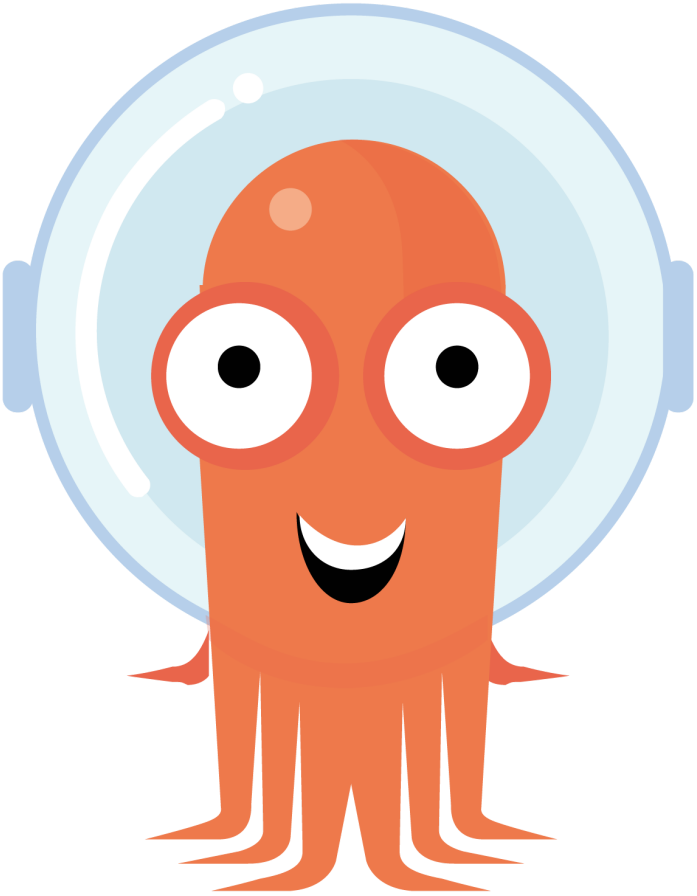
controlplane

CLOUD NATIVE COMPUTING FOUNDATION



Argo CD End User Threat Model

Security Considerations for Hardening Declarative GitOps CD on Kubernetes

Table of Contents

Executive Summary

Cloud native technologies bring a great homogenisation to the skillsets, operational theory, and systems complexity that makes building and running them a more easily understood and secured problem. There are many eyes on the developmental and supply chain security of cloud native software, but far fewer on the greatest cause of cloud breach: misconfiguration.

Progressive delivery of applications with the GitOps pattern has proliferated. It provides inversion of control for build servers that would otherwise write to production, instead indirectly writing to a Git repository to reduce overall privilege, and a statically analysable declarative configuration that supports security testing before deployment, reducing configuration drift.

In light of the important place that GitOps occupies in the cloud native application and security ecosystems, ControlPlane was engaged by the Cloud Native Computing Foundation to provide a comprehensive threat modelling analysis of a representative production setup of Argo CD within the context of an end user application deployment, and its associated management infrastructure.

In this document we enumerate quantifiable hardening recommendations and controls for Argo CD operators and architects. These controls are baselined against threats to an organisation using Argo CD based upon the official documentation, with threats criticality ordered for a non-classified data use case.

The Argo CD deployment in scope for this threat model operates in multi-tenant mode on an "Operations" Kubernetes cluster hosting the Argo CD control plane, and managing three subordinate tenant clusters.

The report findings identify nineteen (19) threats with varying priority levels, including six (6) high-priority threats that pose significant risks to this end user's security posture. These enumerated threats account for risks such as the storage of the Argo CD initial admin password as a Kubernetes Secret object, the use of local users' credentials without strong authentication, and the storage of Argo CD tenant cluster credentials as Kubernetes Secret objects.

To mitigate identified threats, the report provides several immediately actionable recommendations to the end user, such as rotating and storing the admin password in an external key management solution with restricted RBAC for break-glass

activities, using Single Sign-On integration for local users, and restricting RBAC for reading secrets from the Argo CD namespace according to least privilege.

The report also includes the creation of two attack trees that cover highest ranked threats, to provide an approachable visualisation of the identified threat landscape. These attack trees may be used to guide non-technical stakeholders through the model in order to support justification of the application of controls.

In addition, the report provides a summary of the Argo CD deployment architecture and the Terraform code to reproduce it for validation — including the Operations Kubernetes cluster hosting the control plane and Gitea Server, synchronisation configurations, and SSH bastion host access security.

This report's technical insights into threat vectors associated with an unhardened instance of Argo CD are supported by actionable recommendations to enhance its security posture. By implementing these recommendations, end users reduce their overall risk in line with their organisational appetite, whilst enjoying the benefits and efficiencies of Argo CD.

—
Andrew Martin
CEO, ControlPlane

—
Andres Vega
Vice President of Operations, ControlPlane

Preface from the Argo CD Maintainers

The Argo project maintainers are immensely grateful to ControlPlane and the CNCF for creating and funding this initiative. Argo CD comes with a lot of deployment flexibility, which enables operators to tailor it to their situation. Having a threat model and analysis for a representative setup will assist us maintainers in creating a rich set of best practices for our large and diversified user community.

This document marks an important milestone in our ongoing and relentless effort to make Argo CD an indispensable tool in our community's DevSecOps toolbelt.

Introduction

Argo CD is a widely adopted continuous deployment tool that has become an integral component for many organisations seeking to streamline and automate their application delivery processes through GitOps native principles. This report performs an extensive threat modelling analysis of an Argo CD instance, focusing on the end user's application deployment and management infrastructure, by examining the key assumptions (documented in the succeeding architecture overview), and evaluating the existing deployment and configuration practices.

The document structure is outlined below:

- Scope of the threat modelling exercise
- Architecture design
- Data dictionary and data flows
- Threat actors interested in compromising the assets and services
- Key threats and recommendations
- Attack trees for high priority threats

Threats are labelled with a priority rating based on professional opinion, and are driven by the potential impact on a standard client infrastructure in case of an Argo CD-related compromise. With respect to individual priority ratings, "High" refers to the highest impact threats resulting in (ops, tenant) cluster takeover and unauthorised changes to Argo CD configuration/unauthorised deployment of applications in tenant clusters, "Medium" denotes threats involving information tampering and leakage of information, and "Low" refers to minor threats that have a limited impact on the security of operations and information stored in the target infrastructure.

Scope

This threat modelling report provides a comprehensive analysis of the Argo CD deployment, potential risks, and recommendations for improvements. The scope section outlines the aspects that are in scope and out of scope for this report.

In Scope

The focus of this report includes the design and deployment of the Argo CD instance and its interactions with the related Kubernetes clusters (both “Ops” and tenant clusters), but not the clusters themselves. In order to conduct threat modelling workshops and introduce repeatable Argo CD deployment, reusable Terraform modules were developed. Our analysis included reviewing architecture, design decisions, configuration practices, developing attack trees, categorising threats using the Red, Amber, Green (RAG) system, and offering remediation recommendations. Finally, we produced this threat model report and a one-page executive summary for quick reference.

Out of Scope

Technical features and peripheral components not covered in this report are essential to clarify so that accurate expectations can be established. Our analysis does not aim to provide an exhaustive list of hardening guidelines or security baselines, nor does it focus on configuring the Argo CD deployment towards specific data classifications, e.g. PII or health data. This report does not delve into the security assessment of peripheral Kubernetes environment components or involve auditing the Argo CD instance for compliance. Additionally, we have not conducted penetration testing of the Argo CD instance or any related infrastructure. Lastly, the development and building of applications deployed by Argo CD are beyond the scope of this report.

Related Resources

Trail of Bits - Argo Threat Model 2021
Ada Logics - Argo Security Audit 2022

Architecture Overview

The analysed architectural model aims to emulate a client use-case example of Argo CD, and as such, operates on a series of use-case assumptions. The key assumptions help to establish the threat model scope.

