate against a secret or password and ensure that these operations are limited to prevent brute-force attacks.

12. API does not require authentication by default

Severity: Medium
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM12

Description
When Argo Workflows is initially deployed, the default configuration does not require any authentication.

Although Argo Workflows does provide documentation for configuring authn and authz, the default will run without authentication until it is configured by the operator. This means that if an operator naively accepts the default configuration, the operator could be left vulnerable.

Justification
The difficulty is low for the following reason:

- Accessing an external-facing API server would be trivial for an external attacker.

The severity is medium for the following reasons:

- An external attacker could gain access to the system with minimal effort.
- Many operators will intuitively understand that the configuration is insecure and will change it; however, other operators may naively accept the default configuration.

Recommendation
Short term, consider enforcing an authentication method for the API server by default. Alternatively, add a warning to the landing page informing users/operators that continuing with the configuration may leave them vulnerable.

Long term, review all external-facing components within the system to ensure that they enforce appropriate encryption and authentication standards by default.

14. User authentication is not logged if user isn't mapped to RBAC rules

Severity: Low
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM14

Description
During threat modeling sessions with the Intuit team, it was noted that when using SSO authentication, a user is mapped to RBAC rules and logging is performed.

However, when a user is not RBAC-enabled, authentication is not logged for that user. Consequently, in some configurations, authentication may not be logged, which could hinder an operator's ability to mitigate or understand certain classes of attacks.

Justification
The difficulty is high for the following reason:

- It would not be difficult to create a user configuration in which user authentication attempts are not logged.

The severity is low for the following reasons:

- The lack of an audit trail may hinder an operator's ability to understand behavior within a system and reduce future risk.
- However, a lack of logging does not compromise the system in a severe way.

Recommendation
Short term, consider logging authentication operations to ensure that they are performing as intended.

Long term, review all areas within the system where critical operations are performed to ensure that there would be an appropriate audit trail in the event of an attack or system deficiency.

15. TLS is not enabled by default

Severity: Medium
Type: Authentication
Difficulty: Low
Finding ID: TOB-ARGO-TM15

Description
Although TLS-related documentation supports TLS and recommends enabling it, it is not enabled by default within Argo Workflows as it is in Argo CD, and the initial setup guides do not encourage operators to configure it.

As such, a naive operator who does not adhere to best practices could be left vulnerable.

Justification
The difficulty is low for the following reason:

- In the event that an operator externally exposes the Argo Workflows front end, an attacker could easily man-in-the-middle traffic to exfiltrate sensitive information from users.

The severity is medium for the following reasons:

- Many operators will intuitively understand that the configuration is insecure and will change it; however, other operators may naively accept the default configuration.
- If a user does not change this configuration, sensitive information may be disclosed to any attacker performing a man-in-the-middle attack.

Recommendation
Short term, consider enforcing TLS with self-signed certificates by default. Allow users to disable them if they wish to use them in production with TLS in front of the system.

Long term, review all external-facing components within the system to ensure that they enforce appropriate encryption and authentication standards by default.

16. Weak database password for base deployments of Argo Workflows

Severity: Medium
Type: Authentication
Difficulty: Medium
Finding ID: TOB-ARGO-TM16

Description
The Argo Workflows infrastructure deployed in accordance with the Getting Started documentation sets up a PostgreSQL database that uses default credentials (postgres:password). As a result, users who want to quickly stand up Argo Workflows may introduce a database with weak credentials into their infrastructure, leaving their Argo Workflows database vulnerable to password-guessing attacks.

Justification
The difficulty is medium for the following reasons:

- The default database credentials for PostgreSQL databases are well known and easy to find online.
- However, the above mentioned risk assumes that the operator accepts the default Argo Workflows configuration that is deployed when one follows the “Getting Started” guide.
- The attacker would need to be positioned within the Argo Workflows infrastructure to reach the database.

The severity is medium for the following reasons:

- The default credentials would give an attacker admin rights to the database.
- Attackers could modify workflows, compromising the integrity of workflow data.
- Attackers could remove workflow data from the database to compromise the availability of user workflows.

Recommendation
Short term, consider automatically generating strong passwords upon deployment of the database server and storing the credentials in Kubernetes secrets. Additionally, update the “Getting Started” guide to suggest that users change the database password.

17. Documentation does not make recommendations for IP address logging, will not occur by default

Severity: Low
Type: Networking
Difficulty: Low
Finding ID: TOB-ARGO-TM17

Description
While Argo Workflows offers logging services for various cases, it does not log the IP address of any users interacting with the API. In the event of an attack, operators may be unable to discern the attacker.

During the remote threat modeling sessions, it was noted that production infrastructure should have a load balancer placed in front of Argo that logs IP addresses for audit trail purposes. However, the documentation does not appear to make recommendations regarding this issue.

