



Threagile

Agile Threat Modeling

Threat Model Report

Argo CD

13 November 2022

Argoproj Maintainers

Table of Contents

Results Overview

Management Summary	5
Impact Analysis of 88 Initial Risks in 21 Categories	6
Risk Mitigation	9
Impact Analysis of 88 Remaining Risks in 21 Categories	10
Application Overview	13
Data-Flow Diagram	14
Security Requirements	16
Abuse Cases	17
Tag Listing	18
STRIDE Classification of Identified Risks	19
Assignment by Function	23
RAA Analysis	27
Data Mapping	29
Out-of-Scope Assets: 0 Assets	30
Potential Model Failures: 6 / 6 Risks	31
Questions: 1 / 2 Questions	32

Risks by Vulnerability Category

Identified Risks by Vulnerability Category	33
Cross-Site Request Forgery (CSRF): 4 / 4 Risks	34
Cross-Site Scripting (XSS): 4 / 4 Risks	36
Missing Authentication: 10 / 10 Risks	38
Missing File Validation: 1 / 1 Risk	41
Missing Hardening: 2 / 2 Risks	43
Missing Identity Provider Isolation: 2 / 2 Risks	45
Path-Traversal: 1 / 1 Risk	47
Server-Side Request Forgery (SSRF): 27 / 27 Risks	49
Unguarded Access From Internet: 3 / 3 Risks	54
Accidental Secret Leak: 3 / 3 Risks	56
Code Backdooring: 4 / 4 Risks	58
Container Base Image Backdooring: 9 / 9 Risks	61
Missing Build Infrastructure: 1 / 1 Risk	64
Missing Identity Store: 1 / 1 Risk	66
Missing Network Segmentation: 1 / 1 Risk	68
Missing Two-Factor Authentication (2FA): 1 / 1 Risk	70
Missing Vault (Secret Storage): 1 / 1 Risk	72
Missing Web Application Firewall (WAF): 5 / 5 Risks	74

Unchecked Deployment: 4 / 4 Risks	76
Unencrypted Technical Assets: 1 / 1 Risk	78
Unnecessary Data Transfer: 3 / 3 Risks	80

Risks by Technical Asset

Identified Risks by Technical Asset	82
API Server: 16 / 16 Risks	83
Argo CD Source Repo (GitHub): 6 / 6 Risks	91
Docker Hub: 5 / 5 Risks	94
Host Cluster Kubernetes API: 4 / 4 Risks	97
Internal Source Control Management UI: 2 / 2 Risks	102
OIDC Provider (External): 6 / 6 Risks	104
OIDC Proxy (Dex): 6 / 6 Risks	107
Quay: 8 / 8 Risks	110
Rendered Manifests Cache (Redis): 2 / 2 Risks	115
Repo Server: 9 / 9 Risks	117
Application Controller: 6 / 6 Risks	123
ApplicationSet Controller: 4 / 4 Risks	127
Argo CD Build Pipeline (GitHub Actions): 5 / 5 Risks	130
Argo CD Maintainer Git Client: 2 / 2 Risks	133
External Cluster Kubernetes API: 2 / 2 Risks	135
Internal Source Control Management API: 2 / 2 Risks	138
Repo Server Storage: 2 / 2 Risks	141
User CLI: 1 / 1 Risk	143

Data Breach Probabilities by Data Asset

Identified Data Breach Probabilities by Data Asset	145
API Server Secret: 32 / 32 Risks	146
AppProject Manifest: 35 / 35 Risks	148
Application Manifest: 35 / 35 Risks	150
Application Name: 35 / 35 Risks	152
ApplicationSet Manifest: 33 / 33 Risks	154
ApplicationSet Name: 8 / 8 Risks	156
Argo CD Base Image: 11 / 11 Risks	157
Argo CD Container Image: 48 / 48 Risks	158
Argo CD Container Image Tag: 48 / 48 Risks	160
Argo CD GitHub Push Token: 6 / 6 Risks	162
Argo CD RBAC Config: 28 / 28 Risks	163
Argo CD Source: 12 / 12 Risks	165
Argo Tokens: 29 / 29 Risks	166

Cluster Access Configuration: 35 / 35 Risks	168
Cluster Access Credentials: 41 / 41 Risks	170
Git Branch Name: 7 / 7 Risks	172
Git Organization Name: 7 / 7 Risks	173
Git Repo URL: 7 / 7 Risks	174
Live Manifests: 41 / 41 Risks	175
Manifest Sources: 39 / 39 Risks	177
OIDC Public Keys: 43 / 43 Risks	179
OIDC Tokens: 44 / 44 Risks	181
Quay Push Token: 14 / 14 Risks	183
Rendered Manifests: 46 / 46 Risks	184
Repo Access Credentials: 40 / 40 Risks	186
Git Repo Name: 0 / 0 Risks	188

Trust Boundaries

Kubernetes Argo CD Namespace	189
Kubernetes Network	189
Organization Network	189

Shared Runtime

Kubernetes Node	190
-----------------	-----

About Threagile

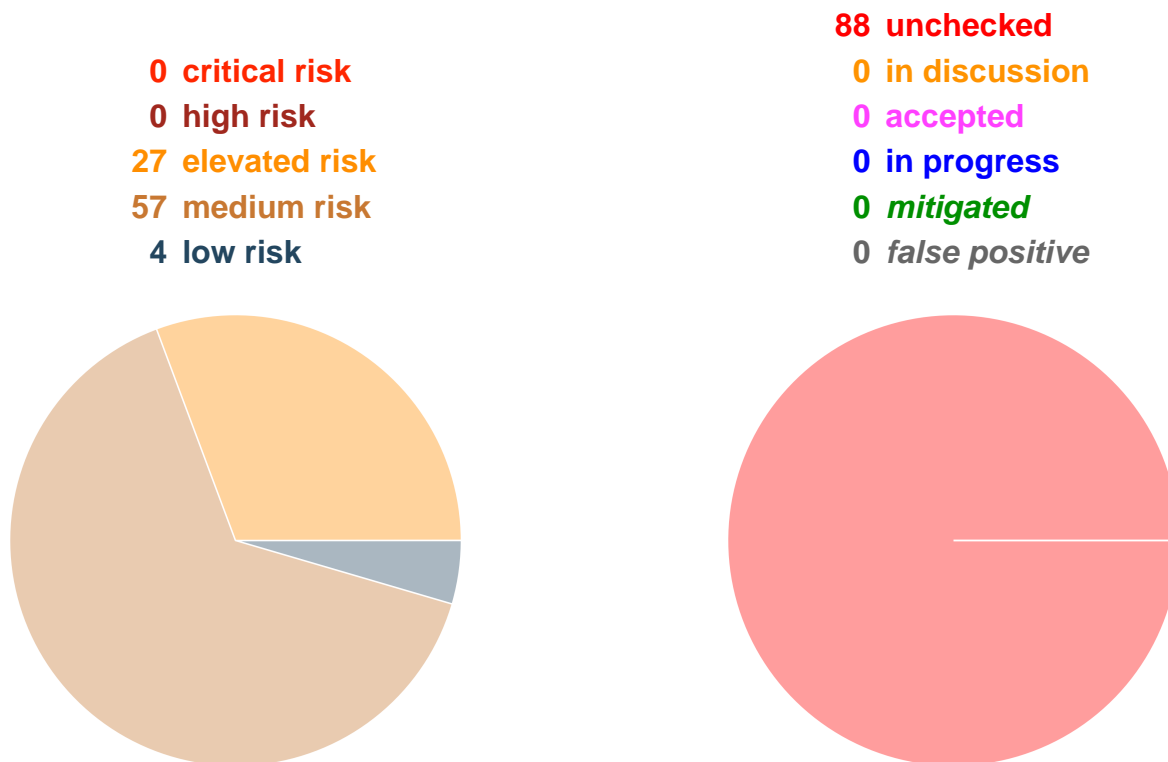
Risk Rules Checked by Threagile	191
Disclaimer	204

Management Summary

Threagile toolkit was used to model the architecture of "Argo CD" and derive risks by analyzing the components and data flows. The risks identified during this analysis are shown in the following chapters. Identified risks during threat modeling do not necessarily mean that the vulnerability associated with this risk actually exists: it is more to be seen as a list of potential risks and threats, which should be individually reviewed and reduced by removing false positives. For the remaining risks it should be checked in the design and implementation of "Argo CD" whether the mitigation advices have been applied or not.

Each risk finding references a chapter of the OWASP ASVS (Application Security Verification Standard) audit checklist. The OWASP ASVS checklist should be considered as an inspiration by architects and developers to further harden the application in a Defense-in-Depth approach. Additionally, for each risk finding a link towards a matching OWASP Cheat Sheet or similar with technical details about how to implement a mitigation is given.

In total **88 initial risks** in **21 categories** have been identified during the threat modeling process:



Just some **more** custom summary possible here...

Impact Analysis of 88 Initial Risks in 21 Categories

The most prevalent impacts of the **88 initial risks** (distributed over **21 risk categories**) are (taking the severity ratings into account and using the highest for each category):

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated: **Cross-Site Request Forgery (CSRF)**: 4 Initial Risks - Exploitation likelihood is *Very Likely with Medium impact*.

If this risk remains unmitigated, attackers might be able to trick logged-in victim users into unwanted actions within the web application by visiting an attacker controlled web site.

Elevated: **Cross-Site Scripting (XSS)**: 4 Initial Risks - Exploitation likelihood is *Likely with High impact*.

If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Elevated: **Missing Authentication**: 10 Initial Risks - Exploitation likelihood is *Likely with High impact*.

If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Elevated: **Missing File Validation**: 1 Initial Risk - Exploitation likelihood is *Very Likely with Medium impact*.

If this risk is unmitigated, attackers might be able to provide malicious files to the application.

Elevated: **Missing Hardening**: 2 Initial Risks - Exploitation likelihood is *Likely with Medium impact*.

If this risk remains unmitigated, attackers might be able to easier attack high-value targets.

Elevated: **Missing Identity Provider Isolation**: 2 Initial Risks - Exploitation likelihood is *Unlikely with Very High impact*.

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards highly sensitive identity provider assets and their identity datastores, as they are not separated by network segmentation.

Elevated: **Path-Traversal**: 1 Initial Risk - Exploitation likelihood is *Likely with High impact*.

If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) from the filesystem of affected components.

Elevated: **Server-Side Request Forgery (SSRF)**: 27 Initial Risks - Exploitation likelihood is *Likely with Medium impact*.

If this risk is unmitigated, attackers might be able to access sensitive services or files of network-reachable components by modifying outgoing calls of affected components.

Elevated: **Unguarded Access From Internet**: 3 Initial Risks - Exploitation likelihood is *Very Likely with Medium impact*.

If this risk is unmitigated, attackers might be able to directly attack sensitive systems without any hardening components in-between due to them being directly exposed on the internet.

Medium: Accidental Secret Leak: 3 Initial Risks - Exploitation likelihood is *Unlikely with High impact*.

If this risk is unmitigated, attackers which have access to affected sourcecode repositories or artifact registries might find secrets accidentally checked-in.

Medium: Code Backdooring: 4 Initial Risks - Exploitation likelihood is *Unlikely with High impact*.

If this risk remains unmitigated, attackers might be able to execute code on and completely takeover production environments.

Medium: Container Base Image Backdooring: 9 Initial Risks - Exploitation likelihood is *Unlikely with High impact*.

If this risk is unmitigated, attackers might be able to deeply persist in the target system by executing code in deployed containers.

Medium: Missing Build Infrastructure: 1 Initial Risk - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model due to critical build infrastructure components missing in the model.

Medium: Missing Identity Store: 1 Initial Risk - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model in the identity provider/store that is currently missing in the model.

Medium: Missing Network Segmentation: 1 Initial Risk - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards more valuable targets, as they are not separated by network segmentation.

Medium: Missing Two-Factor Authentication (2FA): 1 Initial Risk - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers might be able to access or modify highly sensitive data without strong authentication.

Medium: Missing Vault (Secret Storage): 1 Initial Risk - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers might be able to easier steal config secrets (like credentials, private keys, client certificates, etc.) once a vulnerability to access files is present and exploited.

Medium: Missing Web Application Firewall (WAF): 5 Initial Risks - Exploitation likelihood is *Unlikely with Medium impact*.

If this risk is unmitigated, attackers might be able to apply standard attack pattern tests at great speed without any filtering.

Medium: **Unchecked Deployment:** 4 Initial Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk remains unmitigated, vulnerabilities in custom-developed software or their dependencies might not be identified during continuous deployment cycles.

Medium: **Unencrypted Technical Assets:** 1 Initial Risk - Exploitation likelihood is *Unlikely* with *High* impact.

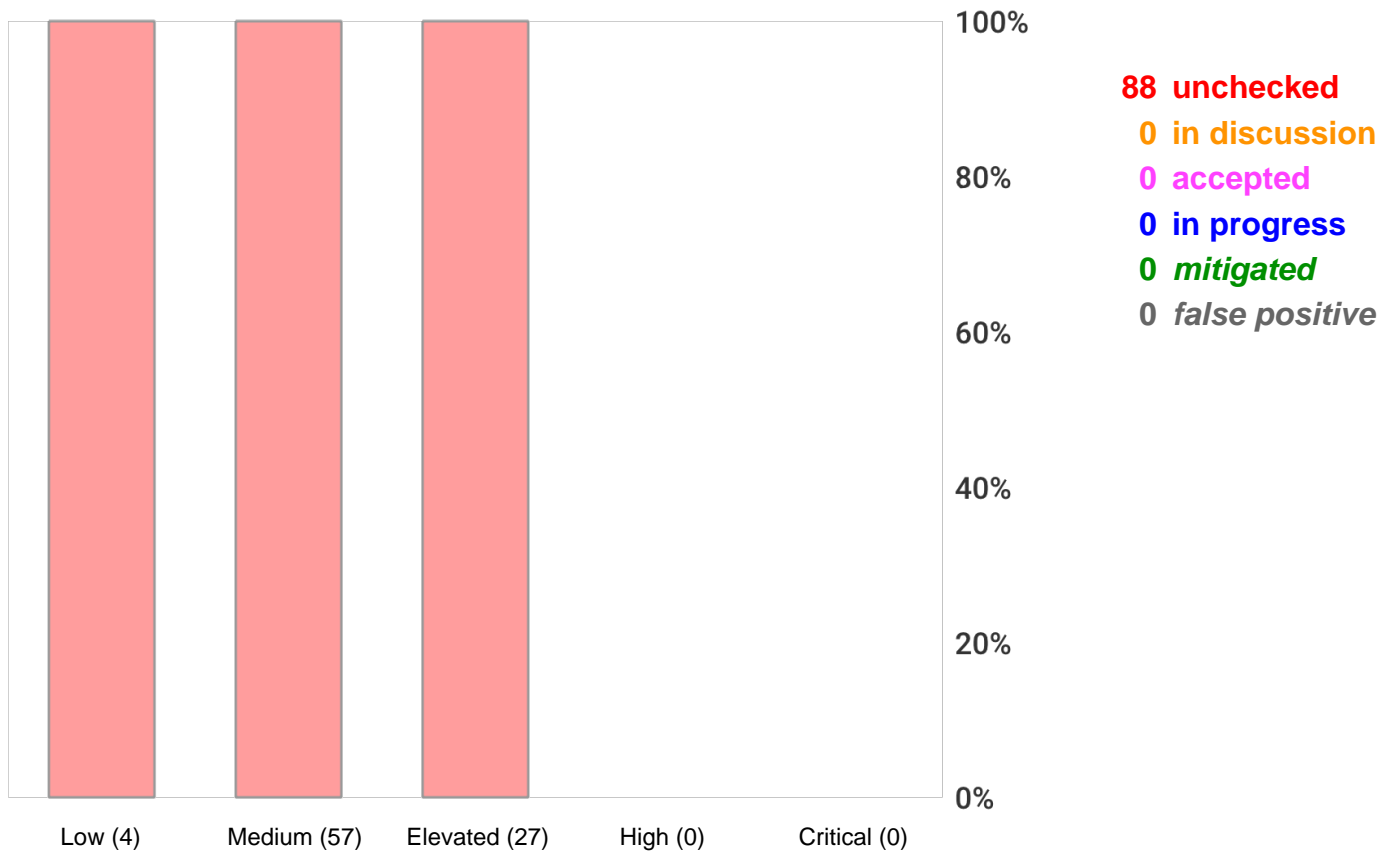
If this risk is unmitigated, attackers might be able to access unencrypted data when successfully compromising sensitive components.

Medium: **Unnecessary Data Transfer:** 3 Initial Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to target unnecessarily transferred data.

Risk Mitigation

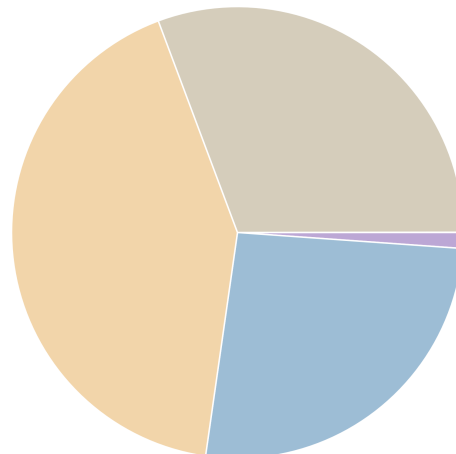
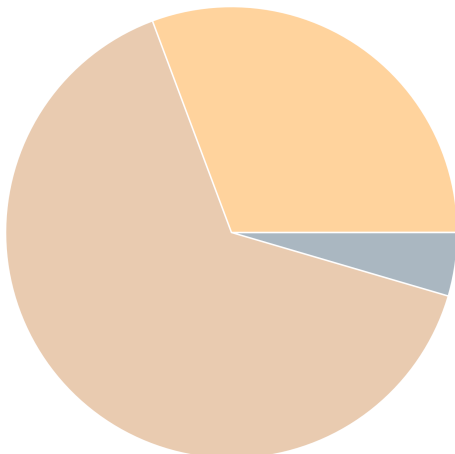
The following chart gives a high-level overview of the risk tracking status (including mitigated risks):



After removal of risks with status *mitigated* and *false positive* the following 88 remain unmitigated:

0 unmitigated critical risk
0 unmitigated high risk
27 unmitigated elevated risk
57 unmitigated medium risk
4 unmitigated low risk

1 business side related
23 architecture related
37 development related
27 operations related



Impact Analysis of 88 Remaining Risks in 21 Categories

The most prevalent impacts of the **88 remaining risks** (distributed over **21 risk categories**) are (taking the severity ratings into account and using the highest for each category):

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated: **Cross-Site Request Forgery (CSRF)**: 4 Remaining Risks - Exploitation likelihood is *Very Likely with Medium impact*.

If this risk remains unmitigated, attackers might be able to trick logged-in victim users into unwanted actions within the web application by visiting an attacker controlled web site.

Elevated: **Cross-Site Scripting (XSS)**: 4 Remaining Risks - Exploitation likelihood is *Likely with High impact*.

If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Elevated: **Missing Authentication**: 10 Remaining Risks - Exploitation likelihood is *Likely with High impact*.

If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Elevated: **Missing File Validation**: 1 Remaining Risk - Exploitation likelihood is *Very Likely with Medium impact*.

If this risk is unmitigated, attackers might be able to provide malicious files to the application.

Elevated: **Missing Hardening**: 2 Remaining Risks - Exploitation likelihood is *Likely with Medium impact*.

If this risk remains unmitigated, attackers might be able to easier attack high-value targets.

Elevated: **Missing Identity Provider Isolation**: 2 Remaining Risks - Exploitation likelihood is *Unlikely with Very High impact*.

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards highly sensitive identity provider assets and their identity datastores, as they are not separated by network segmentation.

Elevated: **Path-Traversal**: 1 Remaining Risk - Exploitation likelihood is *Likely with High impact*.

If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) from the filesystem of affected components.

Elevated: **Server-Side Request Forgery (SSRF)**: 27 Remaining Risks - Exploitation likelihood is *Likely with Medium impact*.

If this risk is unmitigated, attackers might be able to access sensitive services or files of network-reachable components by modifying outgoing calls of affected components.

Elevated: Unguarded Access From Internet: 3 Remaining Risks - Exploitation likelihood is *Very Likely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to directly attack sensitive systems without any hardening components in-between due to them being directly exposed on the internet.

Medium: Accidental Secret Leak: 3 Remaining Risks - Exploitation likelihood is *Unlikely* with *High* impact.

If this risk is unmitigated, attackers which have access to affected sourcecode repositories or artifact registries might find secrets accidentally checked-in.

Medium: Code Backdooring: 4 Remaining Risks - Exploitation likelihood is *Unlikely* with *High* impact.

If this risk remains unmitigated, attackers might be able to execute code on and completely takeover production environments.

Medium: Container Base Image Backdooring: 9 Remaining Risks - Exploitation likelihood is *Unlikely* with *High* impact.

If this risk is unmitigated, attackers might be able to deeply persist in the target system by executing code in deployed containers.

Medium: Missing Build Infrastructure: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model due to critical build infrastructure components missing in the model.

Medium: Missing Identity Store: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model in the identity provider/store that is currently missing in the model.

Medium: Missing Network Segmentation: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards more valuable targets, as they are not separated by network segmentation.

Medium: Missing Two-Factor Authentication (2FA): 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to access or modify highly sensitive data without strong authentication.

Medium: Missing Vault (Secret Storage): 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to easier steal config secrets (like credentials, private keys, client certificates, etc.) once a vulnerability to access files is present and exploited.

Medium: **Missing Web Application Firewall (WAF):** 5 Remaining Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to apply standard attack pattern tests at great speed without any filtering.

Medium: **Unchecked Deployment:** 4 Remaining Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk remains unmitigated, vulnerabilities in custom-developed software or their dependencies might not be identified during continuous deployment cycles.

Medium: **Unencrypted Technical Assets:** 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *High* impact.

If this risk is unmitigated, attackers might be able to access unencrypted data when successfully compromising sensitive components.

Medium: **Unnecessary Data Transfer:** 3 Remaining Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

If this risk is unmitigated, attackers might be able to target unnecessarily transferred data.

Application Overview

Business Criticality

The overall business criticality of "Argo CD" was rated as:

(archive | operational | **IMPORTANT** | critical | mission-critical)

Business Overview

Some more *demo text* here and even images...