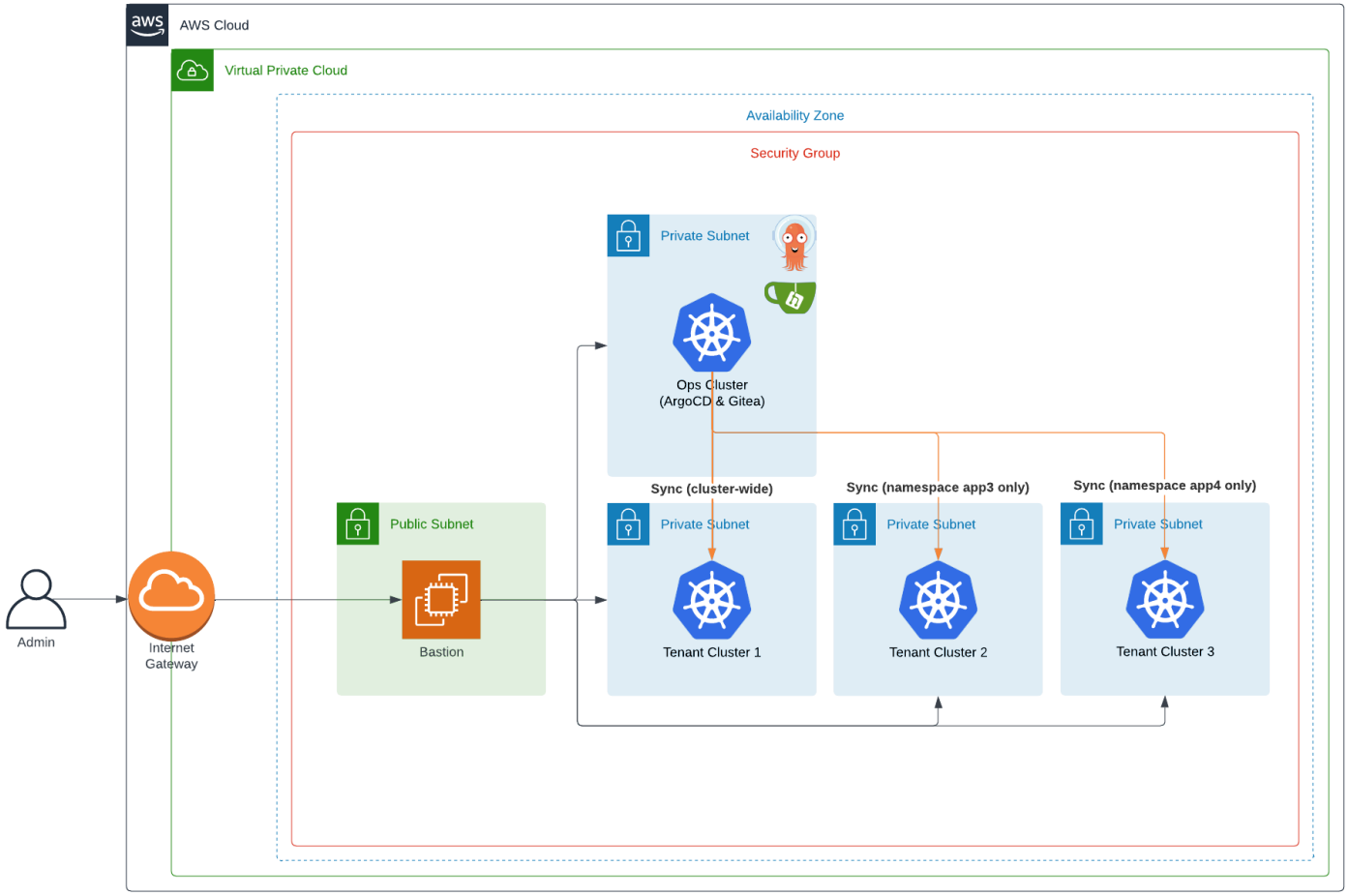
Summary

The assessed Argo CD deployment follows the official documentation, operating in multi-tenant mode on an "Operations" (Ops) Kubernetes 1.25 cluster that hosts the Argo CD control plane (version 2.6.7, the latest stable release at the time of writing) and a Gitea Server for managing application git repositories. Argo CD is deployed and configured through Terraform to an Amazon EKS cluster within its own private subnet. It is not configured to manage itself with a git repository. As such, operations such as adding new applications must be done by an administrator through Terraform, kubectl or the Argo CD CLI. Access is secured via an SSH bastion host.

Synchronisation is configured for select Argo CD applications, with auto-sync triggered by new commits to the main branch. A single git repository is used per application and mapped to clusters (and namespaces) in Argo CD to enable structured management and traceability. In order to threat model different deployment scenarios available, several tenant Kubernetes clusters have been introduced, and are added to the Argo CD instance with different levels of access:

- Tenant cluster 1 with cluster-wide access for the deployment of two applications, configured from a public and a private source code repository in Gitea
- Tenant cluster 2 with initial cluster-wide access, later re-configured to a specific namespace, which is used to deploy a single application (whose code is available from a public repository in Gitea)
- Tenant cluster 3 initially configured with namespace-level access to a specific namespace, used to deploy a single application (whose code is available from a public repository in Gitea)

Architecture Diagram



Key Assumptions

This section outlines the foundational premises that shape our analysis and recommendations for the deployment and management of Argo CD within the organisation. The key assumptions are as follows:

1. **Tenant Assignment**: Tenants are strategically assigned to various applications teams within the same organisation, fostering collaboration and ensuring efficient resource allocation
2. **Argo CD Deployment and Configuration**: Argo CD is deployed following many of the best practices outlined in the official documentation, utilising Kubernetes manifests for deployment and operating in a multi-tenant mode on an Operations cluster. This Operations cluster serves a dual purpose, hosting both the Argo CD control plane and the Gitea Server. It is important to emphasise that the Argo CD instance focuses on synchronising the state on tenant clusters and does not maintain substantial permissions over the Operations cluster
3. **Independent Argo CD Setup**: The Argo CD setup is deployed and configured through Terraform, and does not manage itself. Consequently, tenant clusters do not have Argo CD installed. Argo CD operates on Amazon EKS within a private subnet, which is distinct from the private subnets of the tenant clusters. To access the Argo CD cluster, an SSH bastion host situated on a public subnet is used as a secure gateway
4. **Synchronisation Process**: Argo CD has been configured to synchronise application workloads in select namespaces on a subset of tenant clusters, as well as with unrestricted cluster-wide access on a specific tenant cluster. Applications in tenant clusters are set to auto-sync for all applications, which takes place automatically upon a new commit on the main branch in the repository
5. **Repository Organisation**: A single CD repository is utilised per application. These repositories are mapped to clusters in Argo CD, with a separate folder in the repository dedicated to storing manifests. Using separate repositories for each application allows for more fine-grained control of access to manage application configuration

Data

The Data Dictionary section of this report highlights the sensitive data storage and management aspects within the Argo CD deployment. Various components, such as credentials, application manifests, and integration secrets, are managed using Kubernetes Secrets and ConfigMaps. This summary provides context for the detailed Data Dictionary table that follows, offering an overview of the data elements involved in the Argo CD deployment and their respective storage mechanisms.

Data Dictionary

Priority legend

Red	High
Amber	Medium
Green	Low

Data Name / Type	Notes	Confidentiality	Integrity	Availability
Argo CD Kubernetes Service Account (token)	Service Account token(s) in the argocd namespaces used by Argo CD microservices. By default, the Service Account used by the Argo CD Server has cluster-wide privileges on the Ops cluster.	High (R)	Medium (A)	High (R)
Argo CD initial admin password	Initial password for admin user is created when Argo CD is deployed on the Ops cluster, and is stored as Kubernetes Secret in the argocd namespace	Medium (A)	Medium (A)	Medium (A)
Local users credentials	All local users' credentials are stored with bcrypt hash in a single Kubernetes Secret in the argocd namespace	Medium (A)	Medium (A)	Medium (A)
Tenant cluster credentials	Each tenant cluster is added with a dedicated Kubernetes Secret in the argocd namespace that includes the server API endpoint, bearerToken, and tlsClientConfig parameters	High (R)	Medium (A)	High (R)
Single Sign-On integration secrets	If enabled, Argo CD stores secrets required for Single Sign-On integration as Kubernetes Secrets in the argocd namespace	Medium (A)	Medium (A)	Low (G)
VCS private repository	In the case of a private repository, the dedicated Kubernetes Secret in the argocd namespace includes the credentials (e.g.,	Medium (A)	High (R) (if r/w)	Medium (A)

credentials	SSH key) used to access it from Argo CD			
Application manifests	Kubernetes resources managed by Argo CD that are stored in Redis, and may include injected environment variable secrets	Medium (A)	Low (G)	Low (G)
Git webhook secrets	Stored as Kubernetes Secret in the argocd namespace	Low (G)	Medium (A)	Medium (A)
VCS repository configuration	Repository URL stored in a dedicated Kubernetes Secret in the argocd namespace	Low (G)	Medium (A)	Medium (A)
GPG keys	If enabled, Argo CD stores GPG public keys for enforcing VCS commit verification in a ConfigMap in the argocd namespace	Low (G)	Medium (A)	Medium (A)
Notification service credentials	Stored as Kubernetes Secret in the argocd namespace	Low (G)	Low (G)	Low (G)