Justification
The difficulty is low for the following reasons:

- It would not be difficult for an attacker to evade identification by operators, as operators may be unable to capture the attacker's IP address, depending on the Argo Workflows configuration.
- Users may incorrectly assume that IP addresses are logged; as a result, they may not be prepared for a security incident.

The severity is low for the following reasons:

- The lack of an audit trail may hinder an operator's ability to understand behavior within a system and reduce future risk.
- However, the lack of logging does not directly compromise the system in a severe way.

Recommendation
Short term, consider documenting best practices for ensuring that users can log the IP addresses that access their API server instance.

Long term, review all areas within the system where critical operations are performed to ensure that there would be an appropriate audit trail in the event of an attack or system deficiency.

18. JSON Web Encryption-based authentication

Severity: Low
Type: Authentication
Difficulty: Medium
Finding ID: TOB-ARGO-TM18

Description
Argo Workflows relies on JSON Web Encryption (JWE) for authentication. While using a JWE is a convenient method of providing authentication to multiple clients that share access to a service, JWEs cannot be invalidated upon logout or when new sessions are created before they expire.

Justification
The difficulty is medium for the following reasons:

- Attackers would need to obtain valid JWEs through other vulnerabilities.
- Users cannot take proactive steps to invalidate tokens.

The severity is medium for the following reason:

- Attackers who obtain valid JWE tokens may be able to perform authenticated tasks via the API until the tokens expire.

Recommendation
Short term, given that JWEs are necessary in this type of architecture and intended usage, consider mitigating the associated risks by setting sensible expiration dates for tokens and issuing refresh tokens so that users can maintain a valid session through the Web UI. Additionally, leverage cookies to store JWEs to the server and avoid using LocalStorage to store JWEs.

Argo Events Findings

Argo Events is an event-based dependency management engine. Events can be configured from a variety of sources, such as webhooks, S3 events, and Kafka messages.

Events are monitored by an event source controller pod and placed in a service bus. Service bus messages contain an event-specific payload or message, as well as a set of common metadata created by the event pod. Service bus messages are monitored by a service bus controller. When a sensor controller pulls a message from an event bus, a trigger (such as calling lambda functions or sending an HTTP request) is executed.

19. Insufficient logging of dropped events

Severity: Low
Type: Networking
Difficulty: Medium
Finding ID: TOB-ARGO-TM19

Description
Currently, Argo Events logs information regarding NATS streaming server connections, subscriptions, and requests and responses between sensors and trigger destinations, as well as various pipeline delivery information from event sources.

However, there may be insufficient logging performed when events are dropped from the event delivery queue due to a delivery failure.

Justification
The difficulty is medium for the following reasons:

- Webhooks can trivially become unavailable, resulting in an event being dropped upon delivery failure.
- However, the probability of this occurring and causing an impactful threat scenario in which logs may resolve an issue is not high.

The severity is low for the following reason:

- The absence of an audit trail typically does not result in a critical impact but may reduce an operator's ability to effectively mitigate an attack or system deficiency.

Recommendation
Short term, consider logging more information regarding messages dropped from the event bus and sensors in the event of a failed delivery.

20. Undocumented potential race condition in Event Bus

Severity: Medium
Type: Networking
Difficulty: Low
Finding ID: TOB-ARGO-TM20

Description
Events placed in the event bus may be meant to undergo processing by sensors in a first-in-first-out (FIFO) fashion. However, if a sensor receives an event and is unable to fire the corresponding trigger, it may grab the next message from the bus and process it before it retries sending the original message.

However, users may assume that fired events will retain the order in which they were fired.

Justification
The difficulty is low for the following reason:

- It would not be difficult to trigger such an event. If a trigger destination is unavailable for some time, it may result in this behavior.

The severity is medium for the following reason:

- Maintaining event order integrity may not seem critical to Argo Events itself; however, given the lack of documentation, a user could assume that integrity is maintained and could use Argo Events to drive connected applications into an invalid state.

Recommendation
Short term, consider documenting the behavior of Argo Events message delivery. Ensure that users are aware that event order integrity may not be maintained.

Long term, review documentation to identify cases in which insufficient documentation could cause a user to form incorrect assumptions about the system. Ensure that the behavior of the system, including inputs and outputs, is well documented.

21. Event Bus contains plaintext events

Severity: Low
Type: Cryptography
Difficulty: High
Finding ID: TOB-ARGO-TM21

Description
Events are commonly placed in the event bus through the process of base64 encoding the event data (or wrapped event data).

However, an attacker who gains access to the event bus may be able to discern all received events. This may not seem problematic, but consider a case in which the event bus is compromised due to a vulnerability (possibly in the NATS streaming spec or some other functionality). An operator could attempt to ratchet down the system and could overlook the fact that the attacker gained persistent access to the event bus. The attacker could then continue to consume events due to the lack of cryptography employed.

Justification
The difficulty is high for the following reason:

- Gaining access to the event bus would be non-trivial, as the event bus would not be publicly exposed and would have a smaller attack surface area.