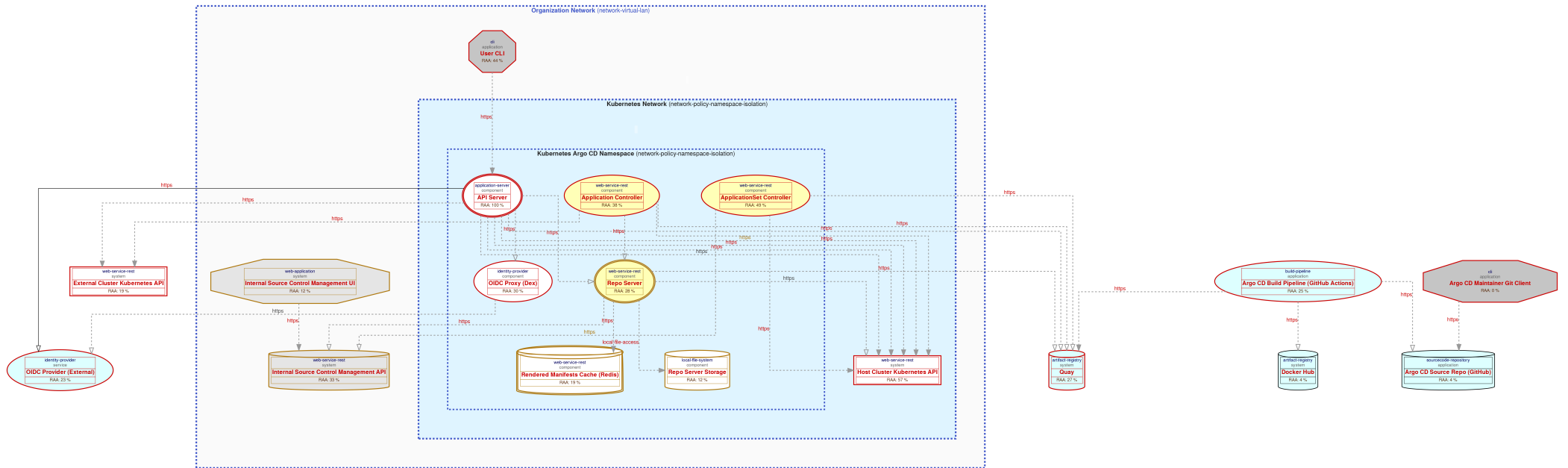
Technical Overview

Some more *demo text* here and even images...

Data-Flow Diagram

The following diagram was generated by Threagile based on the model input and gives a high-level overview of the data-flow between technical assets. The RAA value is the calculated *Relative Attacker Attractiveness* in percent. For a full high-resolution version of this diagram please refer to the PNG image file alongside this report.

Data-Flow Diagram - Argo CD



Security Requirements

This chapter lists the custom security requirements which have been defined for the modeled target.

Users are restricted by RBAC

RBAC rules defined by an Argo CD operator are actually enforced for API access.

This list is not complete and regulatory or law relevant security requirements have to be taken into account as well. Also custom individual security requirements might exist for the project.

Abuse Cases

This chapter lists the custom abuse cases which have been defined for the modeled target.

Deploying Crypto-Miner

Attacker uses a compromised Argo CD instance to deploy a crypto-miner

This list is not complete and regulatory or law relevant abuse cases have to be taken into account as well. Also custom individual abuse cases might exist for the project.

Tag Listing

This chapter lists what tags are used by which elements.

STRIDE Classification of Identified Risks

This chapter clusters and classifies the risks by STRIDE categories: In total **88 potential risks** have been identified during the threat modeling process of which **6 in the Spoofing** category, **29 in the Tampering** category, **0 in the Repudiation** category, **33 in the Information Disclosure** category, **0 in the Denial of Service** category, and **20 in the Elevation of Privilege** category.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Spoofing

Elevated: **Cross-Site Request Forgery (CSRF)**: 4 / 4 Risks - Exploitation likelihood is *Very Likely* with *Medium* impact.

When a web application is accessed via web protocols Cross-Site Request Forgery (CSRF) risks might arise.

Elevated: **Missing File Validation**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.

When a technical asset accepts files, these input files should be strictly validated about filename and type.

Medium: **Missing Identity Store**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

The modeled architecture does not contain an identity store, which might be the risk of a model missing critical assets (and thus not seeing their risks).

Tampering

Elevated: **Cross-Site Scripting (XSS)**: 4 / 4 Risks - Exploitation likelihood is *Likely* with *High* impact.

For each web application Cross-Site Scripting (XSS) risks might arise. In terms of the overall risk level take other applications running on the same domain into account as well.

Elevated: **Missing Hardening**: 2 / 2 Risks - Exploitation likelihood is *Likely* with *Medium* impact.

Technical assets with a Relative Attacker Attractiveness (RAA) value of 55 % or higher should be explicitly hardened taking best practices and vendor hardening guides into account.

Medium: **Code Backdooring**: 4 / 4 Risks - Exploitation likelihood is *Unlikely* with *High* impact.

For each build-pipeline component Code Backdooring risks might arise where attackers compromise the build-pipeline in order to let backdoored artifacts be shipped into production. Aside from direct code backdooring this includes backdooring of dependencies and even of more lower-level build infrastructure, like backdooring compilers (similar to what the XcodeGhost malware did) or dependencies.

Medium: Container Base Image Backdooring: 9 / 9 Risks - Exploitation likelihood is *Unlikely with High impact*.

When a technical asset is built using container technologies, Base Image Backdooring risks might arise where base images and other layers used contain vulnerable components or backdoors.

Medium: Missing Build Infrastructure: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

The modeled architecture does not contain a build infrastructure (devops-client, sourcecode-repo, build-pipeline, etc.), which might be the risk of a model missing critical assets (and thus not seeing their risks). If the architecture contains custom-developed parts, the pipeline where code gets developed and built needs to be part of the model.

Medium: Missing Web Application Firewall (WAF): 5 / 5 Risks - Exploitation likelihood is *Unlikely with Medium impact*.

To have a first line of filtering defense, security architectures with web-services or web-applications should include a WAF in front of them. Even though a WAF is not a replacement for security (all components must be secure even without a WAF) it adds another layer of defense to the overall system by delaying some attacks and having easier attack alerting through it.

Medium: Unchecked Deployment: 4 / 4 Risks - Exploitation likelihood is *Unlikely with Medium impact*.

For each build-pipeline component Unchecked Deployment risks might arise when the build-pipeline does not include established DevSecOps best-practices. DevSecOps best-practices scan as part of CI/CD pipelines for vulnerabilities in source- or byte-code, dependencies, container layers, and dynamically against running test systems. There are several open-source and commercial tools existing in the categories DAST, SAST, and IAST.

Repudiation

n/a

Information Disclosure

Elevated: Path-Traversal: 1 / 1 Risk - Exploitation likelihood is *Likely with High impact*.

When a filesystem is accessed Path-Traversal or Local-File-Inclusion (LFI) risks might arise. The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed or stored.

Elevated: Server-Side Request Forgery (SSRF): 27 / 27 Risks - Exploitation likelihood is *Likely with Medium impact*.

When a server system (i.e. not a client) is accessing other server systems via typical web protocols Server-Side Request Forgery (SSRF) or Local-File-Inclusion (LFI) or Remote-File-Inclusion (RFI) risks might arise.

Medium: **Accidental Secret Leak**: 3 / 3 Risks - Exploitation likelihood is *Unlikely with High impact*.

Sourcecode repositories (including their histories) as well as artifact registries can accidentally contain secrets like checked-in or packaged-in passwords, API tokens, certificates, crypto keys, etc.

Medium: **Missing Vault (Secret Storage)**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

In order to avoid the risk of secret leakage via config files (when attacked through vulnerabilities being able to read files like Path-Traversal and others), it is best practice to use a separate hardened process with proper authentication, authorization, and audit logging to access config secrets (like credentials, private keys, client certificates, etc.). This component is usually some kind of Vault.

Medium: **Unencrypted Technical Assets**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with High impact*.

Due to the confidentiality rating of the technical asset itself and/or the processed data assets this technical asset must be encrypted. The risk rating depends on the sensitivity technical asset itself and of the data assets stored.

Denial of Service

n/a

Elevation of Privilege

Elevated: **Missing Authentication**: 10 / 10 Risks - Exploitation likelihood is *Likely with High impact*.

Technical assets (especially multi-tenant systems) should authenticate incoming requests when the asset processes or stores sensitive data.

Elevated: **Missing Identity Provider Isolation**: 2 / 2 Risks - Exploitation likelihood is *Unlikely with Very High impact*.

Highly sensitive identity provider assets and their identity datastores should be isolated from other assets by their own network segmentation trust-boundary (execution-environment boundaries do not count as network isolation).

Elevated: **Unguarded Access From Internet**: 3 / 3 Risks - Exploitation likelihood is *Very Likely with Medium impact*.

Internet-exposed assets must be guarded by a protecting service, application, or reverse-proxy.

Medium: **Missing Network Segmentation**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

Highly sensitive assets and/or datastores residing in the same network segment than other lower sensitive assets (like webserver or content management systems etc.) should be better protected by a network segmentation trust-boundary.

Medium: **Missing Two-Factor Authentication (2FA): 1 / 1 Risk** - Exploitation likelihood is *Unlikely* with *Medium* impact.

Technical assets (especially multi-tenant systems) should authenticate incoming requests with two-factor (2FA) authentication when the asset processes or stores highly sensitive data (in terms of confidentiality, integrity, and availability) and is accessed by humans.

Medium: **Unnecessary Data Transfer: 3 / 3 Risks** - Exploitation likelihood is *Unlikely* with *Medium* impact.

When a technical asset sends or receives data assets, which it neither processes or stores this is an indicator for unnecessarily transferred data (or for an incomplete model). When the unnecessarily transferred data assets are sensitive, this poses an unnecessary risk of an increased attack surface.

Assignment by Function

This chapter clusters and assigns the risks by functions which are most likely able to check and mitigate them: In total **88 potential risks** have been identified during the threat modeling process of which **1 should be checked by Business Side**, **23 should be checked by Architecture**, **37 should be checked by Development**, and **27 should be checked by Operations**.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Business Side

Medium: **Missing Two-Factor Authentication (2FA)**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

Apply an authentication method to the technical asset protecting highly sensitive data via two-factor authentication for human users.

Architecture

Elevated: **Missing Authentication**: 10 / 10 Risks - Exploitation likelihood is *Likely with High impact*.

Apply an authentication method to the technical asset. To protect highly sensitive data consider the use of two-factor authentication for human users.

Elevated: **Unguarded Access From Internet**: 3 / 3 Risks - Exploitation likelihood is *Very Likely with Medium impact*.

Encapsulate the asset behind a guarding service, application, or reverse-proxy. For admin maintenance a bastion-host should be used as a jump-server. For file transfer a store-and-forward-host should be used as an indirect file exchange platform.

Medium: **Missing Build Infrastructure**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

Include the build infrastructure in the model.

Medium: **Missing Identity Store**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

Include an identity store in the model if the application has a login.

Medium: **Missing Vault (Secret Storage)**: 1 / 1 Risk - Exploitation likelihood is *Unlikely with Medium impact*.

Consider using a Vault (Secret Storage) to securely store and access config secrets (like credentials, private keys, client certificates, etc.).

Medium: **Unchecked Deployment**: 4 / 4 Risks - Exploitation likelihood is *Unlikely with Medium impact*.

Apply DevSecOps best-practices and use scanning tools to identify vulnerabilities in source- or byte-code, dependencies, container layers, and optionally also via dynamic scans against running test systems.

Medium: **Unnecessary Data Transfer: 3 / 3 Risks** - Exploitation likelihood is *Unlikely* with *Medium* impact.

Try to avoid sending or receiving sensitive data assets which are not required (i.e. neither processed or stored) by the involved technical asset.

Development

Elevated: **Cross-Site Request Forgery (CSRF): 4 / 4 Risks** - Exploitation likelihood is *Very Likely* with *Medium* impact.

Try to use anti-CSRF tokens or the double-submit patterns (at least for logged-in requests). When your authentication scheme depends on cookies (like session or token cookies), consider marking them with the same-site flag. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Elevated: **Cross-Site Scripting (XSS): 4 / 4 Risks** - Exploitation likelihood is *Likely* with *High* impact.

Try to encode all values sent back to the browser and also handle DOM-manipulations in a safe way to avoid DOM-based XSS. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Elevated: **Missing File Validation: 1 / 1 Risk** - Exploitation likelihood is *Very Likely* with *Medium* impact.

Filter by file extension and discard (if feasible) the name provided. Whitelist the accepted file types and determine the mime-type on the server-side (for example via "Apache Tika" or similar checks). If the file is retrievable by endusers and/or backoffice employees, consider performing scans for popular malware (if the files can be retrieved much later than they were uploaded, also apply a fresh malware scan during retrieval to scan with newer signatures of popular malware). Also enforce limits on maximum file size to avoid denial-of-service like scenarios.

Elevated: **Path-Traversal: 1 / 1 Risk** - Exploitation likelihood is *Likely* with *High* impact.

Before accessing the file cross-check that it resides in the expected folder and is of the expected type and filename/suffix. Try to use a mapping if possible instead of directly accessing by a filename which is (partly or fully) provided by the caller. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Elevated: **Server-Side Request Forgery (SSRF): 27 / 27 Risks** - Exploitation likelihood is *Likely* with *Medium* impact.

Try to avoid constructing the outgoing target URL with caller controllable values. Alternatively use a mapping (whitelist) when accessing outgoing URLs instead of creating them including caller controllable values. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Operations

Elevated: Missing Hardening: 2 / 2 Risks - Exploitation likelihood is *Likely* with *Medium* impact.

Try to apply all hardening best practices (like CIS benchmarks, OWASP recommendations, vendor recommendations, DevSec Hardening Framework, DBSAT for Oracle databases, and others).

Elevated: Missing Identity Provider Isolation: 2 / 2 Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.

Apply a network segmentation trust-boundary around the highly sensitive identity provider assets and their identity datastores.

Medium: Accidental Secret Leak: 3 / 3 Risks - Exploitation likelihood is *Unlikely* with *High* impact.

Establish measures preventing accidental check-in or package-in of secrets into sourcecode repositories and artifact registries. This starts by using good .gitignore and .dockerignore files, but does not stop there. See for example tools like "*git-secrets*" or "*Talisman*" to have check-in preventive measures for secrets. Consider also to regularly scan your repositories for secrets accidentally checked-in using scanning tools like "*gitLeaks*" or "*gitrob*".

Medium: Code Backdooring: 4 / 4 Risks - Exploitation likelihood is *Unlikely* with *High* impact.

Reduce the attack surface of backdooring the build pipeline by not directly exposing the build pipeline components on the public internet and also not exposing it in front of unmanaged (out-of-scope) developer clients. Also consider the use of code signing to prevent code modifications.

Medium: Container Base Image Backdooring: 9 / 9 Risks - Exploitation likelihood is *Unlikely* with *High* impact.

Apply hardening of all container infrastructures (see for example the *CIS-Benchmarks for Docker and Kubernetes* and the *Docker Bench for Security*). Use only trusted base images of the original vendors, verify digital signatures and apply image creation best practices. Also consider using Google's *Distroless* base images or otherwise very small base images. Regularly execute container image scans with tools checking the layers for vulnerable components.

Medium: Missing Network Segmentation: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

Apply a network segmentation trust-boundary around the highly sensitive assets and/or datastores.

Medium: Missing Web Application Firewall (WAF): 5 / 5 Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

Consider placing a Web Application Firewall (WAF) in front of the web-services and/or web-applications. For cloud environments many cloud providers offer pre-configured WAFs. Even reverse proxies can be enhanced by a WAF component via ModSecurity plugins.

Medium: **Unencrypted Technical Assets:** 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *High* impact.

Apply encryption to the technical asset.

RAA Analysis

For each technical asset the "**Relative Attacker Attractiveness**" (RAA) value was calculated in percent. The higher the RAA, the more interesting it is for an attacker to compromise the asset. The calculation algorithm takes the sensitivity ratings and quantities of stored and processed data into account as well as the communication links of the technical asset. Neighbouring assets to high-value RAA targets might receive an increase in their RAA value when they have a communication link towards that target ("Pivoting-Factor").

The following lists all technical assets sorted by their RAA value from highest (most attacker attractive) to lowest. This list can be used to prioritize on efforts relevant for the most attacker-attractive technical assets:

Technical asset paragraphs are clickable and link to the corresponding chapter.

API Server: RAA 100%

Argo CD API server. Accepts requests from the UI and CLI.

Host Cluster Kubernetes API: RAA 57%

Kubernetes API Server for the cluster Argo CD is deployed to

ApplicationSet Controller: RAA 49%

Some Description

User CLI: RAA 44%

User CLI

Application Controller: RAA 38%

Some Description

Internal Source Control Management API: RAA 33%

Source control manager (GitHub, GitLab, Helm repo, etc.) accessible only from the organization's network.

OIDC Proxy (Dex): RAA 30%

OIDC Proxy (Dex)

Repo Server: RAA 28%

Pulls from manifests sources, builds manifests, caches manifests

Quay: RAA 27%

Quay image repository.

Argo CD Build Pipeline (GitHub Actions): RAA 25%

Argo CD build pipeline, hosted on GitHub Actions.

OIDC Provider (External): RAA 23%
OIDC Provider (External)

External Cluster Kubernetes API: RAA 19%
Kubernetes API Server for a cluster Argo CD is managing

Rendered Manifests Cache (Redis): RAA 19%
Rendered Manifests Cache (Redis)

Internal Source Control Management UI: RAA 12%
Internal Source Control Management UI

Repo Server Storage: RAA 12%
Local (by default, ephemeral) storage for the repo-server.

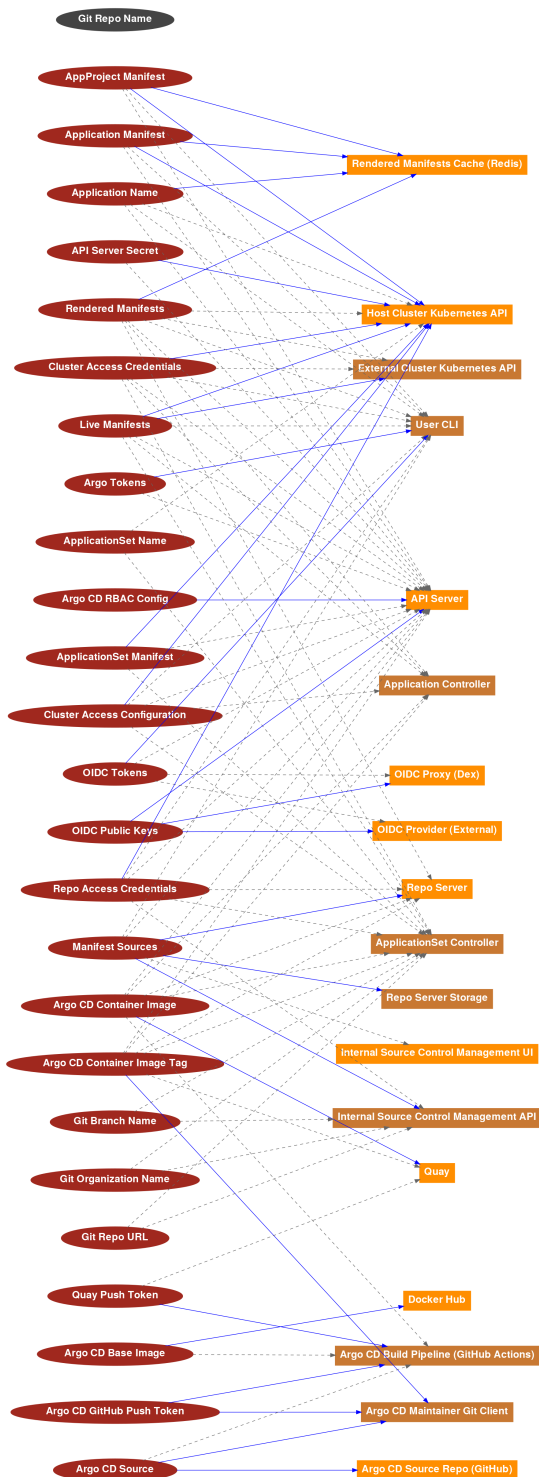
Argo CD Source Repo (GitHub): RAA 4%
GitHub repo holding the Argo CD source code.

Docker Hub: RAA 4%
Docker Hub image repository.

Argo CD Maintainer Git Client: RAA 0%
Git client (and configuration) used by an Argo CD maintainer.

Data Mapping

The following diagram was generated by Threagile based on the model input and gives a high-level distribution of data assets across technical assets. The color matches the identified data breach probability and risk level (see the "Data Breach Probabilities" chapter for more details). A solid line stands for *data is stored by the asset* and a dashed one means *data is processed by the asset*. For a full high-resolution version of this diagram please refer to the PNG image file alongside this report.



Out-of-Scope Assets: 0 Assets

This chapter lists all technical assets that have been defined as out-of-scope. Each one should be checked in the model whether it should better be included in the overall risk analysis:

Technical asset paragraphs are clickable and link to the corresponding chapter.

No technical assets have been defined as out-of-scope.

Potential Model Failures: 6 / 6 Risks

This chapter lists potential model failures where not all relevant assets have been modeled or the model might itself contain inconsistencies. Each potential model failure should be checked in the model against the architecture design:

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium: Missing Build Infrastructure: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

The modeled architecture does not contain a build infrastructure (devops-client, sourcecode-repo, build-pipeline, etc.), which might be the risk of a model missing critical assets (and thus not seeing their risks). If the architecture contains custom-developed parts, the pipeline where code gets developed and built needs to be part of the model.

Medium: Missing Identity Store: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

The modeled architecture does not contain an identity store, which might be the risk of a model missing critical assets (and thus not seeing their risks).

Medium: Missing Vault (Secret Storage): 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

In order to avoid the risk of secret leakage via config files (when attacked through vulnerabilities being able to read files like Path-Traversal and others), it is best practice to use a separate hardened process with proper authentication, authorization, and audit logging to access config secrets (like credentials, private keys, client certificates, etc.). This component is usually some kind of Vault.

Medium: Unnecessary Data Transfer: 3 / 3 Risks - Exploitation likelihood is *Unlikely* with *Medium* impact.

When a technical asset sends or receives data assets, which it neither processes or stores this is an indicator for unnecessarily transferred data (or for an incomplete model). When the unnecessarily transferred data assets are sensitive, this poses an unnecessary risk of an increased attack surface.

Questions: 1 / 2 Questions

This chapter lists custom questions that arose during the threat modeling process.