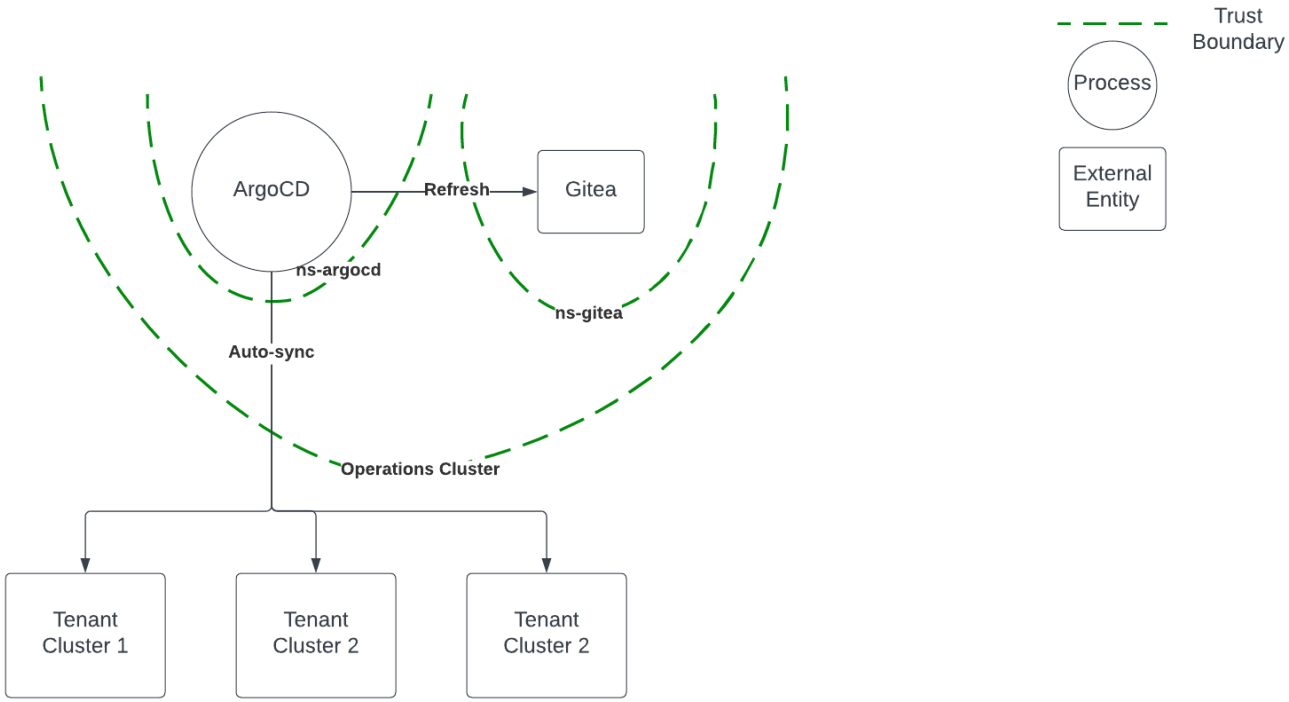
CIA impact assessment

Confidentiality	High	Cluster takeover (Operations, Tenant) due to leak of admin credentials.
	Medium	Sensitive information related to application or cluster state is exfiltrated.
	Low	Non-sensitive information leakage.
Integrity	High	Compromise of source code repositories and application deployments.
	Medium	Application sync fails due to misconfiguration / invalid configuration.
	Low	Non-critical operation is blocked due to misconfiguration / invalid configuration.
Availability	High	Cluster (Operations, Tenant) subject to DoS.
	Medium	Managed applications are blocked for a significant period.
	Low	Managed app synchronisation is blocked for a short period.

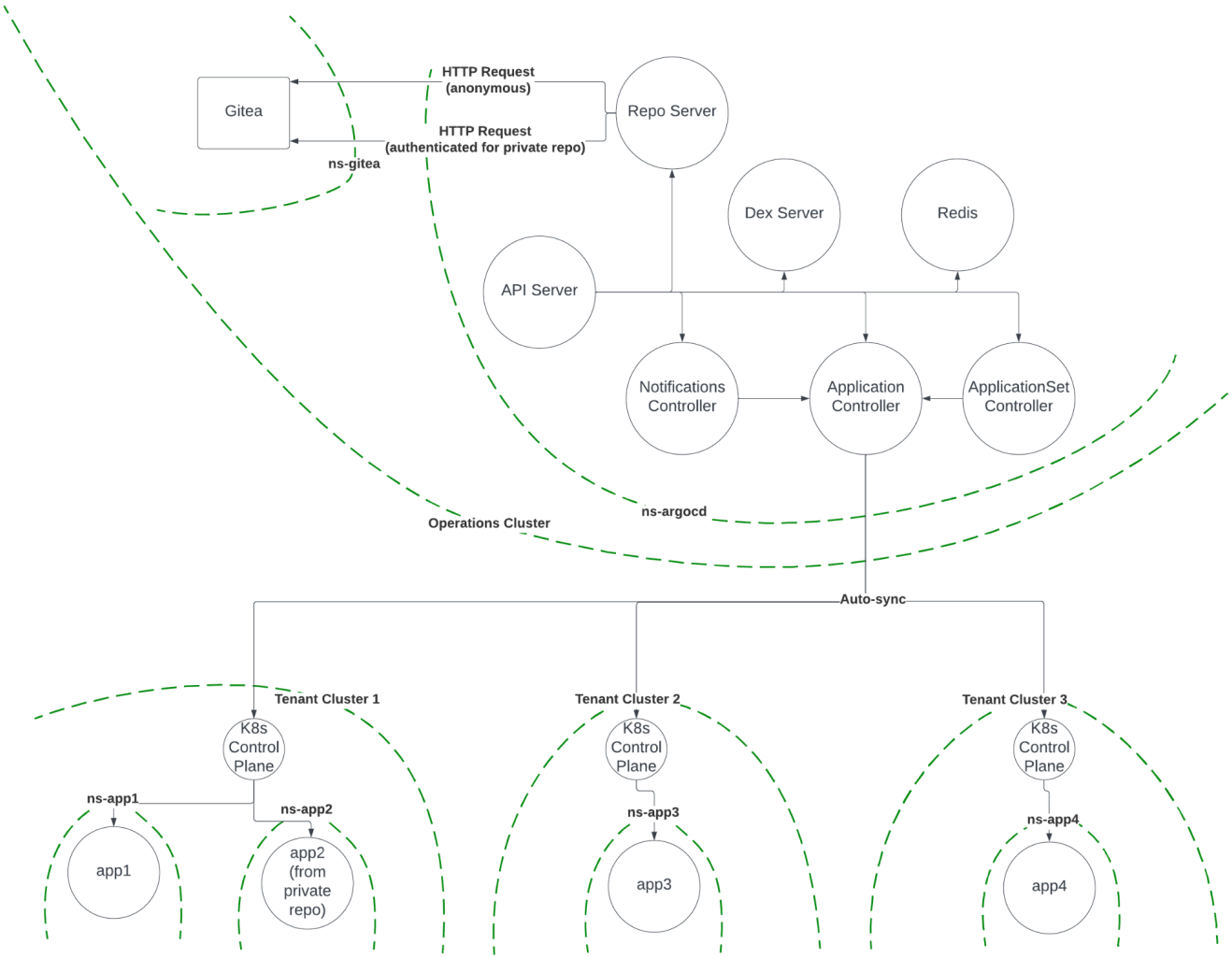
Data Flow Diagrams

The Data Flow Diagrams (DFDs) below describe the flow of data between the various processes, entities and data stores in a system, as well as the trust boundaries between different user roles and network interfaces. The DFDs are drawn at two different levels, starting at L0 (high-level system view) and increasing in granularity.

DFD L0



DFD L1



Key Threats and Recommendations

The scope of this threat model led to us categorising threats into priorities of High, Medium or Low; notably in a production implementation some of the threats' prioritisation may be upgraded or downgraded depending on the business context and data classification.

Risk vs Threat

For every finding, the risk and threat are stated. Risk defines the potential for negative outcome while threat defines the event that causes the negative outcome.

Threat Actors

In order to provide a realistic set of threats that is applicable to most organisations, we de-scoped the most advanced and hard to mitigate threat actors as described below:

In Scope

- Internal Attacker
    - Internal User
    - Administrator
    - Developer
    - End User
- External Attacker
    - Vandal: Script Kiddie, Trespasser
    - Motivated individual: Political activist, Thief, Terrorist
    - Organised crime: syndicates, state-affiliated groups

Out of Scope

- External Attacker
    - Cloud Service Insider: employee, external contractor, temporary worker
    - Foreign Intelligence Services (FIS): nation states

For more information on the threat actors in scope, please see Appendix (Threat Actors).

High Priority Findings

ATM-003 Initial admin password is stored as Kubernetes secret

ID	UID	Category	Priority
ATM-003	KR-AR-001	Argo CD RBAC	High

Risk: Argo CD initial admin password is stored as Kubernetes Secret.

Threat: Unauthorised changes to Argo CD configuration due to unauthorised read of Argo CD initial admin password from Kubernetes Secrets.

Recommendation: Rotate and store the Argo CD admin password in an external key management solution with restricted RBAC for break-glass activities, and delete if from the "argocd-initial-admin-secret" Kubernetes Secret. As stated in the official documentation - "the secret serves no other purpose than to store the initially generated password in clear and can safely be deleted at any time".