The severity is low for the following reasons:

- Access to most components within Argo Events could result in similar event compromise.
- However, if such cases can be avoided in some portions of the infrastructure, they should be.

Recommendation
Short term, consider employing a shared secret among event sources and sensors such that data sent through the event bus cannot be read by an attacker who has gained access to the event bus.

22. Events cannot be optionally rate limited

Severity: Medium
Type: Networking
Difficulty: High
Finding ID: TOB-ARGO-TM22

Description
During the remote threat modeling sessions, it was established that events can be given a customizable expiration date. However, there is no rate-limiting method for events being received and processed.

In the event that a user application is connected to Argo Events as a trigger destination, events connected via Webhooks may overwhelm applications. Otherwise, events received at a high frequency could pollute the event bus, leading to performance or integrity issues.

Justification
The difficulty is high for the following reasons:

- Event origins are unlikely to produce events at a rate too rapid for Argo Events to handle.
- Event origins are opted into in Argo Events. Though removing an event origin is not ideal, it is an option if an event origin becomes problematic.

The severity is low for the following reasons:

- Event origins are typically opted in to and can be removed if problematic.
- Although removing event origins is not an ideal solution and end users may nonetheless wish to set up their applications as trigger destinations. Argo Events should be able to provide a rate-limiting functionality.
- The frequency with which an event origin reports events may increase such that it is unacceptable. Users should be able to protect themselves from increases beyond their desired rate limits.

Recommendation
Short term, consider implementing rate limiting for the consumption of events from event origins to avoid overwhelming a trigger destination with a sudden increase in events. Capture cases in which the rate limit is reached in logs so that users know if events are being produced at a rate faster than desired. Alternatively, document the behavior surrounding event frequencies if this cannot be fixed for some event origin provider types.

Long term, consider cases in which Argo services accept external input. Ensure that this input is not used at a high frequency in a way that triggers resource exhaustion attacks by implementing notions of rate limiting where possible. Similarly, consider implementing controllable limits for the size of this input.

Argo Rollouts Findings

Argo Rollouts is a Kubernetes controller and set of resources which provide advanced deployment capabilities.

The rollout controller modifies a user's existing service mesh, NGINX ingress service, or AWS ALB configuration to enable deployment strategies such as blue-green, canary, canary analysis, experimentation, and progessive delivery features.

Due to Argo Rollouts' small attack surface, we were not able to identify any concerns within the system.

A. RRA Template

Overview

- Component:
- Owner(s):
- SIG/WG(s) at Meeting:
- Service Data Classification:
- Highest Risk Impact:

Service Notes

This portion should walk through the components and discuss connections and their relevant controls and generally lay out how the components serve their relevant functions. For example, a component that accepts an HTTP connection may have relevant questions about channel security (TLS and cryptography), authentication, authorization, non-repudiation/auditing, and logging. The questions are meant to guide our discussions and to keep meetings and calls on track but are not the only drivers of discussions.

How does the service work?

Are there any subcomponents or shared boundaries?

What communication protocols does it use?

Where does it store data?

What is the most sensitive data it stores?

How is that data stored?

Data Dictionary

Name	Classification/Sensitivity	Comments
Data	Goes	Here

Control Families

We're interested in these areas of controls based on the audit working group's choices.

In this context, a "control" is a logical section of an application or system that handles a security requirement. According to CNSSI,

> “The management, operational, and technical controls (i.e., safeguards or countermeasures) prescribed for an information system to protect the confidentiality, integrity, and availability of the system and its information.”

For example, a system may have authorization requirements such as the following:

- Users must be registered with a central authority.
- All requests must be verified to be owned by the requesting user.
- Each account must have attributes associated with it to uniquely identify the user.

For this assessment, we're looking at six basic control families:

- Networking
- Cryptography
- Secrets Management
- Authentication
- Authorization (Access Control)
- Multi-tenancy Isolation

Obviously, we can consider a control family to be "not applicable" in the event that a component does not require it. For example, something with the sole purpose of interacting with the local file system may have no meaningful networking component. This isn't a weakness; the control family is simply "not applicable."

For each control family, we want to ask the following:

- What does the component do for this control?
- What sorts of data pass through that control?
    - For example, a component may have sensitive data (secrets management), but that data never leaves the component's storage via networking.
- What can an attacker do with access to this component?
- What's the simplest attack against it?
- Are there mitigations that we recommend (e.g., "Always use an interstitial firewall")?
- What happens if the component stops working (via DoS or other means)?
- Have there been similar vulnerabilities in the past? What were the mitigations?

Threat Scenarios

- An external attacker without access to the client application

- An external attacker with valid access to the client application
- An internal attacker with access to a cluster
- A malicious internal user

Networking

Cryptography

Secrets Management

Authentication

Authorization

Multi-tenancy Isolation

Summary

Recommendations