Some question with an answer?

Some answer

Some question without an answer?

- answer pending -

Identified Risks by Vulnerability Category

In total **88 potential risks** have been identified during the threat modeling process of which **0 are rated as critical, 0 as high, 27 as elevated, 57 as medium, and 4 as low.**

These risks are distributed across **21 vulnerability categories**. The following sub-chapters of this section describe each identified risk category.

Cross-Site Request Forgery (CSRF): 4 / 4 Risks

Description (Spoofing): [CWE 352](#)

When a web application is accessed via web protocols Cross-Site Request Forgery (CSRF) risks might arise.

Impact

If this risk remains unmitigated, attackers might be able to trick logged-in victim users into unwanted actions within the web application by visiting an attacker controlled web site.

Detection Logic

In-scope web applications accessed via typical web access protocols.

Risk Rating

The risk rating depends on the integrity rating of the data sent across the communication link.

False Positives

Web applications passing the authentication state via custom headers instead of cookies can eventually be false positives. Also when the web application is not accessed via a browser-like component (i.e not by a human user initiating the request that gets passed through all components until it reaches the web application) this can be considered a false positive.

Mitigation (Development): CSRF Prevention

Try to use anti-CSRF tokens or the double-submit patterns (at least for logged-in requests). When your authentication scheme depends on cookies (like session or token cookies), consider marking them with the same-site flag. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

ASVS Chapter: [V4 - Access Control Verification Requirements](#)

Cheat Sheet: [Cross-Site Request Forgery Prevention Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Cross-Site Request Forgery (CSRF)** was found **4 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Cross-Site Request Forgery (CSRF) risk at OIDC Provider (External) via Validate External OIDC Token from API Server: Exploitation likelihood is *Very Likely* with *Medium* impact.

`cross-site-request-forgery@oidc-provider@api-server>validate-external-oidc-token`

Unchecked

Cross-Site Request Forgery (CSRF) risk at API Server via Make Requests to API Server from User CLI: Exploitation likelihood is *Likely* with *Medium* impact.

`cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server`

Unchecked

Cross-Site Request Forgery (CSRF) risk at OIDC Proxy (Dex) via Validate Dex OIDC Token from API Server: Exploitation likelihood is *Likely* with *Medium* impact.

`cross-site-request-forgery@dex-server@api-server>validate-dex-oidc-token`

Unchecked

Medium Risk Severity

Cross-Site Request Forgery (CSRF) risk at OIDC Provider (External) via Proxying to an External OIDC Provider from OIDC Proxy (Dex): Exploitation likelihood is *Likely* with *Low* impact.

`cross-site-request-forgery@oidc-provider@dex-server>proxying-to-an-external-oidc-provider`

Unchecked

Cross-Site Scripting (XSS): 4 / 4 Risks

Description (Tampering): [CWE 79](#)

For each web application Cross-Site Scripting (XSS) risks might arise. In terms of the overall risk level take other applications running on the same domain into account as well.

Impact

If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Detection Logic

In-scope web applications.

Risk Rating

The risk rating depends on the sensitivity of the data processed or stored in the web application.

False Positives

When the technical asset is not accessed via a browser-like component (i.e not by a human user initiating the request that gets passed through all components until it reaches the web application) this can be considered a false positive.

Mitigation (Development): XSS Prevention

Try to encode all values sent back to the browser and also handle DOM-manipulations in a safe way to avoid DOM-based XSS. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

ASVS Chapter: [V5 - Validation, Sanitization and Encoding Verification Requirements](#)

Cheat Sheet: [Cross Site Scripting Prevention Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Cross-Site Scripting (XSS)** was found **4 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Cross-Site Scripting (XSS) risk at API Server: Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@api-server](#)

Unchecked

Cross-Site Scripting (XSS) risk at Internal Source Control Management UI: Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@internal-source-control-management-ui](#)

Unchecked

Cross-Site Scripting (XSS) risk at OIDC Provider (External): Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@oidc-provider](#)

Unchecked

Cross-Site Scripting (XSS) risk at OIDC Proxy (Dex): Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@dex-server](#)

Unchecked

Missing Authentication: 10 / 10 Risks

Description (Elevation of Privilege): [CWE 306](#)

Technical assets (especially multi-tenant systems) should authenticate incoming requests when the asset processes or stores sensitive data.

Impact

If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Detection Logic

In-scope technical assets (except load-balancer, reverse-proxy, service-registry, waf, ids, and ips and in-process calls) should authenticate incoming requests when the asset processes or stores sensitive data. This is especially the case for all multi-tenant assets (there even non-sensitive ones).

Risk Rating

The risk rating (medium or high) depends on the sensitivity of the data sent across the communication link. Monitoring callers are exempted from this risk.

False Positives

Technical assets which do not process requests regarding functionality or data linked to end-users (customers) can be considered as false positives after individual review.

Mitigation (Architecture): Authentication of Incoming Requests

Apply an authentication method to the technical asset. To protect highly sensitive data consider the use of two-factor authentication for human users.

ASVS Chapter: [V2 - Authentication Verification Requirements](#)

Cheat Sheet: [Authentication Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Authentication** was found **10 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Authentication covering communication link **Fetching Rendered Manifests from Cache from API Server to Repo Server**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server](#)

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from API Server to Quay**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@api-server>pull-argo-cd-image@api-server@quay](#)

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from Application Controller to Quay**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@application-controller>pull-argo-cd-image@application-controller@quay](#)

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from ApplicationSet Controller to Quay**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@applicationset-controller>pull-argo-cd-image@applicationset-controller@quay](#)

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from Repo Server to Quay**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@repo-server>pull-argo-cd-image@repo-server@quay](#)

Unchecked

Missing Authentication covering communication link **Pull Base Image from Docker Hub from Argo CD Build Pipeline (GitHub Actions) to Docker Hub**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@argo-cd-build-pipeline>pull-base-image-from-docker-hub@argo-cd-build-pipeline@docker-hub](#)

Unchecked

Missing Authentication covering communication link **Rendered Manifest Requests from Application Controller to Repo Server**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server](#)

Unchecked

Missing Authentication covering communication link **Send/Receive Cached Rendered Manifests from Repo Server to Rendered Manifests Cache (Redis)**: Exploitation likelihood is *Likely* with *High* impact.

missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache

Unchecked

Missing Authentication covering communication link **Validate Dex OIDC Token from API Server to OIDC Proxy (Dex)**: Exploitation likelihood is *Likely* with *High* impact.

missing-authentication@api-server>validate-dex-oidc-token@api-server@dex-server

Unchecked

Missing Authentication covering communication link **Validate External OIDC Token from API Server to OIDC Provider (External)**: Exploitation likelihood is *Likely* with *High* impact.

missing-authentication@api-server>validate-external-oidc-token@api-server@oidc-provider

Unchecked

Missing File Validation: 1 / 1 Risk

Description (Spoofing): [CWE 434](#)

When a technical asset accepts files, these input files should be strictly validated about filename and type.

Impact

If this risk is unmitigated, attackers might be able to provide malicious files to the application.

Detection Logic

In-scope technical assets with custom-developed code accepting file data formats.

Risk Rating

The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

False Positives

Fully trusted (i.e. cryptographically signed or similar) files can be considered as false positives after individual review.

Mitigation (Development): File Validation

Filter by file extension and discard (if feasible) the name provided. Whitelist the accepted file types and determine the mime-type on the server-side (for example via "Apache Tika" or similar checks). If the file is retrievable by endusers and/or backoffice employees, consider performing scans for popular malware (if the files can be retrieved much later than they were uploaded, also apply a fresh malware scan during retrieval to scan with newer signatures of popular malware). Also enforce limits on maximum file size to avoid denial-of-service like scenarios.

ASVS Chapter: [V12 - File and Resources Verification Requirements](#)

Cheat Sheet: [File Upload Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing File Validation** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing File Validation risk at **Repo Server**: Exploitation likelihood is *Very Likely* with *Medium* impact.

[missing-file-validation@repo-server](#)

Unchecked

Missing Hardening: 2 / 2 Risks

Description (Tampering): [CWE 16](#)

Technical assets with a Relative Attacker Attractiveness (RAA) value of 55 % or higher should be explicitly hardened taking best practices and vendor hardening guides into account.

Impact

If this risk remains unmitigated, attackers might be able to easier attack high-value targets.

Detection Logic

In-scope technical assets with RAA values of 55 % or higher. Generally for high-value targets like datastores, application servers, identity providers and ERP systems this limit is reduced to 40 %

Risk Rating

The risk rating depends on the sensitivity of the data processed or stored in the technical asset.

False Positives

Usually no false positives.

Mitigation (Operations): System Hardening

Try to apply all hardening best practices (like CIS benchmarks, OWASP recommendations, vendor recommendations, DevSec Hardening Framework, DBSAT for Oracle databases, and others).

ASVS Chapter: [V14 - Configuration Verification Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Hardening** was found **2 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Hardening risk at **API Server**: Exploitation likelihood is *Likely* with *Medium* impact.

[missing-hardening@api-server](#)

Unchecked

Missing Hardening risk at **Host Cluster Kubernetes API**: Exploitation likelihood is *Likely* with *Medium* impact.

[missing-hardening@host-cluster-kubernetes-api](#)

Unchecked

Missing Identity Provider Isolation: 2 / 2 Risks

Description (Elevation of Privilege): [CWE 1008](#)

Highly sensitive identity provider assets and their identity datastores should be isolated from other assets by their own network segmentation trust-boundary (execution-environment boundaries do not count as network isolation).

Impact

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards highly sensitive identity provider assets and their identity datastores, as they are not separated by network segmentation.

Detection Logic

In-scope identity provider assets and their identity datastores when surrounded by other (not identity-related) assets (without a network trust-boundary in-between). This risk is especially prevalent when other non-identity related assets are within the same execution environment (i.e. same database or same application server).

Risk Rating

Default is high impact. The impact is increased to very-high when the asset missing the trust-boundary protection is rated as strictly-confidential or mission-critical.

False Positives

When all assets within the network segmentation trust-boundary are hardened and protected to the same extend as if all were identity providers with data of highest sensitivity.

Mitigation (Operations): Network Segmentation

Apply a network segmentation trust-boundary around the highly sensitive identity provider assets and their identity datastores.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Identity Provider Isolation** was found **2 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Identity Provider Isolation to further encapsulate and protect identity-related asset **OIDC Provider (External)** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Very High* impact.

[missing-identity-provider-isolation@oidc-provider](#)

Unchecked

Missing Identity Provider Isolation to further encapsulate and protect identity-related asset **OIDC Proxy (Dex)** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Very High* impact.

[missing-identity-provider-isolation@dex-server](#)

Unchecked

Path-Traversal: 1 / 1 Risk

Description (Information Disclosure): [CWE 22](#)

When a filesystem is accessed Path-Traversal or Local-File-Inclusion (LFI) risks might arise. The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed or stored.

Impact

If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) from the filesystem of affected components.

Detection Logic

Filesystems accessed by in-scope callers.

Risk Rating

The risk rating depends on the sensitivity of the data stored inside the technical asset.

False Positives

File accesses by filenames not consisting of parts controllable by the caller can be considered as false positives after individual review.

Mitigation (Development): Path-Traversal Prevention

Before accessing the file cross-check that it resides in the expected folder and is of the expected type and filename/suffix. Try to use a mapping if possible instead of directly accessing by a filename which is (partly or fully) provided by the caller. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

ASVS Chapter: [V12 - File and Resources Verification Requirements](#)

Cheat Sheet: [Input Validation Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Path-Traversal** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Path-Traversal risk at **Repo Server** against filesystem **Repo Server Storage** via **Store Cached Manifest Sources**: Exploitation likelihood is *Likely* with *High* impact.

`path-traversal@repo-server@repo-server-storage@repo-server>store-cached-manifest-sources`

Unchecked

Server-Side Request Forgery (SSRF): 27 / 27 Risks

Description (Information Disclosure): [CWE 918](#)

When a server system (i.e. not a client) is accessing other server systems via typical web protocols Server-Side Request Forgery (SSRF) or Local-File-Inclusion (LFI) or Remote-File-Inclusion (RFI) risks might arise.

Impact

If this risk is unmitigated, attackers might be able to access sensitive services or files of network-reachable components by modifying outgoing calls of affected components.

Detection Logic

In-scope non-client systems accessing (using outgoing communication links) targets with either HTTP or HTTPS protocol.

Risk Rating

The risk rating (low or medium) depends on the sensitivity of the data assets receivable via web protocols from targets within the same network trust-boundary as well on the sensitivity of the data assets receivable via web protocols from the target asset itself. Also for cloud-based environments the exploitation impact is at least medium, as cloud backend services can be attacked via SSRF.

False Positives

Servers not sending outgoing web requests can be considered as false positives after review.

Mitigation (Development): SSRF Prevention

Try to avoid constructing the outgoing target URL with caller controllable values. Alternatively use a mapping (whitelist) when accessing outgoing URLs instead of creating them including caller controllable values. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

ASVS Chapter: [V12 - File and Resources Verification Requirements](#)

Cheat Sheet: [Server Side Request Forgery Prevention Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Server-Side Request Forgery (SSRF)** was found **27 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target OIDC Provider (External) via Validate External OIDC Token: Exploitation likelihood is *Likely with Medium impact*.

server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Unchecked

Medium Risk Severity

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target External Cluster Kubernetes API via Get/Update/Delete Live Resource State from Kubernetes (External): Exploitation likelihood is *Unlikely with Medium impact*.

server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Unchecked

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target Host Cluster Kubernetes API via Get/Update/Delete Live Resource State from Kubernetes (Host): Exploitation likelihood is *Unlikely with Medium impact*.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Unchecked

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target Host Cluster Kubernetes API via Update Cluster Access Config: Exploitation likelihood is *Unlikely with Medium impact*.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Unchecked

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target Host Cluster Kubernetes API via Update RBAC Config: Exploitation likelihood is *Unlikely with Medium impact*.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Unchecked

Server-Side Request Forgery (SSRF) risk at API Server server-side web-requesting the target Host Cluster Kubernetes API via Update Repo Access Credentials: Exploitation likelihood is *Unlikely with Medium impact*.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **OIDC Proxy (Dex)** via **Validate Dex OIDC Token**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Repo Server** via **Fetching Rendered Manifests from Cache**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **External Cluster Kubernetes API** via **Reconcile Resource State (External Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Host Cluster Kubernetes API** via **Reconcile Resource State (Host Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Repo Server** via **Rendered Manifest Requests**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Unchecked

Server-Side Request Forgery (SSRF) risk at **ApplicationSet Controller** server-side web-requesting the target **Host Cluster Kubernetes API** via **Reconcile Resource State (Host Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Unchecked

Server-Side Request Forgery (SSRF) risk at ApplicationSet Controller server-side web-requesting the target Internal Source Control Management API via Git Generator Pull: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Unchecked

Server-Side Request Forgery (SSRF) risk at ApplicationSet Controller server-side web-requesting the target Quay via Pull Argo CD Image: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at Argo CD Build Pipeline (GitHub Actions) server-side web-requesting the target Argo CD Source Repo (GitHub) via Pull Source: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Unchecked

Server-Side Request Forgery (SSRF) risk at Argo CD Build Pipeline (GitHub Actions) server-side web-requesting the target Docker Hub via Pull Base Image from Docker Hub: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Unchecked

Server-Side Request Forgery (SSRF) risk at Argo CD Build Pipeline (GitHub Actions) server-side web-requesting the target Quay via Push Image to Quay: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Unchecked

Server-Side Request Forgery (SSRF) risk at Argo CD Maintainer Git Client server-side web-requesting the target Argo CD Source Repo (GitHub) via Push Code/Tags to GitHub: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Unchecked

Server-Side Request Forgery (SSRF) risk at OIDC Proxy (Dex) server-side web-requesting the target OIDC Provider (External) via Proxying to an External OIDC Provider: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Unchecked

Server-Side Request Forgery (SSRF) risk at Repo Server server-side web-requesting the target Host Cluster Kubernetes API via Get Repo Access Credentials: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Internal Source Control Management API** via **Fetch Manifest Sources**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Rendered Manifests Cache (Redis)** via **Send/Receive Cached Rendered Manifests**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Unchecked

Server-Side Request Forgery (SSRF) risk at **User CLI** server-side web-requesting the target **API Server** via **Make Requests to API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Unchecked

Low Risk Severity

Server-Side Request Forgery (SSRF) risk at **Internal Source Control Management UI** server-side web-requesting the target **Internal Source Control Management API** via **Push Manifest Sources**: Exploitation likelihood is *Unlikely* with *Low* impact.

server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources

Unchecked

Unguarded Access From Internet: 3 / 3 Risks

Description (Elevation of Privilege): [CWE 501](#)

Internet-exposed assets must be guarded by a protecting service, application, or reverse-proxy.

Impact

If this risk is unmitigated, attackers might be able to directly attack sensitive systems without any hardening components in-between due to them being directly exposed on the internet.

Detection Logic

In-scope technical assets (excluding load-balancer) with confidentiality rating of confidential (or higher) or with integrity rating of critical (or higher) when accessed directly from the internet. All web-server, web-application, reverse-proxy, waf, and gateway assets are exempted from this risk when they do not consist of custom developed code and the data-flow only consists of HTTP or FTP protocols. Access from monitoring systems as well as VPN-protected connections are exempted.

Risk Rating

The matching technical assets are at low risk. When either the confidentiality rating is strictly-confidential or the integrity rating is mission-critical, the risk-rating is considered medium. For assets with RAA values higher than 40 % the risk-rating increases.

False Positives

When other means of filtering client requests are applied equivalent of reverse-proxy, waf, or gateway components.

Mitigation (Architecture): Encapsulation of Technical Asset

Encapsulate the asset behind a guarding service, application, or reverse-proxy. For admin maintenance a bastion-host should be used as a jump-server. For file transfer a store-and-forward-host should be used as an indirect file exchange platform.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Unguarded Access From Internet** was found **3 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Unguarded Access from Internet of Argo CD Source Repo (GitHub) by Argo CD Build Pipeline (GitHub Actions) via Pull Source: Exploitation likelihood is *Very Likely* with *Medium* impact.

`unguarded-access-from-internet@argo-cd-source-repo@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-source`

Unchecked

Unguarded Access from Internet of Docker Hub by Argo CD Build Pipeline (GitHub Actions) via Pull Base Image from Docker Hub: Exploitation likelihood is *Very Likely* with *Medium* impact.

`unguarded-access-from-internet@docker-hub@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-base-image-from-docker-hub`

Unchecked

Unguarded Access from Internet of Quay by Argo CD Build Pipeline (GitHub Actions) via Push Image to Quay: Exploitation likelihood is *Very Likely* with *Medium* impact.

`unguarded-access-from-internet@quay@argo-cd-build-pipeline@argo-cd-build-pipeline>push-image-to-quay`

Unchecked

Accidental Secret Leak: 3 / 3 Risks

Description (Information Disclosure): [CWE 200](#)

Sourcecode repositories (including their histories) as well as artifact registries can accidentally contain secrets like checked-in or packaged-in passwords, API tokens, certificates, crypto keys, etc.

Impact

If this risk is unmitigated, attackers which have access to affected sourcecode repositories or artifact registries might find secrets accidentally checked-in.

Detection Logic

In-scope sourcecode repositories and artifact registries.

Risk Rating

The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

False Positives

Usually no false positives.

Mitigation (Operations): Build Pipeline Hardening

Establish measures preventing accidental check-in or package-in of secrets into sourcecode repositories and artifact registries. This starts by using good `.gitignore` and `.dockerignore` files, but does not stop there. See for example tools like *"git-secrets"* or *"Talisman"* to have check-in preventive measures for secrets. Consider also to regularly scan your repositories for secrets accidentally checked-in using scanning tools like *"gitleaks"* or *"gitrob"*.

ASVS Chapter: [V14 - Configuration Verification Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Accidental Secret Leak** was found **3 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Accidental Secret Leak risk at **Argo CD Source Repo (GitHub)**: Exploitation likelihood is *Unlikely with High* impact.

[accidental-secret-leak@argo-cd-source-repo](#)

Unchecked

Accidental Secret Leak risk at **Docker Hub**: Exploitation likelihood is *Unlikely with High* impact.

[accidental-secret-leak@docker-hub](#)

Unchecked

Accidental Secret Leak risk at **Quay**: Exploitation likelihood is *Unlikely with High* impact.

[accidental-secret-leak@quay](#)

Unchecked

Code Backdooring: 4 / 4 Risks

Description (Tampering): [CWE 912](#)

For each build-pipeline component Code Backdooring risks might arise where attackers compromise the build-pipeline in order to let backdoored artifacts be shipped into production. Aside from direct code backdooring this includes backdooring of dependencies and even of more lower-level build infrastructure, like backdooring compilers (similar to what the XcodeGhost malware did) or dependencies.

Impact

If this risk remains unmitigated, attackers might be able to execute code on and completely takeover production environments.

Detection Logic

In-scope development relevant technical assets which are either accessed by out-of-scope unmanaged developer clients and/or are directly accessed by any kind of internet-located (non-VPN) component or are themselves directly located on the internet.

Risk Rating

The risk rating depends on the confidentiality and integrity rating of the code being handled and deployed as well as the placement/calling of this technical asset on/from the internet.

False Positives

When the build-pipeline and sourcecode-repo is not exposed to the internet and considered fully trusted (which implies that all accessing clients are also considered fully trusted in terms of their patch management and applied hardening, which must be equivalent to a managed developer client environment) this can be considered a false positive after individual review.

Mitigation (Operations): Build Pipeline Hardening

Reduce the attack surface of backdooring the build pipeline by not directly exposing the build pipeline components on the public internet and also not exposing it in front of unmanaged (out-of-scope) developer clients. Also consider the use of code signing to prevent code modifications.

ASVS Chapter: [V10 - Malicious Code Verification Requirements](#)

Cheat Sheet: [Vulnerable Dependency Management Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Code Backdooring** was found **4 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Code Backdooring risk at **Argo CD Build Pipeline (GitHub Actions)**: Exploitation likelihood is *Unlikely* with *High* impact.

[code-backdooring@argo-cd-build-pipeline](#)

Unchecked

Code Backdooring risk at **Argo CD Source Repo (GitHub)**: Exploitation likelihood is *Unlikely* with *High* impact.

[code-backdooring@argo-cd-source-repo](#)

Unchecked

Code Backdooring risk at **Docker Hub**: Exploitation likelihood is *Unlikely* with *High* impact.