ATM-004 UI local users' credentials never expire, and without strong authentication

ID	UID	Category	Priority
ATM-004	KR-AR-002	Argo CD RBAC	High

Risk: Argo CD UI local users' credentials are never expiring, and are based on username/password without second factor authentication.

Threat: Unauthorised changes to Argo CD configuration due to weak or compromised local user credentials.

Recommendation: It is recommended to use the local admin user only for initial configuration and then switch to Single Sign-On integration (with the provided Dex server), unless it is considered overkill for a very small team. As soon as other users are created, the official documentation suggests disabling the admin user. The local users don't provide advanced features such as groups and login history. Moreover, it

is not possible to enforce Two-Factor Authentication with local users, as it is only available as part of the OAuth2 exchange with SSO providers.

ATM-006 Never-expiring tenant cluster credentials are stored as Kubernetes secrets

ID	UID	Category	Priority
ATM-006	KR-ASM-001	Argo CD Secrets Management	High

Risk: Argo CD tenant cluster credentials are stored as Kubernetes Secrets. The Argo CD argocd-manager Service Account token on the tenant cluster is set to never expire.

Threat: Unauthorised actions on tenant cluster due to unauthorised read of never-expiring tenant bearer token from Kubernetes Secret.

Recommendation: Restrict RBAC for reading Secrets from the Argo CD namespace according to least privilege. The preferred approach for end users of the project is to implement the cloud provider-specific workload identity solution (e.g. AWS IRSA, GCP Workload Identity, Azure AD Workload Identity) as an alternative to long-lived bearer tokens for Argo CD to apply configurations to tenant clusters. For a cloud-agnostic approach, the SPIFFE/SPIRE projects provide strongly attested, cryptographic identities to workloads across a wide variety of platforms. As stated in the official documentation, Argo CD has built-in support for AWS IAM authentication for EKS clusters through the 'awsAuthConfig' parameters for Cluster secrets. According to a merged PR (https://github.com/argoproj/argo-cd/pull/9190), Workload Identity for GKE clusters can be enabled by creating a GCP service account for Argo CD, binding the Kubernetes Service Account through annotation, and replacing the 'execProviderConfig' for the tenant cluster with the built-in 'argocd-k8s-auth' binary. As an alternative, investigate the use of external Key Management Service (e.g., AWS KMS) for managing Argo CD tenant cluster credentials and exposing them to the Argo CD Pods in a secure fashion. The latter approach does not protect against tenant cluster credentials being exfiltrated from Argo CD workloads at runtime, as they would still have access to the tenant bearer token(s). Ensure that bearer tokens used by Argo CD as tenant cluster credentials are rotated with a frequency that complies with the organisation policies. To manually rotate the bearer token used by Argo CD, the corresponding Kubernetes Secret can be deleted on the tenant cluster, which causes Kubernetes to regenerate a new secret with a new bearer token. The

new token can be re-inputted to Argo CD by re-running the 'argocd cluster add' command.

ATM-011 Default project is unrestricted

ID	UID	Category	Priority
ATM-011	KR-CW-003	CD Workflow	High

Risk: Argo CD "default" project is unrestricted.

Threat: The automatically created "default" project is assigned to any Application without a specified project and by default permits deployments from any source repo, to any cluster, and all resource Kinds. Under this default configuration, any user with access to create Applications in the default project could deploy malicious resources (including cluster-wide resource) from any repo to any tenant cluster.

Recommendation: Delete the "default" project immediately after installation, and consciously decide on a project topology e.g., one project per team/cluster/application. Alternatively, restrict the "default" project spec by replacing all wildcard ('*') access with an empty array ("[]").

ATM-018 Service Account has cluster-admin permissions on the Operations cluster

ID	UID	Category	Priority
ATM-018	KR-KR-001	Kubernetes RBAC	High

Risk: Argo CD Service Account has cluster-admin permissions on the Operations cluster.

Threat: Operations cluster takeover due to Argo CD over privileged Service Account.

Recommendation: Deploy Argo CD via its namespace-install.yaml manifest. This ensures that Argo CD only gets namespace-level privileges on the Argo CD namespace, and cannot apply any modification to its local cluster nor other application namespaces (e.g., Gitea). In this context, Argo CD can only apply changes to clusters and namespaces it is explicitly configured to manage.

ATM-019 ClusterRole and ClusterRoleBinding on tenant cluster grant cluster-admin permissions

ID	UID	Category	Priority
ATM-019	KR-KR-002	Kubernetes RBAC	High

Risk: Argo CD ClusterRole and ClusterRoleBinding on the tenant cluster have cluster-admin permissions in case of non-namespaced cluster add process. When access to the tenant cluster is descoped to namespace-level access, starting from cluster-wide access, these RBAC entities are not updated.

Threat: Tenant cluster takeover due to over-privileged Argo CD tenant cluster credentials in operations cluster.

Recommendation: Restrict privileges of Argo CD argocd-manager Service Account (SA) in tenant cluster adhering to least privilege principle. The SA in the tenant cluster should be granted the minimum set of read-write permissions that can grant synchronisation of managed applications. It should be assessed whether the SA can be descoped to remove access to select namespaces and reduce the attack surface in case of SA token compromise. In this regard, the RBAC configured in the tenant cluster needs to be manually updated to restrict access to specific namespaces in case the tenant cluster was initially added with cluster-wide access (as it happens for the Tenant Cluster 2). As stated in the official documentation, the Argo CD operations cluster still requires cluster-wide read-only privileges in the tenant cluster to function properly, which would enable an attacker to read secrets with more privileges in the tenant cluster and escalate privilege. As secondary precautions, following a defense-in-depth approach, the Argo CD admin should restrict Projects deploying to the destination cluster to only access the permitted namespace(s), and restrict the Cluster configuration to only allow deployment to the permitted namespaces.

Medium Priority Findings

ATM-001 Control plane is deployed in a non high-availability mode

ID	UID	Category	Priority
ATM-001	KR-AA-001	Argo CD Availability	Medium

Risk: Argo CD control plane is deployed in a non high-availability mode.

Threat: Application state sync fails on tenant cluster due to Argo CD control plane failure (internal microservices and/or Redis instance).

Recommendation: Deploy Argo CD with High Availability for production use through the available manifest bundle. This runs more containers, and runs Redis in HA mode. The HA installation will require at least three different nodes due to pod anti-affinity roles in the specs.

ATM-002 Unrestricted or insufficient policies for write access to source repository

ID	UID	Category	Priority
ATM-002	KR-AA-002	Argo CD Availability	Medium

Risk: Unrestricted or insufficient policies for write access to source repository used by Argo CD ApplicationSet git generator.

Threat: Reduced availability due to high computing strain on application workload via unregulated creation of Application resources in tenant clusters through ApplicationSet git generator.

Recommendation: Limit usage of git generators with ApplicationSets to specific use cases (e.g., self-service of Argo CD applications on multitenant clusters). Restrict Argo CD RBAC so that only trusted admins have read/write access to ApplicationSets. Lock-down write access to source repository by ensuring that commits to the target branch are reviewed and approved by the infrastructure team (or equivalent).

ATM-005 Insufficient command and control over ApplicationSet RBAC

ID	UID	Category	Priority
ATM-005	KR-AR-003	Argo CD RBAC	Medium

Risk: Insufficient command and control over ApplicationSet RBAC.

Threat: App misconfiguration and privilege escalation by way of creating unauthorised apps within unrestricted projects.

Recommendation: Hard-code the project field in the ApplicationSet template when possible to ensure that the project remains statically defined by the administrator, and reduce the risk of unauthorised access or manipulation. Configure Argo CD RBAC ConfigMap to restrict project permissions by the principle of least privilege, ensuring that only trusted admins have access to read/write ApplicationSets.

ATM-007 Private code repository credentials are stored as Kubernetes Secrets

ID	UID	Category	Priority
ATM-007	KR-ASM-002	Argo CD Secrets Management	Medium

Risk: Argo CD private code repository credentials are stored as Kubernetes Secrets.

Threat: Information leak and/or integrity compromise on source code repositories due to unauthorised use of read-only and read-write credentials for private code repositories.

Recommendation: Restrict RBAC for reading secrets from the Argo CD namespace according to least privilege. Investigate the use of external Key Management Service (e.g., AWS KMS) for managing Argo CD private repository credentials (e.g., SSH keys, HTTPS username and password, GitHub app credentials) and exposing them to the Argo CD Pods in a secure fashion. Ensure that private code repository credentials are restricted to read-only access, so that unauthorised modifications on source code cannot be performed.

ATM-008 GPG signature verification of Git commits is not enforced

ID	UID	Category	Priority
ATM-008	KR-CW-001	CD Workflow	Medium

Risk: GPG signature verification of Git commits is not enforced.

Threat: Untrusted code commit in CD repository triggers auto-sync of unauthorised changes in the tenant cluster.