[code-backdooring@docker-hub](#)

Unchecked

Code Backdooring risk at **Quay**: Exploitation likelihood is *Unlikely* with *High* impact.

[code-backdooring@quay](#)

Unchecked

Container Base Image Backdooring: 9 / 9 Risks

Description (Tampering): [CWE 912](#)

When a technical asset is built using container technologies, Base Image Backdooring risks might arise where base images and other layers used contain vulnerable components or backdoors.

See for example:

<https://techcrunch.com/2018/06/15/tainted-crypto-mining-containers-pulled-from-docker-hub/>

Impact

If this risk is unmitigated, attackers might be able to deeply persist in the target system by executing code in deployed containers.

Detection Logic

In-scope technical assets running as containers.

Risk Rating

The risk rating depends on the sensitivity of the technical asset itself and of the data assets.

False Positives

Fully trusted (i.e. reviewed and cryptographically signed or similar) base images of containers can be considered as false positives after individual review.

Mitigation (Operations): Container Infrastructure Hardening

Apply hardening of all container infrastructures (see for example the *CIS-Benchmarks for Docker and Kubernetes* and the *Docker Bench for Security*). Use only trusted base images of the original vendors, verify digital signatures and apply image creation best practices. Also consider using Google's *Distroless* base images or otherwise very small base images. Regularly execute container image scans with tools checking the layers for vulnerable components.

ASVS Chapter: [V10 - Malicious Code Verification Requirements](#)

Cheat Sheet: [Docker Security Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS/CSVS applied?

Risk Findings

The risk **Container Base Image Backdooring** was found **9 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Container Base Image Backdooring risk at **API Server**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@api-server](#)

Unchecked

Container Base Image Backdooring risk at **Application Controller**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@application-controller](#)

Unchecked

Container Base Image Backdooring risk at **ApplicationSet Controller**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@applicationset-controller](#)

Unchecked

Container Base Image Backdooring risk at **External Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@external-cluster-kubernetes-api](#)

Unchecked

Container Base Image Backdooring risk at **Host Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@host-cluster-kubernetes-api](#)

Unchecked

Container Base Image Backdooring risk at **OIDC Proxy (Dex)**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@dex-server](#)

Unchecked

Container Base Image Backdooring risk at **Rendered Manifests Cache (Redis)**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@rendered-manifests-cache](#)

Unchecked

Container Base Image Backdooring risk at **Repo Server Storage**: Exploitation likelihood is *Unlikely* with *High* impact.

container-baseimage-backdooring@repo-server-storage

Unchecked

Container Base Image Backdooring risk at **Repo Server**: Exploitation likelihood is *Unlikely* with *High* impact.

container-baseimage-backdooring@repo-server

Unchecked

Missing Build Infrastructure: 1 / 1 Risk

Description (Tampering): [CWE 1127](#)

The modeled architecture does not contain a build infrastructure (devops-client, sourcecode-repo, build-pipeline, etc.), which might be the risk of a model missing critical assets (and thus not seeing their risks). If the architecture contains custom-developed parts, the pipeline where code gets developed and built needs to be part of the model.

Impact

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model due to critical build infrastructure components missing in the model.

Detection Logic

Models with in-scope custom-developed parts missing in-scope development (code creation) and build infrastructure components (devops-client, sourcecode-repo, build-pipeline, etc.).

Risk Rating

The risk rating depends on the highest sensitivity of the in-scope assets running custom-developed parts.

False Positives

Models not having any custom-developed parts can be considered as false positives after individual review.

Mitigation (Architecture): Build Pipeline Hardening

Include the build infrastructure in the model.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Build Infrastructure** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Build Infrastructure in the threat model (referencing asset **Application Controller** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-build-infrastructure@application-controller](#)

Unchecked

Missing Identity Store: 1 / 1 Risk

Description (Spoofing): [CWE 287](#)

The modeled architecture does not contain an identity store, which might be the risk of a model missing critical assets (and thus not seeing their risks).

Impact

If this risk is unmitigated, attackers might be able to exploit risks unseen in this threat model in the identity provider/store that is currently missing in the model.

Detection Logic

Models with authenticated data-flows authorized via enduser-identity missing an in-scope identity store.

Risk Rating

The risk rating depends on the sensitivity of the enduser-identity authorized technical assets and their data assets processed and stored.

False Positives

Models only offering data/services without any real authentication need can be considered as false positives after individual review.

Mitigation (Architecture): Identity Store

Include an identity store in the model if the application has a login.

ASVS Chapter: [V2 - Authentication Verification Requirements](#)

Cheat Sheet: [Authentication Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Identity Store** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Identity Store in the threat model (referencing asset **Argo CD Maintainer Git Client** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-identity-store@argo-cd-maintainer-git-client](#)

Unchecked

Missing Network Segmentation: 1 / 1 Risk

Description (Elevation of Privilege): [CWE 1008](#)

Highly sensitive assets and/or datastores residing in the same network segment than other lower sensitive assets (like webserver or content management systems etc.) should be better protected by a network segmentation trust-boundary.

Impact

If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards more valuable targets, as they are not separated by network segmentation.

Detection Logic

In-scope technical assets with high sensitivity and RAA values as well as datastores when surrounded by assets (without a network trust-boundary in-between) which are of type client-system, web-server, web-application, cms, web-service-rest, web-service-soap, build-pipeline, sourcecode-repository, monitoring, or similar and there is no direct connection between these (hence no requirement to be so close to each other).

Risk Rating

Default is low risk. The risk is increased to medium when the asset missing the trust-boundary protection is rated as strictly-confidential or mission-critical.

False Positives

When all assets within the network segmentation trust-boundary are hardened and protected to the same extend as if all were containing/processing highly sensitive data.

Mitigation (Operations): Network Segmentation

Apply a network segmentation trust-boundary around the highly sensitive assets and/or datastores.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Network Segmentation** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Network Segmentation to further encapsulate and protect **API Server** against unrelated lower protected assets in the same network segment, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-network-segmentation@api-server](#)

Unchecked

Missing Two-Factor Authentication (2FA): 1 / 1 Risk

Description (Elevation of Privilege): [CWE 308](#)

Technical assets (especially multi-tenant systems) should authenticate incoming requests with two-factor (2FA) authentication when the asset processes or stores highly sensitive data (in terms of confidentiality, integrity, and availability) and is accessed by humans.

Impact

If this risk is unmitigated, attackers might be able to access or modify highly sensitive data without strong authentication.

Detection Logic

In-scope technical assets (except load-balancer, reverse-proxy, waf, ids, and ips) should authenticate incoming requests via two-factor authentication (2FA) when the asset processes or stores highly sensitive data (in terms of confidentiality, integrity, and availability) and is accessed by a client used by a human user.

Risk Rating

medium

False Positives

Technical assets which do not process requests regarding functionality or data linked to end-users (customers) can be considered as false positives after individual review.

Mitigation (Business Side): Authentication with Second Factor (2FA)

Apply an authentication method to the technical asset protecting highly sensitive data via two-factor authentication for human users.

ASVS Chapter: [V2 - Authentication Verification Requirements](#)

Cheat Sheet: [Multifactor Authentication Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Missing Two-Factor Authentication (2FA)** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Two-Factor Authentication covering communication link **Make Requests to API Server from User CLI to API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server](#)

Unchecked

Missing Vault (Secret Storage): 1 / 1 Risk

Description (Information Disclosure): [CWE 522](#)

In order to avoid the risk of secret leakage via config files (when attacked through vulnerabilities being able to read files like Path-Traversal and others), it is best practice to use a separate hardened process with proper authentication, authorization, and audit logging to access config secrets (like credentials, private keys, client certificates, etc.). This component is usually some kind of Vault.

Impact

If this risk is unmitigated, attackers might be able to easier steal config secrets (like credentials, private keys, client certificates, etc.) once a vulnerability to access files is present and exploited.

Detection Logic

Models without a Vault (Secret Storage).

Risk Rating

The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

False Positives

Models where no technical assets have any kind of sensitive config data to protect can be considered as false positives after individual review.

Mitigation (Architecture): Vault (Secret Storage)

Consider using a Vault (Secret Storage) to securely store and access config secrets (like credentials, private keys, client certificates, etc.).

ASVS Chapter: [V6 - Stored Cryptography Verification Requirements](#)

Cheat Sheet: [Cryptographic Storage Cheat Sheet](#)

Check

Is a Vault (Secret Storage) in place?

Risk Findings

The risk **Missing Vault (Secret Storage)** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Vault (Secret Storage) in the threat model (referencing asset **Internal Source Control Management API** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-vault@internal-source-control-management-api](#)

Unchecked

Missing Web Application Firewall (WAF): 5 / 5 Risks

Description (Tampering): [CWE 1008](#)

To have a first line of filtering defense, security architectures with web-services or web-applications should include a WAF in front of them. Even though a WAF is not a replacement for security (all components must be secure even without a WAF) it adds another layer of defense to the overall system by delaying some attacks and having easier attack alerting through it.

Impact

If this risk is unmitigated, attackers might be able to apply standard attack pattern tests at great speed without any filtering.

Detection Logic

In-scope web-services and/or web-applications accessed across a network trust boundary not having a Web Application Firewall (WAF) in front of them.

Risk Rating

The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

False Positives

Targets only accessible via WAFs or reverse proxies containing a WAF component (like ModSecurity) can be considered as false positives after individual review.

Mitigation (Operations): Web Application Firewall (WAF)

Consider placing a Web Application Firewall (WAF) in front of the web-services and/or web-applications. For cloud environments many cloud providers offer pre-configured WAFs. Even reverse proxies can be enhanced by a WAF component via ModSecurity plugins.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Virtual Patching Cheat Sheet](#)

Check

Is a Web Application Firewall (WAF) in place?

Risk Findings

The risk **Missing Web Application Firewall (WAF)** was found **5 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Web Application Firewall (WAF) risk at API Server: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@api-server](#)

Unchecked

Missing Web Application Firewall (WAF) risk at External Cluster Kubernetes API: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@external-cluster-kubernetes-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at Host Cluster Kubernetes API: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@host-cluster-kubernetes-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at Internal Source Control Management API: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@internal-source-control-management-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at OIDC Provider (External): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@oidc-provider](#)

Unchecked

Unchecked Deployment: 4 / 4 Risks

Description (Tampering): [CWE 1127](#)

For each build-pipeline component Unchecked Deployment risks might arise when the build-pipeline does not include established DevSecOps best-practices. DevSecOps best-practices scan as part of CI/CD pipelines for vulnerabilities in source- or byte-code, dependencies, container layers, and dynamically against running test systems. There are several open-source and commercial tools existing in the categories DAST, SAST, and IAST.

Impact

If this risk remains unmitigated, vulnerabilities in custom-developed software or their dependencies might not be identified during continuous deployment cycles.

Detection Logic

All development-relevant technical assets.

Risk Rating

The risk rating depends on the highest rating of the technical assets and data assets processed by deployment-receiving targets.

False Positives

When the build-pipeline does not build any software components it can be considered a false positive after individual review.

Mitigation (Architecture): Build Pipeline Hardening

Apply DevSecOps best-practices and use scanning tools to identify vulnerabilities in source- or byte-code, dependencies, container layers, and optionally also via dynamic scans against running test systems.

ASVS Chapter: [V14 - Configuration Verification Requirements](#)

Cheat Sheet: [Vulnerable Dependency Management Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Unchecked Deployment** was found **4 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Unchecked Deployment risk at **Argo CD Build Pipeline (GitHub Actions)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[unchecked-deployment@argo-cd-build-pipeline](#)

Unchecked

Low Risk Severity

Unchecked Deployment risk at **Argo CD Source Repo (GitHub)**: Exploitation likelihood is *Unlikely* with *Low* impact.

[unchecked-deployment@argo-cd-source-repo](#)

Unchecked

Unchecked Deployment risk at **Docker Hub**: Exploitation likelihood is *Unlikely* with *Low* impact.

[unchecked-deployment@docker-hub](#)

Unchecked

Unchecked Deployment risk at **Quay**: Exploitation likelihood is *Unlikely* with *Low* impact.

[unchecked-deployment@quay](#)

Unchecked

Unencrypted Technical Assets: 1 / 1 Risk

Description (Information Disclosure): [CWE 311](#)

Due to the confidentiality rating of the technical asset itself and/or the processed data assets this technical asset must be encrypted. The risk rating depends on the sensitivity technical asset itself and of the data assets stored.

Impact

If this risk is unmitigated, attackers might be able to access unencrypted data when successfully compromising sensitive components.

Detection Logic

In-scope unencrypted technical assets (excluding reverse-proxy, load-balancer, waf, ids, ips and embedded components like library) storing data assets rated at least as confidential or critical. For technical assets storing data assets rated as strictly-confidential or mission-critical the encryption must be of type data-with-enduser-individual-key.

Risk Rating

Depending on the confidentiality rating of the stored data-assets either medium or high risk.

False Positives

When all sensitive data stored within the asset is already fully encrypted on document or data level.

Mitigation (Operations): Encryption of Technical Asset

Apply encryption to the technical asset.

ASVS Chapter: [V6 - Stored Cryptography Verification Requirements](#)

Cheat Sheet: [Cryptographic Storage Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Unencrypted Technical Assets** was found **1 time** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Unencrypted Technical Asset named **Repo Server Storage** missing enduser-individual encryption with data-with-enduser-individual-key: Exploitation likelihood is *Unlikely* with *High* impact.

[unencrypted-asset@repo-server-storage](#)

Unchecked

Unnecessary Data Transfer: 3 / 3 Risks

Description (Elevation of Privilege): [CWE 1008](#)

When a technical asset sends or receives data assets, which it neither processes or stores this is an indicator for unnecessarily transferred data (or for an incomplete model). When the unnecessarily transferred data assets are sensitive, this poses an unnecessary risk of an increased attack surface.

Impact

If this risk is unmitigated, attackers might be able to target unnecessarily transferred data.

Detection Logic

In-scope technical assets sending or receiving sensitive data assets which are neither processed nor stored by the technical asset are flagged with this risk. The risk rating (low or medium) depends on the confidentiality, integrity, and availability rating of the technical asset. Monitoring data is exempted from this risk.

Risk Rating

The risk assessment is depending on the confidentiality and integrity rating of the transferred data asset either low or medium.

False Positives

Technical assets missing the model entries of either processing or storing the mentioned data assets can be considered as false positives (incomplete models) after individual review. These should then be addressed by completing the model so that all necessary data assets are processed and/or stored by the technical asset involved.

Mitigation (Architecture): Attack Surface Reduction

Try to avoid sending or receiving sensitive data assets which are not required (i.e. neither processed or stored) by the involved technical asset.

ASVS Chapter: [V1 - Architecture, Design and Threat Modeling Requirements](#)

Cheat Sheet: [Attack Surface Analysis Cheat Sheet](#)

Check

Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Risk Findings

The risk **Unnecessary Data Transfer** was found **3 times** in the analyzed architecture to be potentially possible. Each spot should be checked individually by reviewing the implementation whether all controls have been applied properly in order to mitigate each risk.

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Unnecessary Data Transfer of Argo CD Container Image Tag data at Argo CD Source Repo (GitHub) from/to Argo CD Maintainer Git Client: Exploitation likelihood is *Unlikely* with *Medium* impact.

[unnecessary-data-transfer@argo-cd-container-image-tag@argo-cd-source-repo@argo-cd-maintainer-git-client](#)

Unchecked

Unnecessary Data Transfer of Argo CD GitHub Push Token data at Argo CD Source Repo (GitHub) from/to Argo CD Maintainer Git Client: Exploitation likelihood is *Unlikely* with *Medium* impact.

[unnecessary-data-transfer@argo-cd-github-push-token@argo-cd-source-repo@argo-cd-maintainer-git-client](#)

Unchecked

Unnecessary Data Transfer of Argo CD RBAC Config data at Host Cluster Kubernetes API from/to API Server: Exploitation likelihood is *Unlikely* with *Medium* impact.

[unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server](#)

Unchecked

Identified Risks by Technical Asset

In total **88 potential risks** have been identified during the threat modeling process of which **0 are rated as critical, 0 as high, 27 as elevated, 57 as medium, and 4 as low.**

These risks are distributed across **18 in-scope technical assets**. The following sub-chapters of this section describe each identified risk grouped by technical asset. The RAA value of a technical asset is the calculated "Relative Attacker Attractiveness" value in percent.

API Server: 16 / 16 Risks

Description

Argo CD API server. Accepts requests from the UI and CLI.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Cross-Site Scripting (XSS) risk at **API Server**: Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@api-server](#)

Unchecked

Cross-Site Request Forgery (CSRF) risk at **API Server** via **Make Requests to API Server** from **User CLI**: Exploitation likelihood is *Likely* with *Medium* impact.

[cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server](#)

Unchecked

Missing Hardening risk at **API Server**: Exploitation likelihood is *Likely* with *Medium* impact.

[missing-hardening@api-server](#)

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **OIDC Provider (External)** via **Validate External OIDC Token**: Exploitation likelihood is *Likely* with *Medium* impact.

[server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token](#)

Unchecked

Medium Risk Severity

Container Base Image Backdooring risk at **API Server**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@api-server](#)

Unchecked

Missing Network Segmentation to further encapsulate and protect **API Server** against unrelated lower protected assets in the same network segment, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-network-segmentation@api-server](#)

Unchecked

Missing Two-Factor Authentication covering communication link **Make Requests to API Server** from **User CLI** to **API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Unchecked

Missing Web Application Firewall (WAF) risk at **API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

missing-waf@api-server

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **External Cluster Kubernetes API** via **Get/Update/Delete Live Resource State from Kubernetes (External)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Host Cluster Kubernetes API** via **Get/Update/Delete Live Resource State from Kubernetes (Host)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Host Cluster Kubernetes API** via **Update Cluster Access Config**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Host Cluster Kubernetes API** via **Update RBAC Config**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Host Cluster Kubernetes API** via **Update Repo Access Credentials**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **OIDC Proxy (Dex)** via **Validate Dex OIDC Token**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at **API Server** server-side web-requesting the target **Repo Server** via **Fetching Rendered Manifests from Cache**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Unchecked

Asset Information

ID:	api-server
Type:	process
Usage:	devops
RAA:	100 %
Size:	component
Technology:	application-server
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	true
Custom-Developed:	false
Client by Human:	false
Data Processed:	API Server Secret, AppProject Manifest, Application Manifest, Application Name, ApplicationSet Manifest, Argo CD Container Image, Argo CD Container Image Tag, Argo CD RBAC Config, Argo Tokens, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Manifest Sources, OIDC Public Keys, OIDC Tokens, Rendered Manifests, Repo Access Credentials
Data Stored:	Argo CD RBAC Config, OIDC Public Keys
Formats Accepted:	CSV, JSON

Asset Rating

Owner:	Argo CD Operator
Confidentiality:	confidential (rated 4 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)

Availability: operational (rated 2 in scale of 5)
CIA-Justification:

Outgoing Communication Links: 9

Target technical asset names are clickable and link to the corresponding chapter.

Validate External OIDC Token (outgoing)

Get public keys from OIDC provider to validate tokens.

Target:	OIDC Provider (External)
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	business
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	none
Data Received:	OIDC Public Keys

Validate Dex OIDC Token (outgoing)

Get public keys from Dex server to validate tokens.

Target:	OIDC Proxy (Dex)
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	true
Data Sent:	none
Data Received:	OIDC Public Keys

Update Repo Access Credentials (outgoing)

Write changes from the UI/CLI/API to repo secrets.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Repo Access Credentials
Data Received:	Repo Access Credentials

Update RBAC Config (outgoing)

Write changes from the UI/CLI/API to RBAC config.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Argo CD RBAC Config
Data Received:	Argo CD RBAC Config

Update Cluster Access Config (outgoing)

Write changes from the UI/CLI/API to cluster secrets.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false

Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Cluster Access Configuration, Cluster Access Credentials
Data Received:	Cluster Access Configuration, Cluster Access Credentials

Pull Argo CD Image (outgoing)

Pull the Argo CD container image from Quay.

Target:	Quay
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Argo CD Container Image Tag
Data Received:	Argo CD Container Image

Get/Update/Delete Live Resource State from Kubernetes (Host) (outgoing)

Get the live state of an Argo CD-managed resource, or potentially update or delete a resource.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Application Name, Argo CD RBAC Config, Cluster Access Configuration, Cluster Access Credentials, Rendered Manifests, Repo Access Credentials
Data Received:	API Server Secret, Argo CD RBAC Config, Cluster Access Configuration,

Cluster Access Credentials, Live Manifests, Repo Access Credentials

Get/Update/Delete Live Resource State from Kubernetes (External) (outgoing)

Get the live state of an Argo CD-managed resource, or potentially update or delete a resource on an external cluster.

Target:	External Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Application Name, Cluster Access Credentials, Rendered Manifests
Data Received:	Live Manifests

Fetching Rendered Manifests from Cache (outgoing)

Fetch manifests from the repo server to display via UI or CLI.

Target:	Repo Server
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	true
Data Sent:	Application Name
Data Received:	Rendered Manifests

Incoming Communication Links: 1

Source technical asset names are clickable and link to the corresponding chapter.

Make Requests to API Server (incoming)

Make requests to the API server.

Source:	User CLI
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	enduser-identity-propagation
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	AppProject Manifest, Application Manifest, Application Name, Cluster Access Configuration, Cluster Access Credentials, Manifest Sources, OIDC Tokens, Rendered Manifests, Repo Access Credentials
Data Sent:	Argo Tokens

Argo CD Source Repo (GitHub): 6 / 6 Risks

Description

GitHub repo holding the Argo CD source code.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Unguarded Access from Internet of Argo CD Source Repo (GitHub) by Argo CD Build Pipeline (GitHub Actions) via Pull Source: Exploitation likelihood is *Very Likely* with *Medium* impact.

`unguarded-access-from-internet@argo-cd-source-repo@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-source`

Unchecked

Medium Risk Severity

Accidental Secret Leak risk at Argo CD Source Repo (GitHub): Exploitation likelihood is *Unlikely* with *High* impact.

`accidental-secret-leak@argo-cd-source-repo`

Unchecked

Code Backdooring risk at Argo CD Source Repo (GitHub): Exploitation likelihood is *Unlikely* with *High* impact.

`code-backdooring@argo-cd-source-repo`

Unchecked

Unnecessary Data Transfer of Argo CD Container Image Tag data at Argo CD Source Repo (GitHub) from/to Argo CD Maintainer Git Client: Exploitation likelihood is *Unlikely* with *Medium* impact.

`unnecessary-data-transfer@argo-cd-container-image-tag@argo-cd-source-repo@argo-cd-maintainer-git-client`

Unchecked

Unnecessary Data Transfer of Argo CD GitHub Push Token data at Argo CD Source Repo (GitHub) from/to Argo CD Maintainer Git Client: Exploitation likelihood is *Unlikely* with *Medium* impact.

`unnecessary-data-transfer@argo-cd-github-push-token@argo-cd-source-repo@argo-cd-maintainer-git-client`

Unchecked

Low Risk Severity

Unchecked Deployment risk at **Argo CD Source Repo (GitHub)**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@argo-cd-source-repo

Unchecked

Asset Information

ID:	argo-cd-source-repo
Type:	datastore
Usage:	devops
RAA:	4 %
Size:	application
Technology:	sourcecode-repository
Tags:	none
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Argo CD Source
Data Stored:	Argo CD Source
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	GitHub	
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Incoming Communication Links: 2

Source technical asset names are clickable and link to the corresponding chapter.

Push Code/Tags to GitHub (incoming)

Push code to the Argo CD repo (as when cherry-picking changes) and/or push tags (as when cutting a release).

Source:	Argo CD Maintainer Git Client
Protocol:	https
Encrypted:	true
Authentication:	two-factor
Authorization:	enduser-identity-propagation
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD Container Image Tag, Argo CD GitHub Push Token, Argo CD Source
Data Sent:	Argo CD Container Image Tag, Argo CD Source

Pull Source (incoming)

Pull the Argo CD source from the GitHub repo.

Source:	Argo CD Build Pipeline (GitHub Actions)
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	none
Data Sent:	Argo CD Source

Docker Hub: 5 / 5 Risks

Description

Docker Hub image repository.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Authentication covering communication link **Pull Base Image from Docker Hub from Argo CD Build Pipeline (GitHub Actions) to Docker Hub**: Exploitation likelihood is *Likely* with *High* impact.

missing-authentication@argo-cd-build-pipeline>pull-base-image-from-docker-hub@argo-cd-build-pipeline@docker-hub

Unchecked

Unguarded Access from Internet of Docker Hub by Argo CD Build Pipeline (GitHub Actions) via Pull Base Image from Docker Hub: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@docker-hub@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Unchecked

Medium Risk Severity

Accidental Secret Leak risk at **Docker Hub**: Exploitation likelihood is *Unlikely* with *High* impact.

accidental-secret-leak@docker-hub

Unchecked

Code Backdooring risk at **Docker Hub**: Exploitation likelihood is *Unlikely* with *High* impact.

code-backdooring@docker-hub

Unchecked

Low Risk Severity

Unchecked Deployment risk at **Docker Hub**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@docker-hub

Unchecked

Asset Information

ID:	docker-hub
Type:	datastore
Usage:	devops
RAA:	4 %
Size:	system
Technology:	artifact-registry
Tags:	none
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Argo CD Base Image
Data Stored:	Argo CD Base Image
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Ubuntu	
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Incoming Communication Links: 1

Source technical asset names are clickable and link to the corresponding chapter.

Pull Base Image from Docker Hub (incoming)
 Pull the Ubuntu base image from Docker Hub.

Source:	Argo CD Build Pipeline (GitHub Actions)
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops

Tags: none
VPN: false
IP-Filtered: false
Data Received: none
Data Sent: Argo CD Base Image

Host Cluster Kubernetes API: 4 / 4 Risks

Description

Kubernetes API Server for the cluster Argo CD is deployed to

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Hardening risk at **Host Cluster Kubernetes API**: Exploitation likelihood is *Likely* with *Medium* impact.

[missing-hardening@host-cluster-kubernetes-api](#)

Unchecked

Medium Risk Severity

Container Base Image Backdooring risk at **Host Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@host-cluster-kubernetes-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at **Host Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@host-cluster-kubernetes-api](#)

Unchecked

Unnecessary Data Transfer of Argo CD RBAC Config data at **Host Cluster Kubernetes API** from/to **API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server](#)

Unchecked

Asset Information

ID:	host-cluster-kubernetes-api
Type:	external-entity
Usage:	devops
RAA:	57 %
Size:	system
Technology:	web-service-rest
Tags:	none
Internet:	false

Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	AppProject Manifest, Application Manifest, Application Name, ApplicationSet Manifest, ApplicationSet Name, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Rendered Manifests
Data Stored:	API Server Secret, AppProject Manifest, Application Manifest, ApplicationSet Manifest, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Repo Access Credentials
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Cluster Operator
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	The Kubernetes API Server is how Argo CD interacts with the cluster. Argo CD's configuration is stored in the cluster, and Argo CD uses the Kubernetes API Server to apply changes to the cluster. If the Kubernetes API Server is down, Argo CD will be down.

Incoming Communication Links: 7

Source technical asset names are clickable and link to the corresponding chapter.

Get Repo Access Credentials (incoming)

Get repo access credentials from Kubernetes to pull manifest sources from source control.

Source:	Repo Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true

Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: none
Data Sent: Repo Access Credentials

Reconcile Resource State (Host Cluster) (incoming)

Reconcile the current desired manifests with the live state.

Source: ApplicationSet Controller
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: false
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: ApplicationSet Manifest, ApplicationSet Name
Data Sent: Live Manifests

Reconcile Resource State (Host Cluster) (incoming)

Reconcile the current desired manifests with the live state.

Source: Application Controller
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: false
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: Application Name, Rendered Manifests
Data Sent: Live Manifests

Update Repo Access Credentials (incoming)

Write changes from the UI/CLI/API to repo secrets.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Repo Access Credentials
Data Sent:	Repo Access Credentials

Update RBAC Config (incoming)

Write changes from the UI/CLI/API to RBAC config.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD RBAC Config
Data Sent:	Argo CD RBAC Config

Update Cluster Access Config (incoming)

Write changes from the UI/CLI/API to cluster secrets.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false

Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: Cluster Access Configuration, Cluster Access Credentials
Data Sent: Cluster Access Configuration, Cluster Access Credentials

Get/Update/Delete Live Resource State from Kubernetes (Host) (incoming)

Get the live state of an Argo CD-managed resource, or potentially update or delete a resource.

Source: API Server
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: false
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: Application Name, Argo CD RBAC Config, Cluster Access Configuration, Cluster Access Credentials, Rendered Manifests, Repo Access Credentials
Data Sent: API Server Secret, Argo CD RBAC Config, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Repo Access Credentials

Internal Source Control Management UI: 2 / 2 Risks

Description

Internal Source Control Management UI

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Cross-Site Scripting (XSS) risk at **Internal Source Control Management UI**: Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@internal-source-control-management-ui](#)

Unchecked

Low Risk Severity

Server-Side Request Forgery (SSRF) risk at **Internal Source Control Management UI** server-side web-requesting the target **Internal Source Control Management API** via **Push Manifest Sources**: Exploitation likelihood is *Unlikely* with *Low* impact.

[server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources](#)

Unchecked

Asset Information

ID:	internal-source-control-management-ui
Type:	datastore
Usage:	devops
RAA:	12 %
Size:	system
Technology:	web-application
Tags:	none
Internet:	false
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	true
Data Processed:	Manifest Sources

Data Stored: none
Formats Accepted: none of the special data formats accepted

Asset Rating

Owner: Source Control Management Operator
Confidentiality: confidential (rated 4 in scale of 5)
Integrity: mission-critical (rated 5 in scale of 5)
Availability: operational (rated 2 in scale of 5)
CIA-Justification:

Outgoing Communication Links: 1

Target technical asset names are clickable and link to the corresponding chapter.

Push Manifest Sources (outgoing)

Pushes manifests to source control.

Target: Internal Source Control Management API
Protocol: https
Encrypted: true
Authentication: credentials
Authorization: enduser-identity-propagation
Read-Only: false
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: Manifest Sources
Data Received: Manifest Sources

OIDC Provider (External): 6 / 6 Risks

Description

OIDC Provider (External)

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Identity Provider Isolation to further encapsulate and protect identity-related asset **OIDC Provider (External)** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Very High* impact.

[missing-identity-provider-isolation@oidc-provider](#)

Unchecked

Cross-Site Scripting (XSS) risk at **OIDC Provider (External)**: Exploitation likelihood is *Likely* with *High* impact.

[cross-site-scripting@oidc-provider](#)

Unchecked

Missing Authentication covering communication link **Validate External OIDC Token** from **API Server to OIDC Provider (External)**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@api-server>validate-external-oidc-token@api-server@oidc-provider](#)

Unchecked

Cross-Site Request Forgery (CSRF) risk at **OIDC Provider (External)** via **Validate External OIDC Token** from **API Server**: Exploitation likelihood is *Very Likely* with *Medium* impact.

[cross-site-request-forgery@oidc-provider@api-server>validate-external-oidc-token](#)

Unchecked

Medium Risk Severity

Missing Web Application Firewall (WAF) risk at **OIDC Provider (External)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@oidc-provider](#)

Unchecked

Cross-Site Request Forgery (CSRF) risk at **OIDC Provider (External)** via **Proxying to an External OIDC Provider** from **OIDC Proxy (Dex)**: Exploitation likelihood is *Likely* with *Low* impact.

[cross-site-request-forgery@oidc-provider@dex-server>proxying-to-an-external-oidc-provider](#)

Unchecked

Asset Information

ID:	oidc-provider
Type:	process
Usage:	business
RAA:	23 %
Size:	service
Technology:	identity-provider
Tags:	none
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	OIDC Public Keys, OIDC Tokens
Data Stored:	OIDC Public Keys
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Organization
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	

Incoming Communication Links: 2

Source technical asset names are clickable and link to the corresponding chapter.

Proxying to an External OIDC Provider (incoming)

Proxy requests to an external OIDC provider.

Source:	OIDC Proxy (Dex)
Protocol:	https
Encrypted:	true
Authentication:	credentials

Authorization: technical-user
 Read-Only: true
 Usage: devops
 Tags: none
 VPN: false
 IP-Filtered: false
 Data Received: none
 Data Sent: OIDC Tokens

Validate External OIDC Token (incoming)

Get public keys from OIDC provider to validate tokens.

Source: API Server
 Protocol: https
 Encrypted: true
 Authentication: none
 Authorization: none
 Read-Only: true
 Usage: business
 Tags: none
 VPN: false
 IP-Filtered: false
 Data Received: none
 Data Sent: OIDC Public Keys

OIDC Proxy (Dex): 6 / 6 Risks

Description

OIDC Proxy (Dex)

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Identity Provider Isolation to further encapsulate and protect identity-related asset **OIDC Proxy (Dex)** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *Very High* impact.

missing-identity-provider-isolation@dex-server

Unchecked

Cross-Site Scripting (XSS) risk at **OIDC Proxy (Dex)**: Exploitation likelihood is *Likely* with *High* impact.

cross-site-scripting@dex-server

Unchecked

Missing Authentication covering communication link **Validate Dex OIDC Token** from **API Server to OIDC Proxy (Dex)**: Exploitation likelihood is *Likely* with *High* impact.

missing-authentication@api-server>validate-dex-oidc-token@api-server@dex-server

Unchecked

Cross-Site Request Forgery (CSRF) risk at **OIDC Proxy (Dex)** via **Validate Dex OIDC Token** from **API Server**: Exploitation likelihood is *Likely* with *Medium* impact.

cross-site-request-forgery@dex-server@api-server>validate-dex-oidc-token

Unchecked

Medium Risk Severity

Container Base Image Backdooring risk at **OIDC Proxy (Dex)**: Exploitation likelihood is *Unlikely* with *High* impact.

container-baseimage-backdooring@dex-server

Unchecked

Server-Side Request Forgery (SSRF) risk at **OIDC Proxy (Dex)** server-side web-requesting the target **OIDC Provider (External)** via **Proxying to an External OIDC Provider**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Unchecked

Asset Information

ID:	dex-server
Type:	process
Usage:	business
RAA:	30 %
Size:	component
Technology:	identity-provider
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	OIDC Public Keys, OIDC Tokens
Data Stored:	OIDC Public Keys
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Argo CD Operator
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	

Outgoing Communication Links: 1

Target technical asset names are clickable and link to the corresponding chapter.

Proxying to an External OIDC Provider (outgoing)

Proxy requests to an external OIDC provider.

Target:	OIDC Provider (External)
Protocol:	https
Encrypted:	true
Authentication:	credentials

Authorization: technical-user
 Read-Only: true
 Usage: devops
 Tags: none
 VPN: false
 IP-Filtered: false
 Data Sent: none
 Data Received: OIDC Tokens

Incoming Communication Links: 1

Source technical asset names are clickable and link to the corresponding chapter.

Validate Dex OIDC Token (incoming)

Get public keys from Dex server to validate tokens.

Source: API Server
 Protocol: https
 Encrypted: true
 Authentication: none
 Authorization: none
 Read-Only: true
 Usage: devops
 Tags: none
 VPN: false
 IP-Filtered: true
 Data Received: none
 Data Sent: OIDC Public Keys

Quay: 8 / 8 Risks

Description

Quay image repository.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Authentication covering communication link **Pull Argo CD Image from API Server to Quay**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@api-server>pull-argo-cd-image@api-server@quay`

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from Application Controller to Quay**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@application-controller>pull-argo-cd-image@application-controller@quay`

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from ApplicationSet Controller to Quay**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@applicationset-controller>pull-argo-cd-image@applicationset-controller@quay`

Unchecked

Missing Authentication covering communication link **Pull Argo CD Image from Repo Server to Quay**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@repo-server>pull-argo-cd-image@repo-server@quay`

Unchecked

Unguarded Access from Internet of Quay by Argo CD Build Pipeline (GitHub Actions) via Push Image to Quay: Exploitation likelihood is *Very Likely* with *Medium* impact.

`unguarded-access-from-internet@quay@argo-cd-build-pipeline@argo-cd-build-pipeline>push-image-to-quay`

Unchecked

Medium Risk Severity

Accidental Secret Leak risk at **Quay**: Exploitation likelihood is *Unlikely* with *High* impact.

`accidental-secret-leak@quay`

Unchecked

Code Backdooring risk at **Quay**: Exploitation likelihood is *Unlikely* with *High* impact.

code-backdooring@quay

Unchecked

Low Risk Severity

Unchecked Deployment risk at **Quay**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@quay

Unchecked

Asset Information

ID:	quay
Type:	datastore
Usage:	devops
RAA:	27 %
Size:	system
Technology:	artifact-registry
Tags:	none
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Argo CD Container Image, Argo CD Container Image Tag, Quay Push Token
Data Stored:	Argo CD Container Image
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Red Hat	
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Incoming Communication Links: 5

Source technical asset names are clickable and link to the corresponding chapter.

Pull Argo CD Image (incoming)

Pull the Argo CD container image from Quay.

Source:	Repo Server
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD Container Image Tag
Data Sent:	Argo CD Container Image

Push Image to Quay (incoming)

Quay image repository.

Source:	Argo CD Build Pipeline (GitHub Actions)
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD Container Image, Quay Push Token
Data Sent:	none

Pull Argo CD Image (incoming)

Pull the Argo CD container image from Quay.

Source:	ApplicationSet Controller
Protocol:	https

Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD Container Image Tag
Data Sent:	Argo CD Container Image

Pull Argo CD Image (incoming)

Pull the Argo CD container image from Quay.

Source:	Application Controller
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Argo CD Container Image Tag
Data Sent:	Argo CD Container Image

Pull Argo CD Image (incoming)

Pull the Argo CD container image from Quay.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false

Data Received: Argo CD Container Image Tag

Data Sent: Argo CD Container Image

Rendered Manifests Cache (Redis): 2 / 2 Risks

Description

Rendered Manifests Cache (Redis)

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Authentication covering communication link **Send/Receive Cached Rendered Manifests from Repo Server to Rendered Manifests Cache (Redis)**: Exploitation likelihood is *Likely* with *High* impact.

[missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache](#)

Unchecked

Medium Risk Severity

Container Base Image Backdooring risk at **Rendered Manifests Cache (Redis)**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@rendered-manifests-cache](#)

Unchecked

Asset Information

ID:	rendered-manifests-cache
Type:	datastore
Usage:	devops
RAA:	19 %
Size:	component
Technology:	web-service-rest
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	true
Custom-Developed:	false
Client by Human:	false
Data Processed:	none

Data Stored: AppProject Manifest, Application Manifest, Application Name, Rendered Manifests

Formats Accepted: JSON

Asset Rating

Owner: Argo CD Operator

Confidentiality: confidential (rated 4 in scale of 5)

Integrity: mission-critical (rated 5 in scale of 5)

Availability: operational (rated 2 in scale of 5)

CIA-Justification:

Incoming Communication Links: 1

Source technical asset names are clickable and link to the corresponding chapter.

Send/Receive Cached Rendered Manifests (incoming)

Sends and receives rendered manifests to and from the cache.

Source: Repo Server

Protocol: https

Encrypted: true

Authentication: none

Authorization: none

Read-Only: false

Usage: devops

Tags: none

VPN: false

IP-Filtered: true

Data Received: Application Name, Rendered Manifests

Data Sent: Application Name, Rendered Manifests

Repo Server: 9 / 9 Risks

Description

Pulls from manifests sources, builds manifests, caches manifests

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Elevated Risk Severity

Missing Authentication covering communication link **Fetching Rendered Manifests from Cache from API Server to Repo Server**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server`

Unchecked

Missing Authentication covering communication link **Rendered Manifest Requests from Application Controller to Repo Server**: Exploitation likelihood is *Likely* with *High* impact.

`missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server`

Unchecked

Path-Traversal risk at **Repo Server** against filesystem **Repo Server Storage** via **Store Cached Manifest Sources**: Exploitation likelihood is *Likely* with *High* impact.

`path-traversal@repo-server@repo-server-storage@repo-server>store-cached-manifest-sources`

Unchecked

Missing File Validation risk at **Repo Server**: Exploitation likelihood is *Very Likely* with *Medium* impact.

`missing-file-validation@repo-server`

Unchecked

Medium Risk Severity

Container Base Image Backdooring risk at **Repo Server**: Exploitation likelihood is *Unlikely* with *High* impact.

`container-baseimage-backdooring@repo-server`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Host Cluster Kubernetes API** via **Get Repo Access Credentials**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Internal Source Control Management API** via **Fetch Manifest Sources**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Unchecked

Server-Side Request Forgery (SSRF) risk at **Repo Server** server-side web-requesting the target **Rendered Manifests Cache (Redis)** via **Send/Receive Cached Rendered Manifests**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Unchecked

Asset Information

ID:	repo-server
Type:	process
Usage:	devops
RAA:	28 %
Size:	component
Technology:	web-service-rest
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	true
Custom-Developed:	true
Client by Human:	false
Data Processed:	Argo CD Container Image, Argo CD Container Image Tag, Manifest Sources, Rendered Manifests, Repo Access Credentials
Data Stored:	Manifest Sources
Formats Accepted:	File, JSON

Asset Rating

Owner:	Argo CD Repo-Server
Confidentiality:	confidential (rated 4 in scale of 5)

Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	critical	(rated 4 in scale of 5)
CIA-Justification:	The repo-server is responsible for fetching manifests and performing transformations on them ("building"). It contains sensitive information, i.e. deployment manifests. Changes to these manifests can change what's deployed. If the application controller is down, no other services will be disrupted, but no deployments will be made.	

Outgoing Communication Links: 5

Target technical asset names are clickable and link to the corresponding chapter.

Store Cached Manifest Sources (outgoing)

Cache manifest sources (from git, helm repo, OCI, etc.) to local ephemeral storage.

Target:	Repo Server Storage
Protocol:	local-file-access
Encrypted:	false
Authentication:	none
Authorization:	none
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Manifest Sources
Data Received:	Manifest Sources

Send/Receive Cached Rendered Manifests (outgoing)

Sends and receives rendered manifests to and from the cache.

Target:	Rendered Manifests Cache (Redis)
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false

IP-Filtered: true
Data Sent: Application Name, Rendered Manifests
Data Received: Application Name, Rendered Manifests

Pull Argo CD Image (outgoing)

Pull the Argo CD container image from Quay.

Target: Quay
Protocol: https
Encrypted: true
Authentication: none
Authorization: none
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: Argo CD Container Image Tag
Data Received: Argo CD Container Image

Get Repo Access Credentials (outgoing)

Get repo access credentials from Kubernetes to pull manifest sources from source control.

Target: Host Cluster Kubernetes API
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: none
Data Received: Repo Access Credentials

Fetch Manifest Sources (outgoing)

Pulls manifest sources from source control.

Target: Internal Source Control Management API

Protocol:	https
Encrypted:	true
Authentication:	credentials
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Application Name
Data Received:	Manifest Sources

Incoming Communication Links: 2

Source technical asset names are clickable and link to the corresponding chapter.

Rendered Manifest Requests (incoming)

Fetch manifests from the repo server to be applied to the cluster.

Source:	Application Controller
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	true
Data Received:	Application Name
Data Sent:	Rendered Manifests

Fetching Rendered Manifests from Cache (incoming)

Fetch manifests from the repo server to display via UI or CLI.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none

Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: true
Data Received: Application Name
Data Sent: Rendered Manifests

Application Controller: 6 / 6 Risks

Description

Some Description

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Container Base Image Backdooring risk at **Application Controller**: Exploitation likelihood is *Unlikely* with *High* impact.

`container-baseimage-backdooring@application-controller`

Unchecked

Missing Build Infrastructure in the threat model (referencing asset **Application Controller** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

`missing-build-infrastructure@application-controller`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **External Cluster Kubernetes API** via **Reconcile Resource State (External Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Host Cluster Kubernetes API** via **Reconcile Resource State (Host Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Application Controller** server-side web-requesting the target **Repo Server** via **Rendered Manifest Requests**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests`

Unchecked

Asset Information

ID:	application-controller
Type:	process
Usage:	devops
RAA:	38 %
Size:	component
Technology:	web-service-rest
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	true
Client by Human:	false
Data Processed:	Argo CD Container Image, Argo CD Container Image Tag, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Rendered Manifests
Data Stored:	none
Formats Accepted:	JSON

Asset Rating

Owner:	Argo CD Operator	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	critical	(rated 4 in scale of 5)
CIA-Justification:	The application controller is responsible for deploying applications. It contains sensitive information, i.e. deployment manifests. Changes to these manifests can change what's deployed. If the application controller is down, no other services will be disrupted, but no deployments will be made.	

Outgoing Communication Links: 4

Target technical asset names are clickable and link to the corresponding chapter.

Rendered Manifest Requests (outgoing)

Fetch manifests from the repo server to be applied to the cluster.