Recommendation: Argo CD has built-in support for verifying GPG signed commits, although this is not enforced by default. It is recommended to assess the impact of enforcing GPG signature verification given the existing developer workflow. This would require importing GPG public keys in Argo CD and configuring the AppProject CustomResourceDefinition to enforce signature verification for the specific key IDs.

ATM-010 Limited capability to manage and validate auto-sync

ID	UID	Category	Priority
ATM-010	KR-CW-003	CD Workflow	Medium

Risk: Limited capability to manage and validate auto-sync between Argo CD and the tenant clusters when self-heal is enabled.

Threat: Unauthorised deployments, misconfigurations (leading to unstable environment), and increased attack surface due to auto-sync with tenant clusters.

Recommendation: Enforce signed commits on the source code repository to help ensure the integrity of the code and establish identity of the author(s). With regards to signing keys, it is recommended to store the private signing keys on a hardware token (e.g. YubiKeys) and to adopt a secure key distribution method. Enable Git branch protection to prevent unauthorised changes and/or force pushes to the main branch. Require pull requests and code reviews for changes to the main branch, ensuring that only approved changes are merged. Configure Argo CD to authenticate and validate incoming webhooks from Gitea (via HMAC signatures stored as Kubernetes secrets), ensuring that only authorised updates trigger the auto-sync process. Enforce HTTPS for webhook communication to encrypt the data in-transit.

ATM-013 Lack of visibility into Argo CD access control

ID	UID	Category	Priority
ATM-013	KR-DL-002	Detection & Logging	Medium

Risk: Lack of visibility into Argo CD access control for the web user interface and API.

Threat: Unauthenticated access to API endpoints can compromise the confidentiality and integrity of the hosted services. Inherent lack of transparency increases the risk of unauthorised individuals gaining access to sensitive data or modifying the application workload, ultimately jeopardising the security and stability of the services hosted within Argo CD.

Recommendation: Argo CD does not log client IP addresses when accessing its API endpoints. It is therefore recommended to configure access logging on the reverse proxy (e.g., Nginx) that sits in front of the 'argocd-service' Service in Kubernetes. The reverse proxy can be configured to track client connection metadata on top of IP addresses such as timestamp, request method, response status, and user agent to enable heuristic-based connection analysis. Moreover, user access should be logged within the secure bastion host (SSH connection details) to provide visibility into tunnelled connections.

ATM-014 Redis is not password-protected by default

ID	UID	Category	Priority
ATM-014	KR-DS-001	Data Security	Medium

Risk: Redis is not password-protected by default.

Threat: Exfiltration of sensitive information from cached information (e.g., rendered manifests) from Redis instance. Argo CD stores rendered manifests in Redis. Those rendered manifests are served to the application-controller by the repo-server when present. Through ability to inject different manifests, the integrity of the data can be compromised and thus deploy malicious resources.

Recommendation: Argo CD supports password-based Redis authentication via an undocumented procedure (https://github.com/argoproj/argo-cd/issues/3130) which requires the REDIS_PASSWORD environment variable to be set in the argocd-application-controller, argocd-server and argocd-repo-server deployments. It should be noted that including secret information as an environment variable is discouraged, and the Secret should be passed to the pod container as a local file via a volumeMount. It should be noted that Argo CD does not support password-based Redis authentication when deployed in high-availability mode.

ATM-015 Redis communications are not encrypted by default

ID	UID	Category	Priority
ATM-015	KR-DS-002	Data Security	Medium

Risk: Redis communications are not encrypted by default.

Threat: Exfiltration or tampering with information (e.g., rendered manifests) via Man-in-the-Middle (MITM) between Argo CD and Redis instance.

Recommendation: Enable TLS communication between Argo CD and Redis (https://github.com/argoproj/argo-cd/issues/5707). As an alternative, consider Service Mesh authentication for mutual TLS communication between all components in the Operations cluster (including the Argo CD and Redis Pods).

ATM-016 Gitea webhook secret(s) are not automatically validated

ID	UID	Category	Priority
ATM-016	KR-DS-003	Data Security	Medium

Risk: Gitea webhook secret(s) are not automatically validated.

Threat: Exposes the environment to unauthorised application updates or synchronise malicious configurations by sending spoofed webhook requests.

Recommendation: Configure Argo CD to validate HMAC signatures (which are auto-generated and included in the webhook HTTP header as a hashed combination of secret token & payload) within incoming webhook requests from the Gitea server. This can be performed by setting the webhookSecret field in the Argo CD CustomResourceDefinition (CRD). Additionally, the secret token should be stored as a Kubernetes Secret to perform webhook HMAC signature validation, ensuring the values match those specified in the CRD. Although the private subnet architecture limits exposure to external threats, HMAC signature validation remains a diligent practice to mitigate risks imposed by insider threats and reduce the potential for unauthorised webhook requests.

ATM-017 Lack of network segregation for the Argo CD resources

ID	UID	Category	Priority
ATM-017	KR-KN-001	Kubernetes Networking	Medium

Risk: Lack of network segregation for the Argo CD resources.

Threat: Exploitation of vulnerability in Argo CD component due to lateral movement from compromised workload in Operations cluster.

Recommendation: Restrict network access to the Argo CD namespace from other namespaces residing on the Operations cluster. This can be achieved via the implementation of a default-deny ingress Network Policy for all Pods in the Argo CD namespace. Implementation of an egress Network Policy may be more complex due to inherent limitations (e.g., lack of support for DNS names for tenant clusters' API server endpoints). Service Mesh authentication can be also implemented based on the target use case to restrict access to Argo CD Pods.

Low Priority Findings

ATM-009 Trusted GPG keys stored in a ConfigMap in the argocd namespace

ID	UID	Category	Priority
ATM-009	KR-CW-002	CD Workflow	Low

Risk: Trusted GPG keys stored in a ConfigMap in the argocd namespace.

Threat: Tampering with GPG keys allowlist to enable commit of untrusted changes to applications. Write access to ConfigMaps also means possessing the ability to edit argo-rbac-cm for an attacker to grant themselves admin access.

Recommendation: Restrict read-write access to ConfigMaps in the Operations cluster so that only authorised users and/or Service Accounts can edit the list of trusted GPG keys.

ATM-012 Lack of visibility into Argo CD application-level activity

ID	UID	Category	Priority
ATM-012	KR-DL-001	Detection & Logging	Low

Risk: Lack of visibility into Argo CD application-level activity.

Threat: Unauthorised access to the Argo CD environment (e.g., brute-forcing credentials, exploiting misconfigurations, or using leaked credentials). Once inside, they could escalate privileges to gain control over the Argo CD environment, deploy malicious applications, modify existing deployments, or exfiltrate sensitive data.

Recommendation: Argo CD emits Kubernetes Events of application activity, which include the username of the account that is responsible for any action when applicable. Kubernetes Event logs can be complemented with native SCM logging capabilities such as access, application, and git operation logs (via the Gitea source of truth), to provide an audit log of changes to Application configuration(s). Advanced configuration of Gitea logging capabilities can be performed in the app.ini config file provided within the official helm chart. It is recommended that these logs are maintained and actively monitored within the Argo CD environment through a log collection agent to provide visibility into application-level activity on private clusters.

Appendix

In Scope Threat Actor Details

Threat Actor	Capability	Personal Motivation	GitOps Attack Samples
Internal User	- Leverage internal knowledge and personal access to the Argo CD infrastructure to move laterally and transit trust boundaries	- Disgruntled / personal grievances - Financial incentives	- Expose sensitive information such as application secrets and API auth keys - Bypass or re-configure RBAC policies to gain unauthorised access beyond their privileges, which could lead to cluster-wide takeover
Administrator	- Abuse privileged status to disrupt operations and tenant cluster services through Argo CD misconfiguratio n	- Disgruntled / personal grievances - Financial incentives	- Unauthorised read of initial admin password to disable security settings (e.g., RBAC & network policies), modify existing app deployments to auto-sync - impacting tenant clusters, and cover tracks by deleting/altering logs - Alter application deployment (e.g., scale up) to consume extensive resources, causing service degradation - Fall victim to phishing attacks and inadvertently share authentication credentials to cloud infrastructure or Kubernetes clusters
Developer	- Alter application-level deployments by misconfiguring resource dependencies & SCM to introduce vulnerabilities	- Disgruntled / personal grievances - Financial incentives - Notoriety	- Craft malicious Kubernetes config (e.g., privileged pod/deployment) and deploy through the overprivileged Argo CD service account to abuse cluster-admin permissions, which can lead to reverse shell access, privilege escalation, service disruption, and cluster takeover - Introduce a remote code execution vulnerability (e.g., backdoor) within an application deployment