Target:	Repo Server
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	true
Data Sent:	Application Name
Data Received:	Rendered Manifests

Reconcile Resource State (Host Cluster) (outgoing)

Reconcile the current desired manifests with the live state.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Application Name, Rendered Manifests
Data Received:	Live Manifests

Reconcile Resource State (External Cluster) (outgoing)

Reconcile the current desired manifests with the live state in an external cluster.

Target:	External Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops

Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Application Name, Cluster Access Credentials, Rendered Manifests
Data Received:	Live Manifests

Pull Argo CD Image (outgoing)

Pull the Argo CD container image from Quay.

Target:	Quay
Protocol:	https
Encrypted:	true
Authentication:	none
Authorization:	none
Read-Only:	true
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Argo CD Container Image Tag
Data Received:	Argo CD Container Image

ApplicationSet Controller: 4 / 4 Risks

Description

Some Description

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Container Base Image Backdooring risk at **ApplicationSet Controller**: Exploitation likelihood is *Unlikely* with *High* impact.

`container-baseimage-backdooring@applicationset-controller`

Unchecked

Server-Side Request Forgery (SSRF) risk at **ApplicationSet Controller** server-side web-requesting the target **Host Cluster Kubernetes API** via **Reconcile Resource State (Host Cluster)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster`

Unchecked

Server-Side Request Forgery (SSRF) risk at **ApplicationSet Controller** server-side web-requesting the target **Internal Source Control Management API** via **Git Generator Pull**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull`

Unchecked

Server-Side Request Forgery (SSRF) risk at **ApplicationSet Controller** server-side web-requesting the target **Quay** via **Pull Argo CD Image**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image`

Unchecked

Asset Information

ID:	applicationset-controller
Type:	process
Usage:	devops
RAA:	49 %
Size:	component
Technology:	web-service-rest
Tags:	none
Internet:	false

Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	true
Client by Human:	false
Data Processed:	ApplicationSet Manifest, ApplicationSet Name, Argo CD Container Image, Argo CD Container Image Tag, Cluster Access Configuration, Cluster Access Credentials, Git Branch Name, Git Organization Name, Git Repo URL, Live Manifests, Repo Access Credentials
Data Stored:	none
Formats Accepted:	JSON

Asset Rating

Owner:	Argo CD Operator	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	critical	(rated 4 in scale of 5)
CIA-Justification:	The ApplicationSet controller is responsible for deploying ApplicationSets. It contains sensitive information, i.e. deployment manifests. Changes to these manifests can change what's deployed.	

Outgoing Communication Links: 3

Target technical asset names are clickable and link to the corresponding chapter.

Reconcile Resource State (Host Cluster) (outgoing)

Reconcile the current desired manifests with the live state.

Target:	Host Cluster Kubernetes API
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false

IP-Filtered: false
Data Sent: ApplicationSet Manifest, ApplicationSet Name
Data Received: Live Manifests

Pull Argo CD Image (outgoing)

Pull the Argo CD container image from Quay.

Target: Quay
Protocol: https
Encrypted: true
Authentication: none
Authorization: none
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: Argo CD Container Image Tag
Data Received: Argo CD Container Image

Git Generator Pull (outgoing)

Get information about organizations, branches, and pull requests from the SCM.

Target: Internal Source Control Management API
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: Git Repo URL, Repo Access Credentials
Data Received: Git Branch Name, Git Organization Name, Git Repo Name

Argo CD Build Pipeline (GitHub Actions): 5 / 5 Risks

Description

Argo CD build pipeline, hosted on GitHub Actions.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Code Backdooring risk at **Argo CD Build Pipeline (GitHub Actions)**: Exploitation likelihood is *Unlikely* with *High* impact.

`code-backdooring@argo-cd-build-pipeline`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Argo CD Build Pipeline (GitHub Actions)** server-side web-requesting the target **Argo CD Source Repo (GitHub)** via **Pull Source**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Argo CD Build Pipeline (GitHub Actions)** server-side web-requesting the target **Docker Hub** via **Pull Base Image from Docker Hub**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub`

Unchecked

Server-Side Request Forgery (SSRF) risk at **Argo CD Build Pipeline (GitHub Actions)** server-side web-requesting the target **Quay** via **Push Image to Quay**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay`

Unchecked

Unchecked Deployment risk at **Argo CD Build Pipeline (GitHub Actions)**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`unchecked-deployment@argo-cd-build-pipeline`

Unchecked

Asset Information

ID:	argo-cd-build-pipeline
Type:	process
Usage:	devops
RAA:	25 %

Size:	application
Technology:	build-pipeline
Tags:	none
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Argo CD Base Image, Argo CD Container Image, Argo CD Source, Quay Push Token
Data Stored:	Argo CD GitHub Push Token, Quay Push Token
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	GitHub
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	

Outgoing Communication Links: 3

Target technical asset names are clickable and link to the corresponding chapter.

Push Image to Quay (outgoing)

Quay image repository.

Target:	Quay
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false

IP-Filtered: false
Data Sent: Argo CD Container Image, Quay Push Token
Data Received: none

Pull Source (outgoing)

Pull the Argo CD source from the GitHub repo.

Target: Argo CD Source Repo (GitHub)
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: none
Data Received: Argo CD Source

Pull Base Image from Docker Hub (outgoing)

Pull the Ubuntu base image from Docker Hub.

Target: Docker Hub
Protocol: https
Encrypted: true
Authentication: none
Authorization: none
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Sent: none
Data Received: Argo CD Base Image

Argo CD Maintainer Git Client: 2 / 2 Risks

Description

Git client (and configuration) used by an Argo CD maintainer.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Identity Store in the threat model (referencing asset **Argo CD Maintainer Git Client** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-identity-store@argo-cd-maintainer-git-client](#)

Unchecked

Server-Side Request Forgery (SSRF) risk at **Argo CD Maintainer Git Client** server-side web-requesting the target **Argo CD Source Repo (GitHub)** via **Push Code/Tags to GitHub**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github](#)

Unchecked

Asset Information

ID:	argo-cd-maintainer-git-client
Type:	process
Usage:	devops
RAA:	0 %
Size:	application
Technology:	cli
Tags:	none
Internet:	false
Machine:	physical
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	true
Data Processed:	Argo CD Container Image Tag, Argo CD Source
Data Stored:	Argo CD Container Image Tag, Argo CD GitHub Push Token, Argo CD Source

Formats Accepted: File

Asset Rating

Owner: Argo CD Maintainer
Confidentiality: strictly-confidential (rated 5 in scale of 5)
Integrity: mission-critical (rated 5 in scale of 5)
Availability: operational (rated 2 in scale of 5)
CIA-Justification:

Outgoing Communication Links: 1

Target technical asset names are clickable and link to the corresponding chapter.

Push Code/Tags to GitHub (outgoing)

Push code to the Argo CD repo (as when cherry-picking changes) and/or push tags (as when cutting a release).

Target:	Argo CD Source Repo (GitHub)
Protocol:	https
Encrypted:	true
Authentication:	two-factor
Authorization:	enduser-identity-propagation
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	Argo CD Container Image Tag, Argo CD GitHub Push Token, Argo CD Source
Data Received:	Argo CD Container Image Tag, Argo CD Source

External Cluster Kubernetes API: 2 / 2 Risks

Description

Kubernetes API Server for a cluster Argo CD is managing

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Container Base Image Backdooring risk at **External Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@external-cluster-kubernetes-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at **External Cluster Kubernetes API**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@external-cluster-kubernetes-api](#)

Unchecked

Asset Information

ID:	external-cluster-kubernetes-api
Type:	external-entity
Usage:	devops
RAA:	19 %
Size:	system
Technology:	web-service-rest
Tags:	none
Internet:	false
Machine:	container
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Cluster Access Credentials, Live Manifests, Rendered Manifests
Data Stored:	Live Manifests
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	External Cluster Operator	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Incoming Communication Links: 2

Source technical asset names are clickable and link to the corresponding chapter.

Reconcile Resource State (External Cluster) (incoming)

Reconcile the current desired manifests with the live state in an external cluster.

Source:	Application Controller
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Application Name, Cluster Access Credentials, Rendered Manifests
Data Sent:	Live Manifests

Get/Update/Delete Live Resource State from Kubernetes (External) (incoming)

Get the live state of an Argo CD-managed resource, or potentially update or delete a resource on an external cluster.

Source:	API Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	none

VPN: false
IP-Filtered: false
Data Received: Application Name, Cluster Access Credentials, Rendered Manifests
Data Sent: Live Manifests

Internal Source Control Management API: 2 / 2 Risks

Description

Source control manager (GitHub, GitLab, Helm repo, etc.) accessible only from the organization's network.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Missing Vault (Secret Storage) in the threat model (referencing asset **Internal Source Control Management API** as an example): Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-vault@internal-source-control-management-api](#)

Unchecked

Missing Web Application Firewall (WAF) risk at **Internal Source Control Management API**: Exploitation likelihood is *Unlikely* with *Medium* impact.

[missing-waf@internal-source-control-management-api](#)

Unchecked

Asset Information

ID:	internal-source-control-management-api
Type:	datastore
Usage:	devops
RAA:	33 %
Size:	system
Technology:	web-service-rest
Tags:	none
Internet:	false
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Git Branch Name, Git Organization Name, Git Repo URL, Repo Access Credentials
Data Stored:	Manifest Sources

Formats Accepted: none of the special data formats accepted

Asset Rating

Owner: Source Control Management Operator
Confidentiality: confidential (rated 4 in scale of 5)
Integrity: mission-critical (rated 5 in scale of 5)
Availability: mission-critical (rated 5 in scale of 5)
CIA-Justification:

Incoming Communication Links: 3

Source technical asset names are clickable and link to the corresponding chapter.

Fetch Manifest Sources (incoming)

Pulls manifest sources from source control.

Source: Repo Server
Protocol: https
Encrypted: true
Authentication: credentials
Authorization: technical-user
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: Application Name
Data Sent: Manifest Sources

Push Manifest Sources (incoming)

Pushes manifests to source control.

Source: Internal Source Control Management UI
Protocol: https
Encrypted: true
Authentication: credentials
Authorization: enduser-identity-propagation
Read-Only: false
Usage: devops

Tags: none
VPN: false
IP-Filtered: false
Data Received: Manifest Sources
Data Sent: Manifest Sources

Git Generator Pull (incoming)

Get information about organizations, branches, and pull requests from the SCM.

Source: ApplicationSet Controller
Protocol: https
Encrypted: true
Authentication: token
Authorization: technical-user
Read-Only: true
Usage: devops
Tags: none
VPN: false
IP-Filtered: false
Data Received: Git Repo URL, Repo Access Credentials
Data Sent: Git Branch Name, Git Organization Name, Git Repo Name

Repo Server Storage: 2 / 2 Risks

Description

Local (by default, ephemeral) storage for the repo-server.

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Container Base Image Backdooring risk at **Repo Server Storage**: Exploitation likelihood is *Unlikely* with *High* impact.

[container-baseimage-backdooring@repo-server-storage](#)

Unchecked

Unencrypted Technical Asset named **Repo Server Storage** missing enduser-individual encryption with data-with-enduser-individual-key: Exploitation likelihood is *Unlikely* with *High* impact.

[unencrypted-asset@repo-server-storage](#)

Unchecked

Asset Information

ID:	repo-server-storage
Type:	datastore
Usage:	devops
RAA:	12 %
Size:	component
Technology:	local-file-system
Tags:	none
Internet:	false
Machine:	container
Encryption:	none
Multi-Tenant:	true
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Manifest Sources
Data Stored:	Manifest Sources
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Cluster Operator	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Incoming Communication Links: 1

Source technical asset names are clickable and link to the corresponding chapter.

Store Cached Manifest Sources (incoming)
Cache manifest sources (from git, helm repo, OCI, etc.) to local ephemeral storage.

Source:	Repo Server
Protocol:	local-file-access
Encrypted:	false
Authentication:	none
Authorization:	none
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Received:	Manifest Sources
Data Sent:	Manifest Sources

User CLI: 1 / 1 Risk

Description

User CLI

Identified Risks of Asset

Risk finding paragraphs are clickable and link to the corresponding chapter.

Medium Risk Severity

Server-Side Request Forgery (SSRF) risk at **User CLI** server-side web-requesting the target **API Server** via **Make Requests to API Server**: Exploitation likelihood is *Unlikely* with *Medium* impact.

`server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server`

Unchecked

Asset Information

ID:	user-cli
Type:	process
Usage:	devops
RAA:	44 %
Size:	application
Technology:	cli
Tags:	none
Internet:	false
Machine:	physical
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	true
Data Processed:	AppProject Manifest, Application Manifest, Application Name, Argo Tokens, Cluster Access Configuration, Cluster Access Credentials, Live Manifests, Manifest Sources, OIDC Tokens, Rendered Manifests, Repo Access Credentials
Data Stored:	Argo Tokens, OIDC Tokens
Formats Accepted:	none of the special data formats accepted

Asset Rating

Owner:	Argo CD User	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		

Outgoing Communication Links: 1

Target technical asset names are clickable and link to the corresponding chapter.

Make Requests to API Server (outgoing)

Make requests to the API server.

Target:	API Server
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	enduser-identity-propagation
Read-Only:	false
Usage:	devops
Tags:	none
VPN:	false
IP-Filtered:	false
Data Sent:	AppProject Manifest, Application Manifest, Application Name, Cluster Access Configuration, Cluster Access Credentials, Manifest Sources, OIDC Tokens, Rendered Manifests, Repo Access Credentials
Data Received:	Argo Tokens

Identified Data Breach Probabilities by Data Asset

In total **88 potential risks** have been identified during the threat modeling process of which **0 are rated as critical, 0 as high, 27 as elevated, 57 as medium, and 4 as low.**

These risks are distributed across **26 data assets**. The following sub-chapters of this section describe the derived data breach probabilities grouped by data asset.

Technical asset names and risk IDs are clickable and link to the corresponding chapter.

API Server Secret: 32 / 32 Risks

A randomly-generated key used by the API server to sign access tokens.

ID:	api-server-secret	
Usage:	devops	
Quantity:	few	
Tags:	none	
Origin:	Argo CD API Server	
Owner:		
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	operational	(rated 2 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server	
Stored by:	Host Cluster Kubernetes API	
Sent via:	none	
Received via:	Get/Update/Delete Live Resource State from Kubernetes (Host)	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 32 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Possible: cross-site-scripting@api-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

AppProject Manifest: 35 / 35 Risks

Manifest of an AppProject, a CRD which expresses certain rules applied to Applications.

ID:	appproject-manifest
Usage:	devops
Quantity:	few
Tags:	none
Origin:	Argo CD Operator or Users
Owner:	Argo CD Operator
Confidentiality:	confidential (rated 4 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	
Processed by:	API Server, Host Cluster Kubernetes API, User CLI
Stored by:	Host Cluster Kubernetes API, Rendered Manifests Cache (Redis)
Sent via:	Make Requests to API Server
Received via:	none
Data Breach:	probable
Data Breach Risks:	This data asset has data breach potential because of 35 remaining risks:

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Probable: container-baseimage-backdooring@rendered-manifests-cache

Possible: cross-site-scripting@api-server

Possible: missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Application Manifest: 35 / 35 Risks

Manifest of an Application, defining things such as manifest source and destination cluster.

ID:	application-manifest	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD Operator or Users	
Owner:	Argo CD Operator	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	important	(rated 3 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, Host Cluster Kubernetes API, User CLI	
Stored by:	Host Cluster Kubernetes API, Rendered Manifests Cache (Redis)	
Sent via:	Make Requests to API Server	
Received via:	none	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 35 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Probable: container-baseimage-backdooring@rendered-manifests-cache

Possible: cross-site-scripting@api-server

Possible: missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Application Name: 35 / 35 Risks

Name of the Application

ID:	application-name	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Some Origin	
Owner:	Argo CD	
Confidentiality:	restricted	(rated 3 in scale of 5)
Integrity:	operational	(rated 2 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:	The application name is used to identify the application and the project which controls it. It should be protected from tenants or users who do not have access to that application.	
Processed by:	API Server, Host Cluster Kubernetes API, User CLI	
Stored by:	Rendered Manifests Cache (Redis)	
Sent via:	Send/Receive Cached Rendered Manifests, Rendered Manifest Requests, Reconcile Resource State (Host Cluster), Reconcile Resource State (External Cluster), Make Requests to API Server, Get/Update/Delete Live Resource State from Kubernetes (Host), Get/Update/Delete Live Resource State from Kubernetes (External), Fetching Rendered Manifests from Cache, Fetch Manifest Sources	
Received via:	Send/Receive Cached Rendered Manifests	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 35 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Probable: container-baseimage-backdooring@rendered-manifests-cache

Possible: cross-site-scripting@api-server

Possible: missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

ApplicationSet Manifest: 33 / 33 Risks

Manifest representing an ApplicationSet.

ID:	applicationset-manifest	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD Operator	
Owner:		
Confidentiality:	restricted	(rated 3 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, ApplicationSet Controller, Host Cluster Kubernetes API	
Stored by:	Host Cluster Kubernetes API	
Sent via:	Reconcile Resource State (Host Cluster)	
Received via:	none	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 33 remaining risks:	
	Probable: container-baseimage-backdooring@api-server	
	Probable: container-baseimage-backdooring@applicationset-controller	
	Probable: container-baseimage-backdooring@host-cluster-kubernetes-api	
	Possible: cross-site-scripting@api-server	
	Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server	
	Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials	
	Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token	
	Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token	
	Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image	
	Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache	
	Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster	
	Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image	
	Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests	
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull	
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image	
	Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider	
	Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials	
	Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources	

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

ApplicationSet Name: 8 / 8 Risks

Name of an ApplicationSet.

ID:	applicationset-name	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD Operator	
Owner:		
Confidentiality:	restricted	(rated 3 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	ApplicationSet Controller, Host Cluster Kubernetes API	
Stored by:	none	
Sent via:	Reconcile Resource State (Host Cluster)	
Received via:	none	
Data Breach:	probable	

Data Breach Risks: This data asset has data breach potential because of 8 remaining risks:

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Argo CD Base Image: 11 / 11 Risks

Ubuntu base image which the Argo CD image is built on.

ID:	argo-cd-base-image	
Usage:	devops	
Quantity:	very-few	
Tags:	none	
Origin:	Ubuntu	
Owner:		
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	Argo CD Build Pipeline (GitHub Actions), Docker Hub	
Stored by:	Docker Hub	
Sent via:	none	
Received via:	Pull Base Image from Docker Hub	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 11 remaining risks:	

Probable: accidental-secret-leak@docker-hub

Probable: code-backdooring@argo-cd-build-pipeline

Probable: code-backdooring@docker-hub

Possible: missing-authentication@argo-cd-build-pipeline>pull-base-image-from-docker-hub@argo-cd-build-pipeline@docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: unchecked-deployment@argo-cd-build-pipeline

Possible: unchecked-deployment@docker-hub

Possible: unguarded-access-from-internet@docker-hub@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Argo CD Container Image: 48 / 48 Risks

Argo CD container image, primarily hosted on Quay.

ID:	argo-cd-container-image	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD GitHub build pipeline	
Owner:		
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, Application Controller, ApplicationSet Controller, Argo CD Build Pipeline (GitHub Actions), Quay, Repo Server	
Stored by:	Quay	
Sent via:	Push Image to Quay	
Received via:	Pull Argo CD Image, Pull Argo CD Image, Pull Argo CD Image, Pull Argo CD Image	

Data Breach: **probable**

Data Breach Risks: This data asset has data breach potential because of 48 remaining risks:

Probable: accidental-secret-leak@quay

Probable: code-backdooring@argo-cd-build-pipeline

Probable: code-backdooring@quay

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@application-controller

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@repo-server

Probable: missing-file-validation@repo-server

Possible: cross-site-scripting@api-server

Possible: missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server

Possible: missing-authentication@api-server>pull-argo-cd-image@api-server@quay

Possible: missing-authentication@application-controller>pull-argo-cd-image@application-controller@quay

Possible: missing-authentication@applicationset-controller>pull-argo-cd-image@applicationset-controller@quay

Possible: missing-authentication@repo-server>pull-argo-cd-image@repo-server@quay

Possible: missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: unchecked-deployment@argo-cd-build-pipeline

Possible: unchecked-deployment@quay

Possible: unguarded-access-from-internet@quay@argo-cd-build-pipeline@argo-cd-build-pipeline>push-image-to-quay

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Argo CD Container Image Tag: 48 / 48 Risks

Repo name/tag for the Argo CD container image.

ID:	argo-cd-container-image-tag	
Usage:	devops	
Quantity:	very-few	
Tags:	none	
Origin:	Manifests repository.	
Owner:		
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, Application Controller, ApplicationSet Controller, Argo CD Maintainer Git Client, Quay, Repo Server	
Stored by:	Argo CD Maintainer Git Client	
Sent via:	Push Code/Tags to GitHub, Pull Argo CD Image, Pull Argo CD Image, Pull Argo CD Image, Pull Argo CD Image	
Received via:	Push Code/Tags to GitHub	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 48 remaining risks:	

Probable: accidental-secret-leak@quay

Probable: code-backdooring@argo-cd-build-pipeline

Probable: code-backdooring@quay

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@application-controller

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@repo-server

Probable: missing-file-validation@repo-server

Possible: cross-site-scripting@api-server

Possible: missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server

Possible: missing-authentication@api-server>pull-argo-cd-image@api-server@quay

Possible: missing-authentication@application-controller>pull-argo-cd-image@application-controller@quay

Possible: missing-authentication@applicationset-controller>pull-argo-cd-image@applicationset-controller@quay

Possible: missing-authentication@repo-server>pull-argo-cd-image@repo-server@quay

Possible: missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: unchecked-deployment@argo-cd-build-pipeline

Possible: unchecked-deployment@quay

Possible: unguarded-access-from-internet@quay@argo-cd-build-pipeline@argo-cd-build-pipeline>push-image-to-quay

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Argo CD GitHub Push Token: 6 / 6 Risks

A token granting push access to the Argo CD GitHub repo.

ID:	argo-cd-github-push-token	
Usage:	devops	
Quantity:	few	
Tags:	none	
Origin:	GitHub UI or API	
Owner:		
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	none	
Stored by:	Argo CD Build Pipeline (GitHub Actions), Argo CD Maintainer Git Client	
Sent via:	Push Code/Tags to GitHub	
Received via:	none	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 6 remaining risks:	

Probable: code-backdooring@argo-cd-build-pipeline

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: unchecked-deployment@argo-cd-build-pipeline

Argo CD RBAC Config: 28 / 28 Risks

The RBAC settings for an Argo CD instance.

ID:	argo-cd-rbac-config	
Usage:	devops	
Quantity:	very-few	
Tags:	none	
Origin:	Argo CD Operator	
Owner:		
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server	
Stored by:	API Server	
Sent via:	Update RBAC Config, Get/Update/Delete Live Resource State from Kubernetes (Host)	
Received via:	Update RBAC Config, Get/Update/Delete Live Resource State from Kubernetes (Host)	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 28 remaining risks:	
	Probable: container-baseimage-backdooring@api-server	
	Possible: cross-site-scripting@api-server	
	Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server	
	Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials	
	Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token	
	Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token	
	Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image	
	Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache	
	Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster	
	Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image	
	Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests	
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull	
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image	
	Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider	
	Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials	
	Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources	

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Argo CD Source: 12 / 12 Risks

The source code for Argo CD.

ID:	argo-cd-source	
Usage:	devops	
Quantity:	very-few	
Tags:	none	
Origin:	Argo CD maintainers	
Owner:		
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	Argo CD Build Pipeline (GitHub Actions), Argo CD Maintainer Git Client, Argo CD Source Repo (GitHub)	
Stored by:	Argo CD Maintainer Git Client, Argo CD Source Repo (GitHub)	
Sent via:	Push Code/Tags to GitHub	
Received via:	Push Code/Tags to GitHub, Pull Source	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 12 remaining risks:	

Probable: accidental-secret-leak@argo-cd-source-repo

Probable: code-backdooring@argo-cd-build-pipeline

Probable: code-backdooring@argo-cd-source-repo

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: unchecked-deployment@argo-cd-build-pipeline

Possible: unchecked-deployment@argo-cd-source-repo

Possible: unguarded-access-from-internet@argo-cd-source-repo@argo-cd-build-pipeline@argo-cd-build-pipeline>pull-source

Improbable: unnecessary-data-transfer@argo-cd-container-image-tag@argo-cd-source-repo@argo-cd-maintainer-git-client

Improbable: unnecessary-data-transfer@argo-cd-github-push-token@argo-cd-source-repo@argo-cd-maintainer-git-client

Argo Tokens: 29 / 29 Risks

API access tokens generated by the API server and validated by the API server.

ID:	argo-tokens
Usage:	devops
Quantity:	very-many
Tags:	none
Origin:	Argo CD API Server
Owner:	
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	operational (rated 2 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	
Processed by:	API Server, User CLI
Stored by:	User CLI
Sent via:	none
Received via:	Make Requests to API Server
Data Breach:	probable

Data Breach Risks: This data asset has data breach potential because of 29 remaining risks:

Probable: container-baseimage-backdooring@api-server

Possible: cross-site-scripting@api-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: `server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server`

Improbable: `cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server`

Improbable: `missing-hardening@api-server`

Improbable: `missing-network-segmentation@api-server`

Improbable: `missing-waf@api-server`

Cluster Access Configuration: 35 / 35 Risks

Configuration for external cluster access. Includes clusterResources boolean and namespaces list.

ID:	cluster-access-configuration	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD Operator	
Owner:		
Confidentiality:	restricted	(rated 3 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:	Integrity is critical because an attacker could disable TLS validation or enable write access to cluster resources or out-of-bounds namespaces, of those were previously restricted at the cluster config level. Resource scope and destinations may be restricted elsewhere as well (for example, in an AppProject).	
Processed by:	API Server, Application Controller, ApplicationSet Controller, Host Cluster Kubernetes API, User CLI	
Stored by:	Host Cluster Kubernetes API	
Sent via:	Update Cluster Access Config, Make Requests to API Server, Get/Update/Delete Live Resource State from Kubernetes (Host)	
Received via:	Update Cluster Access Config, Get/Update/Delete Live Resource State from Kubernetes (Host)	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 35 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@application-controller

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Possible: cross-site-scripting@api-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Cluster Access Credentials: 41 / 41 Risks

Credentials granting access to manage an external Kubernetes cluster's resources.

ID:	cluster-access-credentials
Usage:	devops
Quantity:	many
Tags:	none
Origin:	External cluster API.
Owner:	
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	operational (rated 2 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	
Processed by:	API Server, Application Controller, ApplicationSet Controller, External Cluster Kubernetes API, Host Cluster Kubernetes API, User CLI
Stored by:	Host Cluster Kubernetes API
Sent via:	Update Cluster Access Config, Reconcile Resource State (External Cluster), Make Requests to API Server, Get/Update/Delete Live Resource State from Kubernetes (Host), Get/Update/Delete Live Resource State from Kubernetes (External)
Received via:	Update Cluster Access Config, Get/Update/Delete Live Resource State from Kubernetes (Host)

Data Breach: **probable**

Data Breach Risks: This data asset has data breach potential because of 41 remaining risks:

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@application-controller

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@external-cluster-kubernetes-api

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Possible: cross-site-scripting@api-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@external-cluster-kubernetes-api

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Git Branch Name: 7 / 7 Risks

Name of a branch in a git repo.

ID:	git-branch-name		
Usage:	devops		
Quantity:	very-many		
Tags:	none		
Origin:	SCM API or Argo CD User		
Owner:			
Confidentiality:	confidential	(rated 4 in scale of 5)	
Integrity:	critical	(rated 4 in scale of 5)	
Availability:	operational	(rated 2 in scale of 5)	
CIA-Justification:			
Processed by:	ApplicationSet Controller, Internal Source Control Management API		
Stored by:	none		
Sent via:	none		
Received via:	Git Generator Pull		
Data Breach:	probable		
Data Breach Risks:	This data asset has data breach potential because of 7 remaining risks:		
	Probable: container-baseimage-backdooring@applicationset-controller		
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster		
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull		
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image		
	Possible:		
	server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources		
	Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server		
	Improbable: missing-waf@internal-source-control-management-api		

Git Organization Name: 7 / 7 Risks

Name of an organization/project in a git source control management system.

ID:	git-org-name		
Usage:	devops		
Quantity:	many		
Tags:	none		
Origin:	SCM API or Argo CD User		
Owner:			
Confidentiality:	confidential	(rated 4 in scale of 5)	
Integrity:	important	(rated 3 in scale of 5)	
Availability:	operational	(rated 2 in scale of 5)	
CIA-Justification:			
Processed by:	ApplicationSet Controller, Internal Source Control Management API		
Stored by:	none		
Sent via:	none		
Received via:	Git Generator Pull		
Data Breach:	probable		
Data Breach Risks:	This data asset has data breach potential because of 7 remaining risks:		
	Probable: container-baseimage-backdooring@applicationset-controller		
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster		
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull		
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image		
	Possible:		
	server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources		
	Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server		
	Improbable: missing-waf@internal-source-control-management-api		

Git Repo URL: 7 / 7 Risks

URL of a git repo.

ID:	git-repo-url		
Usage:	devops		
Quantity:	many		
Tags:	none		
Origin:	SCM API or Argo CD User		
Owner:			
Confidentiality:	confidential	(rated 4 in scale of 5)	
Integrity:	important	(rated 3 in scale of 5)	
Availability:	operational	(rated 2 in scale of 5)	
CIA-Justification:			
Processed by:	ApplicationSet Controller, Internal Source Control Management API		
Stored by:	none		
Sent via:	Git Generator Pull		
Received via:	none		
Data Breach:	probable		
Data Breach Risks:	This data asset has data breach potential because of 7 remaining risks:		
	Probable: container-baseimage-backdooring@applicationset-controller		
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster		
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull		
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image		
	Possible:		
	server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources		
	Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server		
	Improbable: missing-waf@internal-source-control-management-api		

Live Manifests: 41 / 41 Risks

Live manifests representing some Kubernetes resource. May include contents of secrets.

ID:	live-manifests
Usage:	devops
Quantity:	very-many
Tags:	none
Origin:	Kubernetes
Owner:	Argo CD User
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	
Processed by:	API Server, Application Controller, ApplicationSet Controller, External Cluster Kubernetes API, Host Cluster Kubernetes API, User CLI
Stored by:	External Cluster Kubernetes API, Host Cluster Kubernetes API
Sent via:	none
Received via:	Reconcile Resource State (Host Cluster), Reconcile Resource State (Host Cluster), Reconcile Resource State (External Cluster), Get/Update/Delete Live Resource State from Kubernetes (Host), Get/Update/Delete Live Resource State from Kubernetes (External)

Data Breach: **probable**

Data Breach Risks: This data asset has data breach potential because of 41 remaining risks:

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@application-controller

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@external-cluster-kubernetes-api

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Possible: cross-site-scripting@api-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@external-cluster-kubernetes-api

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Manifest Sources: 39 / 39 Risks

Some Description

ID:	manifest-sources
Usage:	devops
Quantity:	many
Tags:	none
Origin:	Some Origin
Owner:	Argo CD
Confidentiality:	confidential (rated 4 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	The manifest sources are potentially-sensitive, especially in a multi-tenant Argo CD installation. They shouldn't contain any secrets. Integrity is important to ensure the correct manifests are deployed. The cache helps mitigate denial-of-service on the controller and API server, but Argo CD can still function without the cache.
Processed by:	API Server, Internal Source Control Management UI, Repo Server, Repo Server Storage, User CLI
Stored by:	Internal Source Control Management API, Repo Server, Repo Server Storage
Sent via:	Store Cached Manifest Sources, Push Manifest Sources, Make Requests to API Server
Received via:	Store Cached Manifest Sources, Push Manifest Sources, Fetch Manifest Sources
Data Breach:	probable
Data Breach Risks:	This data asset has data breach potential because of 39 remaining risks:

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@repo-server-storage

Probable: container-baseimage-backdooring@repo-server

Probable: missing-file-validation@repo-server

Probable: path-traversal@repo-server@repo-server-storage@repo-server>store-cached-manifest-sources

Possible: cross-site-scripting@api-server

Possible: cross-site-scripting@internal-source-control-management-ui

Possible: missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server

Possible: missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible:

server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@internal-source-control-management-api

Improbable: unencrypted-asset@repo-server-storage

OIDC Public Keys: 43 / 43 Risks

Public keys used to validate OIDC tokens.

ID:	oidc-public-keys	
Usage:	business	
Quantity:	few	
Tags:	none	
Origin:	OIDC provider	
Owner:		
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	mission-critical	(rated 5 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, OIDC Provider (External), OIDC Proxy (Dex)	
Stored by:	API Server, OIDC Provider (External), OIDC Proxy (Dex)	
Sent via:	none	
Received via:	Validate External OIDC Token, Validate Dex OIDC Token	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 43 remaining risks:	
	Probable: container-baseimage-backdooring@api-server	
	Probable: container-baseimage-backdooring@dex-server	
	Possible: cross-site-scripting@api-server	
	Possible: cross-site-scripting@oidc-provider	
	Possible: cross-site-scripting@dex-server	
	Possible: missing-authentication@api-server>validate-dex-oidc-token@api-server@dex-server	
	Possible: missing-authentication@api-server>validate-external-oidc-token@api-server@oidc-provider	
	Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server	
	Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config	
	Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials	
	Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token	
	Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token	
	Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image	
	Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache	
	Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster	
	Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image	
	Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests	
	Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster	
	Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull	
	Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image	

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Improbable: cross-site-request-forgery@oidc-provider@api-server>validate-external-oidc-token

Improbable: cross-site-request-forgery@dex-server@api-server>validate-dex-oidc-token

Improbable: missing-hardening@api-server

Improbable: missing-identity-provider-isolation@oidc-provider

Improbable: missing-identity-provider-isolation@dex-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@oidc-provider

OIDC Tokens: 44 / 44 Risks

JWTs holding user information, including group membership.

ID:	oidc-tokens	
Usage:	business	
Quantity:	very-many	
Tags:	none	
Origin:	OIDC provider	
Owner:		
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	operational	(rated 2 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, OIDC Provider (External), OIDC Proxy (Dex), User CLI	
Stored by:	User CLI	
Sent via:	Make Requests to API Server	
Received via:	Proxying to an External OIDC Provider	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 44 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@dex-server

Possible: cross-site-scripting@api-server

Possible: cross-site-scripting@oidc-provider

Possible: cross-site-scripting@dex-server

Possible: missing-authentication@api-server>validate-dex-oidc-token@api-server@dex-server

Possible: missing-authentication@api-server>validate-external-oidc-token@api-server@oidc-provider

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Improbable: cross-site-request-forgery@oidc-provider@api-server>validate-external-oidc-token

Improbable: cross-site-request-forgery@dex-server@api-server>validate-dex-oidc-token

Improbable: missing-hardening@api-server

Improbable: missing-identity-provider-isolation@oidc-provider

Improbable: missing-identity-provider-isolation@dex-server

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@oidc-provider

Quay Push Token: 14 / 14 Risks

Quay token with push access to the Argo CD repository.

ID:	quay-push-token
Usage:	devops
Quantity:	very-few
Tags:	none
Origin:	Quay
Owner:	Argo CD build team.
Confidentiality:	strictly-confidential (rated 5 in scale of 5)
Integrity:	operational (rated 2 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	
Processed by:	Argo CD Build Pipeline (GitHub Actions), Quay
Stored by:	Argo CD Build Pipeline (GitHub Actions)
Sent via:	Push Image to Quay
Received via:	none
Data Breach:	probable

Data Breach Risks: This data asset has data breach potential because of 14 remaining risks:

Probable: accidental-secret-leak@quay

Probable: code-backdooring@argo-cd-build-pipeline

Probable: code-backdooring@quay

Possible: missing-authentication@api-server>pull-argo-cd-image@api-server@quay

Possible: missing-authentication@application-controller>pull-argo-cd-image@application-controller@quay

Possible: missing-authentication@applicationset-controller>pull-argo-cd-image@applicationset-controller@quay

Possible: missing-authentication@repo-server>pull-argo-cd-image@repo-server@quay

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: unchecked-deployment@argo-cd-build-pipeline

Possible: unchecked-deployment@quay

Possible: unguarded-access-from-internet@quay@argo-cd-build-pipeline@argo-cd-build-pipeline>push-image-to-quay

Rendered Manifests: 46 / 46 Risks

Some Description

ID:	rendered-manifests
Usage:	devops
Quantity:	many
Tags:	none
Origin:	Some Origin
Owner:	Argo CD
Confidentiality:	confidential (rated 4 in scale of 5)
Integrity:	mission-critical (rated 5 in scale of 5)
Availability:	operational (rated 2 in scale of 5)
CIA-Justification:	The manifests are potentially-sensitive, especially in a multi-tenant Argo CD installation. They shouldn't contain any secrets. Integrity is important to ensure the correct manifests are deployed. The cache helps mitigate denial-of-service on the controller and API server, but Argo CD can still function without the cache.
Processed by:	API Server, Application Controller, External Cluster Kubernetes API, Host Cluster Kubernetes API, Repo Server, User CLI
Stored by:	Rendered Manifests Cache (Redis)
Sent via:	Send/Receive Cached Rendered Manifests, Reconcile Resource State (Host Cluster), Reconcile Resource State (External Cluster), Make Requests to API Server, Get/Update/Delete Live Resource State from Kubernetes (Host), Get/Update/Delete Live Resource State from Kubernetes (External)
Received via:	Send/Receive Cached Rendered Manifests, Rendered Manifest Requests, Fetching Rendered Manifests from Cache
Data Breach:	probable
Data Breach Risks:	This data asset has data breach potential because of 46 remaining risks: <ul style="list-style-type: none"> Probable: container-baseimage-backdooring@api-server Probable: container-baseimage-backdooring@application-controller Probable: container-baseimage-backdooring@external-cluster-kubernetes-api Probable: container-baseimage-backdooring@host-cluster-kubernetes-api Probable: container-baseimage-backdooring@rendered-manifests-cache Probable: container-baseimage-backdooring@repo-server Probable: missing-file-validation@repo-server Possible: cross-site-scripting@api-server Possible: missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server Possible: missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server Possible: missing-authentication@repo-server>send-receive-cached-rendered-manifests@repo-server@rendered-manifests-cache Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible: server-side-request-forgery@argo-cd-build-pipeline@argo-cd-source-repo@argo-cd-build-pipeline>pull-source

Possible: server-side-request-forgery@argo-cd-build-pipeline@docker-hub@argo-cd-build-pipeline>pull-base-image-from-docker-hub

Possible: server-side-request-forgery@argo-cd-build-pipeline@quay@argo-cd-build-pipeline>push-image-to-quay

Possible: server-side-request-forgery@argo-cd-maintainer-git-client@argo-cd-source-repo@argo-cd-maintainer-git-client>push-code-tags-to-github

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@external-cluster-kubernetes-api

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Repo Access Credentials: 40 / 40 Risks

Credentials for retrieving manifest sources from a source control manager (git, Helm, etc.).

ID:	repo-access-credentials	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	Argo CD Operator	
Owner:		
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	operational	(rated 2 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	API Server, ApplicationSet Controller, Internal Source Control Management API, Repo Server, User CLI	
Stored by:	Host Cluster Kubernetes API	
Sent via:	Update Repo Access Credentials, Make Requests to API Server, Git Generator Pull, Get/Update/Delete Live Resource State from Kubernetes (Host)	
Received via:	Update Repo Access Credentials, Get/Update/Delete Live Resource State from Kubernetes (Host), Get Repo Access Credentials	
Data Breach:	probable	
Data Breach Risks:	This data asset has data breach potential because of 40 remaining risks:	

Probable: container-baseimage-backdooring@api-server

Probable: container-baseimage-backdooring@applicationset-controller

Probable: container-baseimage-backdooring@host-cluster-kubernetes-api

Probable: container-baseimage-backdooring@repo-server

Probable: missing-file-validation@repo-server

Possible: cross-site-scripting@api-server

Possible: missing-authentication@api-server>fetching-rendered-manifests-from-cache@api-server@repo-server

Possible: missing-authentication@application-controller>rendered-manifest-requests@application-controller@repo-server

Possible: missing-authentication-second-factor@user-cli>make-requests-to-api-server@user-cli@api-server

Possible: server-side-request-forgery@api-server@external-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-external

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>get-update-delete-live-resource-state-from-kubernetes-host

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-cluster-access-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-rbac-config

Possible: server-side-request-forgery@api-server@host-cluster-kubernetes-api@api-server>update-repo-access-credentials

Possible: server-side-request-forgery@api-server@oidc-provider@api-server>validate-external-oidc-token

Possible: server-side-request-forgery@api-server@dex-server@api-server>validate-dex-oidc-token

Possible: server-side-request-forgery@api-server@quay@api-server>pull-argo-cd-image

Possible: server-side-request-forgery@api-server@repo-server@api-server>fetching-rendered-manifests-from-cache

Possible: server-side-request-forgery@application-controller@external-cluster-kubernetes-api@application-controller>reconcile-resource-state-external-cluster

Possible: server-side-request-forgery@application-controller@host-cluster-kubernetes-api@application-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@application-controller@quay@application-controller>pull-argo-cd-image

Possible: server-side-request-forgery@application-controller@repo-server@application-controller>rendered-manifest-requests

Possible: server-side-request-forgery@applicationset-controller@host-cluster-kubernetes-api@applicationset-controller>reconcile-resource-state-host-cluster

Possible: server-side-request-forgery@applicationset-controller@internal-source-control-management-api@applicationset-controller>git-generator-pull

Possible: server-side-request-forgery@applicationset-controller@quay@applicationset-controller>pull-argo-cd-image

Possible:

server-side-request-forgery@internal-source-control-management-ui@internal-source-control-management-api@internal-source-control-management-ui>push-manifest-sources

Possible: server-side-request-forgery@dex-server@oidc-provider@dex-server>proxying-to-an-external-oidc-provider

Possible: server-side-request-forgery@repo-server@host-cluster-kubernetes-api@repo-server>get-repo-access-credentials

Possible: server-side-request-forgery@repo-server@internal-source-control-management-api@repo-server>fetch-manifest-sources

Possible: server-side-request-forgery@repo-server@quay@repo-server>pull-argo-cd-image

Possible: server-side-request-forgery@repo-server@rendered-manifests-cache@repo-server>send-receive-cached-rendered-manifests

Possible: server-side-request-forgery@user-cli@api-server@user-cli>make-requests-to-api-server

Improbable: cross-site-request-forgery@api-server@user-cli>make-requests-to-api-server

Improbable: missing-hardening@api-server

Improbable: missing-hardening@host-cluster-kubernetes-api

Improbable: missing-network-segmentation@api-server

Improbable: missing-waf@api-server

Improbable: missing-waf@host-cluster-kubernetes-api

Improbable: missing-waf@internal-source-control-management-api

Improbable: unnecessary-data-transfer@argo-cd-rbac-config@host-cluster-kubernetes-api@api-server

Git Repo Name: 0 / 0 Risks

Name of a git repo.

ID:	git-repo-name	
Usage:	devops	
Quantity:	many	
Tags:	none	
Origin:	SCM API or Argo CD User	
Owner:		
Confidentiality:	restricted	(rated 3 in scale of 5)
Integrity:	operational	(rated 2 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:		
Processed by:	none	
Stored by:	none	
Sent via:	none	
Received via:	Git Generator Pull	
Data Breach:	none	
Data Breach Risks:	This data asset has no data breach potential.	

Trust Boundaries

In total **3 trust boundaries** have been modeled during the threat modeling process.

Kubernetes Argo CD Namespace

The Kubernetes namespace where Argo CD is deployed.

ID:	kubernetes-argo-cd-namespace
Type:	network-policy-namespace-isolation
Tags:	none
Assets inside:	API Server, Application Controller, ApplicationSet Controller, OIDC Proxy (Dex), Rendered Manifests Cache (Redis), Repo Server, Repo Server Storage
Boundaries nested:	none

Kubernetes Network

Some Description

ID:	kubernetes-network
Type:	network-policy-namespace-isolation
Tags:	none
Assets inside:	Host Cluster Kubernetes API
Boundaries nested:	Kubernetes Argo CD Namespace

Organization Network

For example, a company VPN.

ID:	organization-network
Type:	network-virtual-lan
Tags:	none
Assets inside:	Internal Source Control Management API, Internal Source Control Management UI, User CLI
Boundaries nested:	Kubernetes Network

Shared Runtimes

In total **1 shared runtime** has been modeled during the threat modeling process.

Kubernetes Node

Multiple Argo CD components may share a single node.