Threat Actor	Capability	Personal Motivation	GitOps Attack Samples
			- Commit sensitive data (e.g., app secrets and API tokens) to the Git repository, making them publicly accessible - Abuse ApplicationSets git generators to consume extensive resources, causing service degradation - Fall victim to phishing attack and inadvertently share authentication credentials to Argo CD hosted services
End User	- Misuse of authentication to hosted services and inadvertently introducing risk byway of insecure security practices	- Curiosity - Financial incentives - Ideological differences - Unwitting victim	- Abuse GUI features or API endpoints to perform unauthorised actions leading to information disclosure and performance degradation
Vandal: Script Kiddie, Trespasser	- Uses publicly available tools and applications (Nmap, Metasploit, CVE PoCs)	- Curiosity - Personal fame through defacement / denial of service of prominent public facing web resources	- Small scale DOS. - Launches prepackaged exploits, runs crypto mining tools - Exploit public-facing application services such as the bastion host to gain an initial foothold in the environment
Motivated individual: Political activist, Thief, Terrorist	- Write tools and exploits required for their means if sufficiently motivated - Tend to use these in a targeted fashion against specific organisations - May combine publicly available exploits in a	- Personal Gain (Political or Ideological)	- Phishing, DDOS, exploit known vulnerabilities - Compromise third-party components such as Helm charts and container images to inject malicious codes to propagate access throughout the environment

Threat Actor	Capability	Personal Motivation	GitOps Attack Samples
	targeted fashion - Tamper with open source supply chains		
Organised crime: syndicates, state-affiliated groups	- Write tools and exploits required for their means - Tend to use these in a non-targeted fashion, unless motivation is sufficiently high - Devotes considerable resources, writes exploits, can bribe/coerce, can launch targeted attacks	- Ransom - Mass extraction of PII / credentials / PCI data - Financial incentives	- Social Engineering, phishing, ransomware, coordinated attacks - Intercept and modify (via MiTM) unencrypted traffic between Argo CD and Git webhook requests to modify app configs in-transit

Identified Threats by Priority

Threats are ordered by priority for the scoped deployment model as outlined in the introductory rubric.

Priority legend

Red	High
Amber	Medium
Green	Low

Threats

ID	UID	Category	Risk	Threat	Priority	Recommendation
ATM-003	KR-AR-001	Argo CD RBAC	Argo CD initial admin password is stored as Kubernetes Secret.	Unauthorised changes to Argo CD configuration due to unauthorised read of Argo CD initial admin password from Kubernetes Secrets.	High (R)	Rotate and store the Argo CD admin password in an external key management solution with restricted RBAC for break-glass activities, and delete if from the “argocd-initial-admin-secret” Kubernetes Secret. As stated in the official documentation - “the secret serves no other purpose than to store the initially generated password in clear and can safely be deleted at any time”.
ATM-004	KR-AR-002	Argo CD RBAC	Argo CD UI local users’ credentials are never expiring,	Unauthorised changes to Argo CD configuration due to weak or compromised	High (R)	It is recommended to use the local admin user only for initial configuration and then switch to Single Sign-On integration (with the provided Dex server), unless it is considered overkill for a very small team. As soon as other users are

			and are based on username/password without second factor authentication.	local user credentials.		created, the official documentation suggests disabling the admin user. The local users don't provide advanced features such as groups and login history. Moreover, it is not possible to enforce Two-Factor Authentication with local users, as it is only available as part of the OAuth2 exchange with SSO providers.
ATM-006	KR-ASM-001	Argo CD Secrets Management	Argo CD tenant cluster credentials are stored as Kubernetes Secrets. The Argo CD argocd-manager Service Account token on the tenant cluster is set to never expire.	Unauthorised actions on tenant cluster due to unauthorised read of never-expiring tenant bearer token from Kubernetes Secret.	High (R)	Restrict RBAC for reading Secrets from the Argo CD namespace according to least privilege. The preferred approach for end users of the project is to implement the cloud provider-specific workload identity solution (e.g. AWS IRSA, GCP Workload Identity, Azure AD Workload Identity) as an alternative to long-lived bearer tokens for Argo CD to apply configurations to tenant clusters. For a cloud-agnostic approach, the SPIFFE/SPIRE projects provide strongly attested, cryptographic identities to workloads across a wide variety of platforms. As stated in the official documentation, Argo CD has built-in support for AWS IAM authentication for EKS clusters through the 'awsAuthConfig' parameters for Cluster secrets. According to a merged PR (https://github.com/argoproj/argo-cd/pull/9190), Workload Identity for GKE clusters can be enabled by creating a GCP service account for Argo CD, binding the Kubernetes Service Account through annotation, and replacing the 'execProviderConfig' for the tenant cluster with the built-in 'argocd-k8s-auth' binary. As an

						alternative, investigate the use of external Key Management Service (e.g., AWS KMS) for managing Argo CD tenant cluster credentials and exposing them to the Argo CD Pods in a secure fashion. The latter approach does not protect against tenant cluster credentials being exfiltrated from Argo CD workloads at runtime, as they would still have access to the tenant bearer token(s). Ensure that bearer tokens used by Argo CD as tenant cluster credentials are rotated with a frequency that complies with the organisation policies. To manually rotate the bearer token used by Argo CD, the corresponding Kubernetes Secret can be deleted on the tenant cluster, which causes Kubernetes to regenerate a new secret with a new bearer token. The new token can be re-inputted to Argo CD by re-running the 'argocd cluster add' command.

ATM-011	KR-CW-003	CD Workflow	Argo CD "default" project is unrestricted.	The automatically created "default" project is assigned to any Application without a specified project and by default permits deployments from any source repo, to any cluster, and all resource Kinds. Under this default configuration, any user with access to create Applications in the default project could deploy malicious resources (including cluster-wide resource) from any repo to any tenant cluster.	High (R)	Delete the "default" project immediately after installation, and consciously decide on a project topology e.g., one project per team/cluster/application. Alternatively, restrict the "default" project spec by replacing all wildcard ('*') access with an empty array ("[]").
ATM-018	KR-KR-001	Kubernetes RBAC	Argo CD Service Account has cluster-admin	Operations cluster takeover due to Argo CD over privileged Service Account.	High (R)	Deploy Argo CD via its namespace-install.yaml manifest. This ensures that Argo CD only gets namespace-level privileges on the Argo CD namespace, and cannot apply any modification to its local cluster nor other application namespaces (e.g., Gitea). In this context, Argo CD

			permissions on the Operations cluster.			can only apply changes to clusters and namespaces it is explicitly configured to manage.
ATM-019	KR-KR-002	Kubernetes RBAC	Argo CD ClusterRole and ClusterRole Binding on the tenant cluster have cluster-admin permissions in case of non-namespaced cluster add process. When access to the tenant cluster is descoped to namespace-level access, starting from cluster-wide access,	Tenant cluster takeover due to over-privileged Argo CD tenant cluster credentials in operations cluster.	High (R)	Restrict privileges of Argo CD argocd-manager Service Account (SA) in tenant cluster adhering to least privilege principle. The SA in the tenant cluster should be granted the minimum set of read-write permissions that can grant synchronisation of managed applications. It should be assessed whether the SA can be descoped to remove access to select namespaces and reduce the attack surface in case of SA token compromise. In this regard, the RBAC configured in the tenant cluster needs to be manually updated to restrict access to specific namespaces in case the tenant cluster was initially added with cluster-wide access (as it happens for the Tenant Cluster 2). As stated in the official documentation, the Argo CD operations cluster still requires cluster-wide read-only privileges in the tenant cluster to function properly, which would enable an attacker to read secrets with more privileges in the tenant cluster and escalate privilege. As secondary precautions, following a defense-in-depth approach, the Argo CD admin should restrict Projects deploying to the destination cluster to only access the permitted namespace(s), and restrict the Cluster configuration to only allow deployment to the permitted namespaces.

			these RBAC entities are not updated.			
ATM-001	KR-AA-001	Argo CD Availability	Argo CD control plane is deployed in a non high-availability mode.	Application state sync fails on tenant cluster due to Argo CD control plane failure (internal microservices and/or Redis instance).	Medium (A)	Deploy Argo CD with High Availability for production use through the available manifest bundle. This runs more containers, and runs Redis in HA mode. The HA installation will require at least three different nodes due to pod anti-affinity roles in the specs. Additionally, IPv6-only clusters are not supported.
ATM-002	KR-AA-002	Argo CD Availability	Unrestricted or insufficient policies for write access to source repository used by Argo CD Application Set git generator.	Reduced availability due to high computing strain on application workload via unregulated creation of Application resources in tenant clusters through ApplicationSet git generator.	Medium (A)	Limit usage of git generators with ApplicationSets to specific use cases (e.g., self-service of Argo CD applications on multitenant clusters). Restrict Argo CD RBAC so that only trusted admins have read/write access to ApplicationSets. Lockdown write access to source repository by ensuring that commits to the target branch are reviewed and approved by the infrastructure team (or equivalent).
ATM-005	KR-AR-003	Argo CD RBAC	Insufficient command and control over	App misconfiguration and privilege escalation by way	Medium (A)	Hard-code the project field in the ApplicationSet template when possible to ensure that the project remains statically defined by the administrator, and reduce the risk of

			Application Set RBAC.	of creating unauthorised apps within unrestricted projects.		unauthorised access or manipulation. Configure Argo CD RBAC ConfigMap to restrict project permissions by the principle of least privilege, ensuring that only trusted admins have access to read/write ApplicationSets.
ATM-007	KR-ASM-002	Argo CD Secrets Management	Argo CD private code repository credentials are stored as Kubernetes Secrets.	Information leak and/or integrity compromise on source code repositories due to unauthorised use of read-only and read-write credentials for private code repositories.	Medium (A)	Restrict RBAC for reading secrets from the Argo CD namespace according to least privilege. Investigate the use of external Key Management Service (e.g., AWS KMS) for managing Argo CD private repository credentials (e.g., SSH keys, HTTPS username and password, GitHub app credentials) and exposing them to the Argo CD Pods in a secure fashion. Ensure that private code repository credentials are restricted to read-only access, so that unauthorised modifications on source code cannot be performed.
ATM-008	KR-CW-001	CD Workflow	GPG signature verification of Git commits is not enforced.	Untrusted code commit in CD repository triggers auto-sync of unauthorised changes in the tenant cluster.	Medium (A)	Argo CD has built-in support for verifying GPG signed commits, although this is not enforced by default. It is recommended to assess the impact of enforcing GPG signature verification given the existing developer workflow. This would require importing GPG public keys in Argo CD and configuring the AppProject CustomResourceDefinition to enforce signature verification for the specific key IDs.

ATM-010	KR-CW-003	CD Workflow	Limited capability to manage and validate auto-sync between Argo CD and the tenant clusters when self-heal is enabled.	Unauthorised deployments, misconfigurations (leading to unstable environment), and increased attack surface due to auto-sync with tenant clusters.	Medium (A)	Enforce signed commits on the source code repository to help ensure the integrity of the code and establish identity of the author(s). With regards to signing keys, it is recommended to store the private signing keys on a hardware token (e.g. YubiKeys) and to adopt a secure key distribution method. Enable Git branch protection to prevent unauthorised changes and/or force pushes to the main branch. Require pull requests and code reviews for changes to the main branch, ensuring that only approved changes are merged. Configure Argo CD to authenticate and validate incoming webhooks from Gitea (via HMAC signatures stored as Kubernetes secrets), ensuring that only authorised updates trigger the auto-sync process. Enforce HTTPS for webhook communication to encrypt the data in-transit.
ATM-013	KR-DL-002	Detection & Logging	Lack of visibility into Argo CD access control for the web user interface and API.	Unauthenticated access to API endpoints can compromise the confidentiality and integrity of the hosted services. Inherent lack of transparency increases the risk of unauthorised individuals gaining access to sensitive data or	Medium (A)	Argo CD does not log client IP addresses when accessing its API endpoints. It is therefore recommended to configure access logging on the reverse proxy (e.g., Nginx) that sits in front of the 'argocd-service' Service in Kubernetes. The reverse proxy can be configured to track client connection metadata on top of IP addresses such as timestamp, request method, response status, and user agent to enable heuristic-based connection analysis. Moreover, user access should be logged within the secure bastion host (SSH connection details) to provide visibility into tunnelled connections.

				modifying the application workload, ultimately jeopardising the security and stability of the services hosted within Argo CD.		
ATM-014	KR-DS-001	Data Security	Redis is not password-protected by default.	Exfiltration of sensitive information from cached information (e.g., rendered manifests) from Redis instance. Argo CD stores rendered manifests in Rredis. Those rendered manifests are served to the application-controller by the repo-server when present. Through ability to inject different manifests, the integrity of the	Medium (A)	Argo CD supports password-based Redis authentication via an undocumented procedure (https://github.com/argoproj/argo-cd/issues/3130) which requires the REDIS_PASSWORD environment variable to be set in the argocd-application-controller, argocd-server and argocd-repo-server deployments. It should be noted that including secret information as an environment variable is discouraged, and the Secret should be passed to the pod container as a local file via a volumeMount. It should be noted that Argo CD does not support password-based Redis authentication when deployed in high-availability mode.

				data can be compromised and thus deploy malicious resources.		
ATM-015	KR-DS-002	Data Security	Redis communications are not encrypted by default.	Exfiltration or tampering with information (e.g., rendered manifests) via Man-in-the-Middle (MITM) between Argo CD and Redis instance.	Medium (A)	Enable TLS communication between Argo CD and Redis (https://github.com/argoproj/argo-cd/issues/5707). As an alternative, consider Service Mesh authentication for mutual TLS communication between all components in the Operations cluster (including the Argo CD and Redis Pods).
ATM-016	KR-DS-003	Data Security	Gitea webhook secret(s) are not automatically validated.	Exposes the environment to unauthorised application updates or synchronise malicious configurations by sending spoofed webhook requests.	Medium (A)	Configure Argo CD to validate HMAC signatures (which are auto-generated and included in the webhook HTTP header as a hashed combination of secret token & payload) within incoming webhook requests from the Gitea server. This can be performed by setting the webhookSecret field in the Argo CD CustomResourceDefinition (CRD). Additionally, the secret token should be stored as a Kubernetes Secret to perform webhook HMAC signature validation, ensuring the values match those specified in the CRD. Although the private subnet architecture limits exposure to external threats, HMAC signature validation remains a diligent practice to mitigate risks imposed by insider threats and reduce the potential for unauthorised webhook requests.

ATM-017	KR-KN-001	Kubernetes Networking	Lack of network segregation for the Argo CD resources.	Exploitation of vulnerability in Argo CD component due to lateral movement from compromised workload in Operations cluster.	Medium (A)	Restrict network access to the Argo CD namespace from other namespaces residing on the Operations cluster. This can be achieved via the implementation of a default-deny ingress Network Policy for all Pods in the Argo CD namespace. Implementation of an egress Network Policy may be more complex due to inherent limitations (e.g., lack of support for DNS names for tenant clusters' API server endpoints). Service Mesh authentication can be also implemented based on the target use case to restrict access to Argo CD Pods.
ATM-009	KR-CW-002	CD Workflow	Trusted GPG keys stored in a ConfigMap in the argocd namespace.	Tampering with GPG keys allowlist to enable commit of untrusted changes to applications.	Low (G)	Restrict read-write access to ConfigMaps in the Operations cluster so that only authorised users and/or Service Accounts can edit the list of trusted GPG keys.

ATM-012	KR-DL-001	Detection & Logging	Lack of visibility into Argo CD application-level activity.	Unauthorised access to the Argo CD environment (e.g., brute-forcing credentials, exploiting misconfigurations, or using leaked credentials). Once inside, they could escalate privileges to gain control over the Argo CD environment, deploy malicious applications, modify existing deployments, or exfiltrate sensitive data.	Low (G)	Argo CD emits Kubernetes Events of application activity, which include the username of the account that is responsible for any action when applicable. Kubernetes Event logs can be complemented with native SCM logging capabilities such as access, application, and git operation logs (via the Gitea source of truth), to provide an audit log of changes to Application configuration(s). Advanced configuration of Gitea logging capabilities can be performed in the app.ini config file provided within the official helm chart. It is recommended that these logs are maintained and actively monitored within the Argo CD environment through a log collection agent to provide visibility into application-level activity on private clusters.