ID:	kubernetes-node
Tags:	none
Assets running:	Repo Server, Application Controller

Risk Rules Checked by Threagile

Threagile Version: 1.0.0

Threagile Build Timestamp: 20211121124511

Threagile Execution Timestamp: 20221113231242

Model Filename: /app/work/threagile-stub-model.yaml

Model Hash (SHA256): d9ec4f3f0dc8d9271574afa1f87520a6ae619842af94e7cdded2aa94bdfa7dbf

Threagile (see <https://threagile.io> for more details) is an open-source toolkit for agile threat modeling, created by Christian Schneider (<https://christian-schneider.net>): It allows to model an architecture with its assets in an agile fashion as a YAML file directly inside the IDE. Upon execution of the Threagile toolkit all standard risk rules (as well as individual custom rules if present) are checked against the architecture model. At the time the Threagile toolkit was executed on the model input file the following risk rules were checked:

Accidental Secret Leak

accidental-secret-leak

STRIDE: Information Disclosure

Description: Sourcecode repositories (including their histories) as well as artifact registries can accidentally contain secrets like checked-in or packaged-in passwords, API tokens, certificates, crypto keys, etc.

Detection: In-scope sourcecode repositories and artifact registries.

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Code Backdooring

code-backdooring

STRIDE: Tampering

Description: For each build-pipeline component Code Backdooring risks might arise where attackers compromise the build-pipeline in order to let backdoored artifacts be shipped into production. Aside from direct code backdooring this includes backdooring of dependencies and even of more lower-level build infrastructure, like backdooring compilers (similar to what the XcodeGhost malware did) or dependencies.

Detection: In-scope development relevant technical assets which are either accessed by out-of-scope unmanaged developer clients and/or are directly accessed by any kind of internet-located (non-VPN) component or are themselves directly located on the internet.

Rating: The risk rating depends on the confidentiality and integrity rating of the code being handled and deployed as well as the placement/calling of this technical asset on/from the internet.

Container Base Image Backdooring

container-baseimage-backdooring

STRIDE: Tampering

Description: When a technical asset is built using container technologies, Base Image Backdooring risks might arise where base images and other layers used contain vulnerable components or backdoors.

Detection: In-scope technical assets running as containers.

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets.

Container Platform Escape

container-platform-escape

STRIDE: Elevation of Privilege

Description: Container platforms are especially interesting targets for attackers as they host big parts of a containerized runtime infrastructure. When not configured and operated with security best practices in mind, attackers might exploit a vulnerability inside an container and escape towards the platform as highly privileged users. These scenarios might give attackers capabilities to attack every other container as owning the container platform (via container escape attacks) equals to owning every container.

Detection: In-scope container platforms.

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Cross-Site Request Forgery (CSRF)

cross-site-request-forgery

STRIDE: Spoofing

Description: When a web application is accessed via web protocols Cross-Site Request Forgery (CSRF) risks might arise.

Detection: In-scope web applications accessed via typical web access protocols.

Rating: The risk rating depends on the integrity rating of the data sent across the communication link.

Cross-Site Scripting (XSS)

cross-site-scripting

STRIDE: Tampering

Description: For each web application Cross-Site Scripting (XSS) risks might arise. In terms of the overall risk level take other applications running on the same domain into account as well.

Detection: In-scope web applications.

Rating: The risk rating depends on the sensitivity of the data processed or stored in the web application.

DoS-risky Access Across Trust-Boundary

dos-risky-access-across-trust-boundary

STRIDE: Denial of Service

Description: Assets accessed across trust boundaries with critical or mission-critical availability rating are more prone to Denial-of-Service (DoS) risks.

Detection: In-scope technical assets (excluding load-balancer) with availability rating of critical or higher which have incoming data-flows across a network trust-boundary (excluding devops usage).

Rating: Matching technical assets with availability rating of critical or higher are at low risk. When the availability rating is mission-critical and neither a VPN nor IP filter for the incoming data-flow nor redundancy for the asset is applied, the risk-rating is considered medium.

Incomplete Model**incomplete-model**

STRIDE: Information Disclosure

Description: When the threat model contains unknown technologies or transfers data over unknown protocols, this is an indicator for an incomplete model.

Detection: All technical assets and communication links with technology type or protocol type specified as unknown.

Rating: low

LDAP-Injection**ldap-injection**

STRIDE: Tampering

Description: When an LDAP server is accessed LDAP-Injection risks might arise. The risk rating depends on the sensitivity of the LDAP server itself and of the data assets processed or stored.

Detection: In-scope clients accessing LDAP servers via typical LDAP access protocols.

Rating: The risk rating depends on the sensitivity of the LDAP server itself and of the data assets processed or stored.

Missing Authentication**missing-authentication**

STRIDE: Elevation of Privilege

Description: Technical assets (especially multi-tenant systems) should authenticate incoming requests when the asset processes or stores sensitive data.

Detection: In-scope technical assets (except load-balancer, reverse-proxy, service-registry, waf, ids, and ips and in-process calls) should authenticate incoming requests when the asset processes or stores sensitive data. This is especially the case for all multi-tenant assets (there even non-sensitive ones).

Rating: The risk rating (medium or high) depends on the sensitivity of the data sent across

the communication link. Monitoring callers are exempted from this risk.

Missing Two-Factor Authentication (2FA)

missing-authentication-second-factor

STRIDE: Elevation of Privilege

Description: Technical assets (especially multi-tenant systems) should authenticate incoming requests with two-factor (2FA) authentication when the asset processes or stores highly sensitive data (in terms of confidentiality, integrity, and availability) and is accessed by humans.

Detection: In-scope technical assets (except load-balancer, reverse-proxy, waf, ids, and ips) should authenticate incoming requests via two-factor authentication (2FA) when the asset processes or stores highly sensitive data (in terms of confidentiality, integrity, and availability) and is accessed by a client used by a human user.

Rating: medium

Missing Build Infrastructure

missing-build-infrastructure

STRIDE: Tampering

Description: The modeled architecture does not contain a build infrastructure (devops-client, sourcecode-repo, build-pipeline, etc.), which might be the risk of a model missing critical assets (and thus not seeing their risks). If the architecture contains custom-developed parts, the pipeline where code gets developed and built needs to be part of the model.

Detection: Models with in-scope custom-developed parts missing in-scope development (code creation) and build infrastructure components (devops-client, sourcecode-repo, build-pipeline, etc.).

Rating: The risk rating depends on the highest sensitivity of the in-scope assets running custom-developed parts.

Missing Cloud Hardening

missing-cloud-hardening

STRIDE: Tampering

Description: Cloud components should be hardened according to the cloud vendor best practices. This affects their configuration, auditing, and further areas.

Detection: In-scope cloud components (either residing in cloud trust boundaries or more specifically tagged with cloud provider types).

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Missing File Validation

missing-file-validation

STRIDE: Spoofing

- Description:** When a technical asset accepts files, these input files should be strictly validated about filename and type.
- Detection:** In-scope technical assets with custom-developed code accepting file data formats.
- Rating:** The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Missing Hardening

missing-hardening

- STRIDE:** Tampering
- Description:** Technical assets with a Relative Attacker Attractiveness (RAA) value of 55 % or higher should be explicitly hardened taking best practices and vendor hardening guides into account.
- Detection:** In-scope technical assets with RAA values of 55 % or higher. Generally for high-value targets like datastores, application servers, identity providers and ERP systems this limit is reduced to 40 %
- Rating:** The risk rating depends on the sensitivity of the data processed or stored in the technical asset.

Missing Identity Propagation

missing-identity-propagation

- STRIDE:** Elevation of Privilege
- Description:** Technical assets (especially multi-tenant systems), which usually process data for endusers should authorize every request based on the identity of the enduser when the data flow is authenticated (i.e. non-public). For DevOps usages at least a technical-user authorization is required.
- Detection:** In-scope service-like technical assets which usually process data based on enduser requests, if authenticated (i.e. non-public), should authorize incoming requests based on the propagated enduser identity when their rating is sensitive. This is especially the case for all multi-tenant assets (there even less-sensitive rated ones). DevOps usages are exempted from this risk.
- Rating:** The risk rating (medium or high) depends on the confidentiality, integrity, and availability rating of the technical asset.

Missing Identity Provider Isolation

missing-identity-provider-isolation

- STRIDE:** Elevation of Privilege
- Description:** Highly sensitive identity provider assets and their identity datastores should be isolated from other assets by their own network segmentation trust-boundary (execution-environment boundaries do not count as network isolation).
- Detection:** In-scope identity provider assets and their identity datastores when surrounded by other (not identity-related) assets (without a network trust-boundary in-between).

This risk is especially prevalent when other non-identity related assets are within the same execution environment (i.e. same database or same application server).

Rating: Default is high impact. The impact is increased to very-high when the asset missing the trust-boundary protection is rated as strictly-confidential or mission-critical.

Missing Identity Store

missing-identity-store

STRIDE: Spoofing

Description: The modeled architecture does not contain an identity store, which might be the risk of a model missing critical assets (and thus not seeing their risks).

Detection: Models with authenticated data-flows authorized via enduser-identity missing an in-scope identity store.

Rating: The risk rating depends on the sensitivity of the enduser-identity authorized technical assets and their data assets processed and stored.

Missing Network Segmentation

missing-network-segmentation

STRIDE: Elevation of Privilege

Description: Highly sensitive assets and/or datastores residing in the same network segment than other lower sensitive assets (like webserver or content management systems etc.) should be better protected by a network segmentation trust-boundary.

Detection: In-scope technical assets with high sensitivity and RAA values as well as datastores when surrounded by assets (without a network trust-boundary in-between) which are of type client-system, web-server, web-application, cms, web-service-rest, web-service-soap, build-pipeline, sourcecode-repository, monitoring, or similar and there is no direct connection between these (hence no requirement to be so close to each other).

Rating: Default is low risk. The risk is increased to medium when the asset missing the trust-boundary protection is rated as strictly-confidential or mission-critical.

Missing Vault (Secret Storage)

missing-vault

STRIDE: Information Disclosure

Description: In order to avoid the risk of secret leakage via config files (when attacked through vulnerabilities being able to read files like Path-Traversal and others), it is best practice to use a separate hardened process with proper authentication, authorization, and audit logging to access config secrets (like credentials, private keys, client certificates, etc.). This component is usually some kind of Vault.

Detection: Models without a Vault (Secret Storage).

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Missing Vault Isolation

missing-vault-isolation

STRIDE: Elevation of Privilege

Description: Highly sensitive vault assets and their datastores should be isolated from other assets by their own network segmentation trust-boundary (execution-environment boundaries do not count as network isolation).

Detection: In-scope vault assets when surrounded by other (not vault-related) assets (without a network trust-boundary in-between). This risk is especially prevalent when other non-vault related assets are within the same execution environment (i.e. same database or same application server).

Rating: Default is medium impact. The impact is increased to high when the asset missing the trust-boundary protection is rated as strictly-confidential or mission-critical.

Missing Web Application Firewall (WAF)

missing-waf

STRIDE: Tampering

Description: To have a first line of filtering defense, security architectures with web-services or web-applications should include a WAF in front of them. Even though a WAF is not a replacement for security (all components must be secure even without a WAF) it adds another layer of defense to the overall system by delaying some attacks and having easier attack alerting through it.

Detection: In-scope web-services and/or web-applications accessed across a network trust boundary not having a Web Application Firewall (WAF) in front of them.

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Mixed Targets on Shared Runtime

mixed-targets-on-shared-runtime

STRIDE: Elevation of Privilege

Description: Different attacker targets (like frontend and backend/datastore components) should not be running on the same shared (underlying) runtime.

Detection: Shared runtime running technical assets of different trust-boundaries is at risk. Also mixing backend/datastore with frontend components on the same shared runtime is considered a risk.

Rating: The risk rating (low or medium) depends on the confidentiality, integrity, and availability rating of the technical asset running on the shared runtime.

Path-Traversal

path-traversal

STRIDE: Information Disclosure

Description: When a filesystem is accessed Path-Traversal or Local-File-Inclusion (LFI) risks might arise. The risk rating depends on the sensitivity of the technical asset itself

and of the data assets processed or stored.

Detection: Filesystems accessed by in-scope callers.

Rating: The risk rating depends on the sensitivity of the data stored inside the technical asset.

Push instead of Pull Deployment

push-instead-of-pull-deployment

STRIDE: Tampering

Description: When comparing push-based vs. pull-based deployments from a security perspective, pull-based deployments improve the overall security of the deployment targets. Every exposed interface of a production system to accept a deployment increases the attack surface of the production system, thus a pull-based approach exposes less attack surface relevant interfaces.

Detection: Models with build pipeline components accessing in-scope targets of deployment (in a non-readonly way) which are not build-related components themselves.

Rating: The risk rating depends on the highest sensitivity of the deployment targets running custom-developed parts.

Search-Query Injection

search-query-injection

STRIDE: Tampering

Description: When a search engine server is accessed Search-Query Injection risks might arise.

Detection: In-scope clients accessing search engine servers via typical search access protocols.

Rating: The risk rating depends on the sensitivity of the search engine server itself and of the data assets processed or stored.

Server-Side Request Forgery (SSRF)

server-side-request-forgery

STRIDE: Information Disclosure

Description: When a server system (i.e. not a client) is accessing other server systems via typical web protocols Server-Side Request Forgery (SSRF) or Local-File-Inclusion (LFI) or Remote-File-Inclusion (RFI) risks might arise.

Detection: In-scope non-client systems accessing (using outgoing communication links) targets with either HTTP or HTTPS protocol.

Rating: The risk rating (low or medium) depends on the sensitivity of the data assets receivable via web protocols from targets within the same network trust-boundary as well on the sensitivity of the data assets receivable via web protocols from the target asset itself. Also for cloud-based environments the exploitation impact is at least medium, as cloud backend services can be attacked via SSRF.

Service Registry Poisoning

service-registry-poisoning**STRIDE:** Spoofing**Description:** When a service registry used for discovery of trusted service endpoints Service Registry Poisoning risks might arise.**Detection:** In-scope service registries.**Rating:** The risk rating depends on the sensitivity of the technical assets accessing the service registry as well as the data assets processed or stored.**SQL/NoSQL-Injection****sql-nosql-injection****STRIDE:** Tampering**Description:** When a database is accessed via database access protocols SQL/NoSQL-Injection risks might arise. The risk rating depends on the sensitivity technical asset itself and of the data assets processed or stored.**Detection:** Database accessed via typical database access protocols by in-scope clients.**Rating:** The risk rating depends on the sensitivity of the data stored inside the database.**Unchecked Deployment****unchecked-deployment****STRIDE:** Tampering**Description:** For each build-pipeline component Unchecked Deployment risks might arise when the build-pipeline does not include established DevSecOps best-practices. DevSecOps best-practices scan as part of CI/CD pipelines for vulnerabilities in source- or byte-code, dependencies, container layers, and dynamically against running test systems. There are several open-source and commercial tools existing in the categories DAST, SAST, and IAST.**Detection:** All development-relevant technical assets.**Rating:** The risk rating depends on the highest rating of the technical assets and data assets processed by deployment-receiving targets.**Unencrypted Technical Assets****unencrypted-asset****STRIDE:** Information Disclosure**Description:** Due to the confidentiality rating of the technical asset itself and/or the processed data assets this technical asset must be encrypted. The risk rating depends on the sensitivity technical asset itself and of the data assets stored.**Detection:** In-scope unencrypted technical assets (excluding reverse-proxy, load-balancer, waf, ids, ips and embedded components like library) storing data assets rated at least as confidential or critical. For technical assets storing data assets rated as strictly-confidential or mission-critical the encryption must be of type data-with-enduser-individual-key.

Rating: Depending on the confidentiality rating of the stored data-assets either medium or high risk.

Unencrypted Communication

unencrypted-communication

STRIDE: Information Disclosure

Description: Due to the confidentiality and/or integrity rating of the data assets transferred over the communication link this connection must be encrypted.

Detection: Unencrypted technical communication links of in-scope technical assets (excluding monitoring traffic as well as local-file-access and in-process-library-call) transferring sensitive data.

Rating: Depending on the confidentiality rating of the transferred data-assets either medium or high risk.

Unguarded Access From Internet

unguarded-access-from-internet

STRIDE: Elevation of Privilege

Description: Internet-exposed assets must be guarded by a protecting service, application, or reverse-proxy.

Detection: In-scope technical assets (excluding load-balancer) with confidentiality rating of confidential (or higher) or with integrity rating of critical (or higher) when accessed directly from the internet. All web-server, web-application, reverse-proxy, waf, and gateway assets are exempted from this risk when they do not consist of custom developed code and the data-flow only consists of HTTP or FTP protocols. Access from monitoring systems as well as VPN-protected connections are exempted.

Rating: The matching technical assets are at low risk. When either the confidentiality rating is strictly-confidential or the integrity rating is mission-critical, the risk-rating is considered medium. For assets with RAA values higher than 40 % the risk-rating increases.

Unguarded Direct Datastore Access

unguarded-direct-datastore-access

STRIDE: Elevation of Privilege

Description: Datastores accessed across trust boundaries must be guarded by some protecting service or application.

Detection: In-scope technical assets of type datastore (except identity-store-ldap when accessed from identity-provider and file-server when accessed via file transfer protocols) with confidentiality rating of confidential (or higher) or with integrity rating of critical (or higher) which have incoming data-flows from assets outside across a network trust-boundary. DevOps config and deployment access is excluded from this risk.

Rating: The matching technical assets are at low risk. When either the confidentiality rating is strictly-confidential or the integrity rating is mission-critical, the risk-rating is considered medium. For assets with RAA values higher than 40 % the risk-rating increases.

Unnecessary Communication Link

unnecessary-communication-link

STRIDE: Elevation of Privilege

Description: When a technical communication link does not send or receive any data assets, this is an indicator for an unnecessary communication link (or for an incomplete model).

Detection: In-scope technical assets' technical communication links not sending or receiving any data assets.

Rating: low

Unnecessary Data Asset

unnecessary-data-asset

STRIDE: Elevation of Privilege

Description: When a data asset is not processed or stored by any data assets and also not transferred by any communication links, this is an indicator for an unnecessary data asset (or for an incomplete model).

Detection: Modelled data assets not processed or stored by any data assets and also not transferred by any communication links.

Rating: low

Unnecessary Data Transfer

unnecessary-data-transfer

STRIDE: Elevation of Privilege

Description: When a technical asset sends or receives data assets, which it neither processes or stores this is an indicator for unnecessarily transferred data (or for an incomplete model). When the unnecessarily transferred data assets are sensitive, this poses an unnecessary risk of an increased attack surface.

Detection: In-scope technical assets sending or receiving sensitive data assets which are neither processed nor stored by the technical asset are flagged with this risk. The risk rating (low or medium) depends on the confidentiality, integrity, and availability rating of the technical asset. Monitoring data is exempted from this risk.

Rating: The risk assessment is depending on the confidentiality and integrity rating of the transferred data asset either low or medium.

Unnecessary Technical Asset

unnecessary-technical-asset

STRIDE: Elevation of Privilege

Description: When a technical asset does not process or store any data assets, this is an

indicator for an unnecessary technical asset (or for an incomplete model). This is also the case if the asset has no communication links (either outgoing or incoming).

Detection: Technical assets not processing or storing any data assets.

Rating: low

Untrusted Deserialization

untrusted-deserialization

STRIDE: Tampering

Description: When a technical asset accepts data in a specific serialized form (like Java or .NET serialization), Untrusted Deserialization risks might arise.

Detection: In-scope technical assets accepting serialization data formats (including EJB and RMI protocols).

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

Wrong Communication Link Content

wrong-communication-link-content

STRIDE: Information Disclosure

Description: When a communication link is defined as readonly, but does not receive any data asset, or when it is defined as not readonly, but does not send any data asset, it is likely to be a model failure.

Detection: Communication links with inconsistent data assets being sent/received not matching their readonly flag or otherwise inconsistent protocols not matching the target technology type.

Rating: low

Wrong Trust Boundary Content

wrong-trust-boundary-content

STRIDE: Elevation of Privilege

Description: When a trust boundary of type network-policy-namespace-isolation contains non-container assets it is likely to be a model failure.

Detection: Trust boundaries which should only contain containers, but have different assets inside.

Rating: low

XML External Entity (XXE)

xml-external-entity

STRIDE: Information Disclosure

Description: When a technical asset accepts data in XML format, XML External Entity (XXE) risks might arise.

Detection: In-scope technical assets accepting XML data formats.

Rating: The risk rating depends on the sensitivity of the technical asset itself and of the data

assets processed and stored. Also for cloud-based environments the exploitation impact is at least medium, as cloud backend services can be attacked via SSRF (and XXE vulnerabilities are often also SSRF vulnerabilities).

Disclaimer

Argoproj Maintainers conducted this threat analysis using the open-source Threagile toolkit on the applications and systems that were modeled as of this report's date. Information security threats are continually changing, with new vulnerabilities discovered on a daily basis, and no application can ever be 100% secure no matter how much threat modeling is conducted. It is recommended to execute threat modeling and also penetration testing on a regular basis (for example yearly) to ensure a high ongoing level of security and constantly check for new attack vectors.

This report cannot and does not protect against personal or business loss as the result of use of the applications or systems described. Argoproj Maintainers and the Threagile toolkit offers no warranties, representations or legal certifications concerning the applications or systems it tests. All software includes defects: nothing in this document is intended to represent or warrant that threat modeling was complete and without error, nor does this document represent or warrant that the architecture analyzed is suitable to task, free of other defects than reported, fully compliant with any industry standards, or fully compatible with any operating system, hardware, or other application. Threat modeling tries to analyze the modeled architecture without having access to a real working system and thus cannot and does not test the implementation for defects and vulnerabilities. These kinds of checks would only be possible with a separate code review and penetration test against a working system and not via a threat model.

By using the resulting information you agree that Argoproj Maintainers and the Threagile toolkit shall be held harmless in any event.

This report is confidential and intended for internal, confidential use by the client. The recipient is obligated to ensure the highly confidential contents are kept secret. The recipient assumes responsibility for further distribution of this document.

In this particular project, a timebox approach was used to define the analysis effort. This means that the author allotted a prearranged amount of time to identify and document threats. Because of this, there is no guarantee that all possible threats and risks are discovered. Furthermore, the analysis applies to a snapshot of the current state of the modeled architecture (based on the architecture information provided by the customer) at the examination time.

Report Distribution

Distribution of this report (in full or in part like diagrams or risk findings) requires that this disclaimer as well as the chapter about the Threagile toolkit and method used is kept intact as part of the distributed report or referenced from the distributed parts.