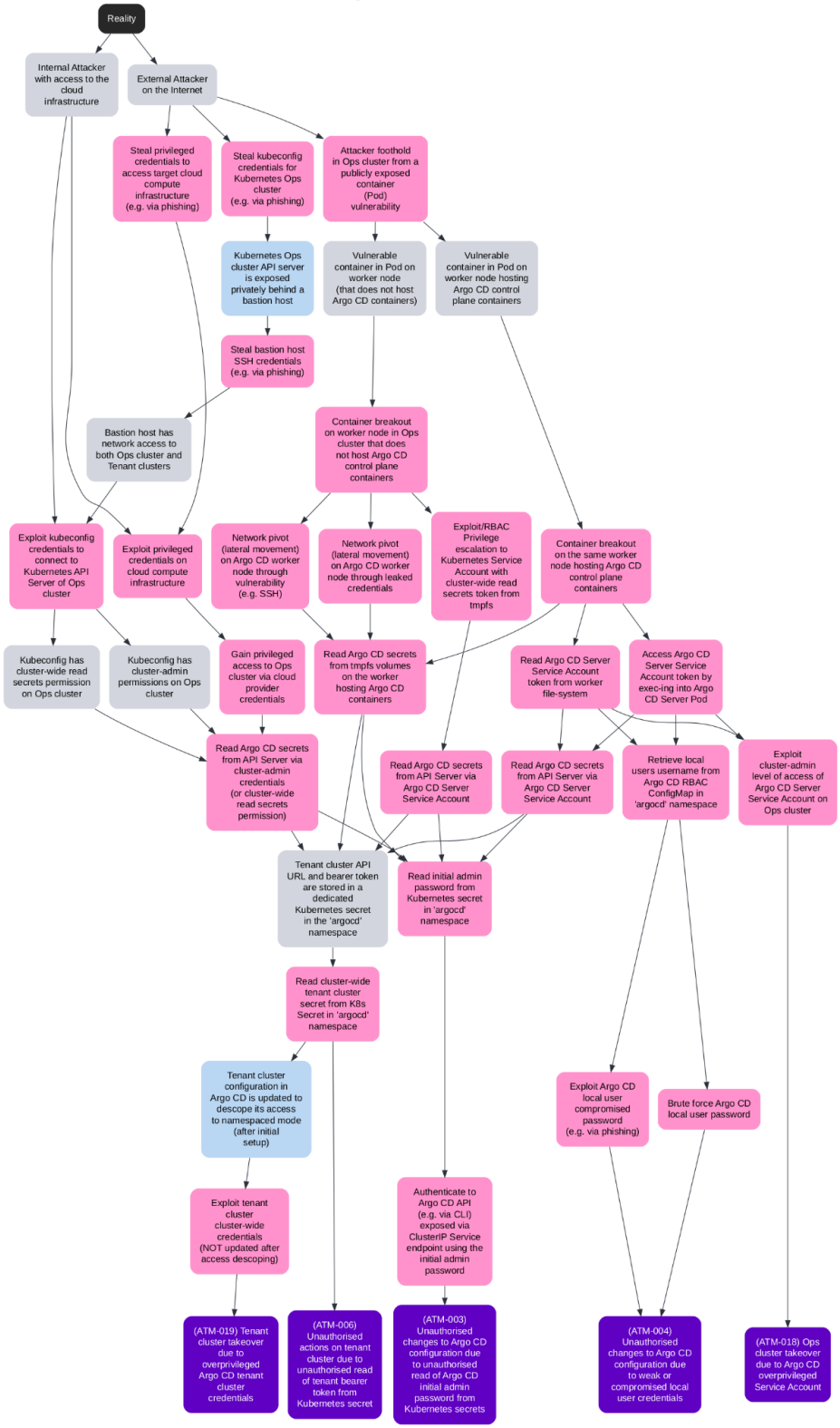
Attack Trees

Attack trees have been developed with the Deciduous web tool, and should be reviewed with a top-down approach, from the starting point for each scenario (e.g., “Reality”) to the goal of the attack. Typically, each attack has multiple branches and multiple goals, all starting from the same assumptions.
Attack paths include mitigations whenever a control is already in place to remediate an attack step.

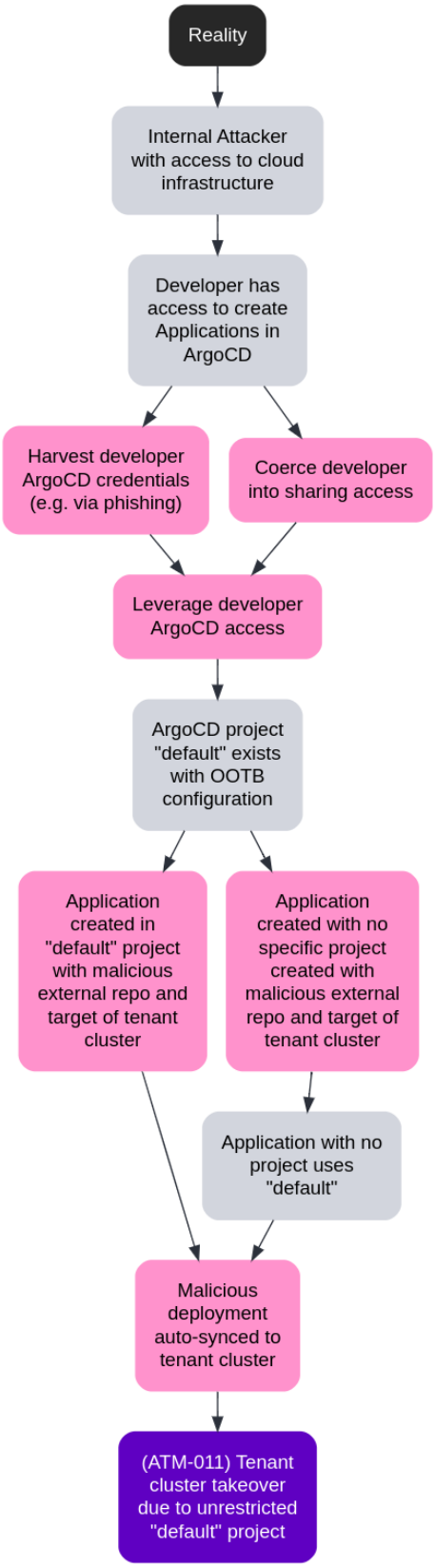
Attack Tree node legend

Reality	The starting point for the scenario
Fact	Something that is true to the system, and it’s neither an attack or a mitigation
Attack	A step in an attack path
Mitigation	An existing control to mitigate an attack step
Goal	The end goal for the attack

Attack Tree (ATM-003, ATM-004, ATM-006, ATM-018, ATM-019)



Attack Tree (ATM-011)



Sample Deployment

As part of this piece of work, we developed Terraform modules to recreate the Argo CD deployment analysed in this threat model. This allows third-party reproduction of the setup for the replication of the test exercise, and to serve as a baseline for a typical Argo CD deployment processing non-classified data. The codebase for the sample deployment is open-sourced at the following GitHub repository: https://github.com/controlplaneio/argocd-threat-model-infra.

As a prerequisite, an AWS account should already be in-place and available to the end user of the repository for the deployment of the Argo CD resources. The repository includes Terraform modules and supporting scripts to set up a primary EKS cluster (the "Ops" cluster) to instal Argo CD on, and three tenant EKS clusters that will be managed by Argo CD. It also instals Argo CD on the Operations cluster, connects the tenant clusters to it and deploys a Gitea instance, which serves as the SCM solution for our target architecture.

Starting from the infrastructure setup, the following application repositories have been instantiated on the VCS and then configured on Argo CD for the threat model exercise:

Organisation	Repository	Deployment Cluster
tenant0	app1 (public)	tenant0
tenant0	app2 (private)	tenant0
tenant1	app3 (public)	tenant1
tenant2	app4 (public)	tenant2

These resources are all private within a VPC in AWS, which is accessible from a bastion host via SSH. A Squid proxy is also configured so that HTTP traffic can be proxied from your own machine (e.g., running kubectl commands) via the bastion host.

There are four clusters that have been set up via Terraform, located in the "./terraform/clusters" directory. The "argo-eks" cluster is an Operations cluster that includes installations of Argo CD and Gitea in their respective namespaces. The "argo-tenant-0" cluster is a Tenant0 cluster where "app1" and "app2" (a private repo) are deployed to, and Argo CD is configured to the cluster clusterwide. The "argo-tenant-1" cluster is a Tenant1 cluster where "app3" is deployed to, and Argo CD

is also configured to the cluster clusterwide, but it is limited to managing the "app3" namespace only after setup. Lastly, the "argo-tenant-2" cluster is a Tenant2 cluster where "app4" is deployed to, and Argo CD is configured namespaced right with the setup.

About

ControlPlane is a global cloud native and open source cybersecurity consultancy operating in London, New York, and Auckland. We have industry-leading expertise in the architecture, audit, and implementation of zero trust infrastructure for regulated industries. With a deep understanding of secure-by-design and secure-by-default cloud, Kubernetes, and supply chain security we conduct threat modelling, penetration testing, and cloud native security training to the highest standard.

ControlPlane is trusted as the partner of choice in securing: multinational banks; major public clouds; international financial institutions; critical national infrastructure programs; multinational oil and gas companies, healthcare and insurance providers; and global media firms.

Threat Modelling Team

Ollie Cuffley-Hur
Marco De Benedictis
Jared Henderson
Henry Mortimer
Thomas Richter
Torin van den Bulk

Reviewers

Zach Aller
Rowan Baker
Henrik Blixt
James Callaghan
Justin Cappos
Michael Crenshaw
Dan Garfield
Andrew Martin
Katie Lamkin
Andrés Vega

Acknowledgments

While this threat modelling report was produced by the Cloud Native Computing Foundation engaging ControlPlane, the security knowledge and operational experience leading to it are a result of the hard work and dedicated collaboration of numerous practitioners across many contributing organisations to the Argo CD project and the broader cloud native security ecosystem..

Gratitude and thanks for the hard work to the Argo CD project team sustaining the advancement of the project alongside cloud native end user community for their selfless contributions to the project.

Special thanks to Priyanka Sharma and Chris Aniszczyk in the CNCF leadership. We would like to thank you for your vision, relentless innovation, and the platform for open systems engineering and collaboration you provide to the industry.