Threagile
Agile Threat Modeling

Threat Model Report
DfE SSPHP Threat Model

2 May 2024

Sam Pritchard

Table of Contents

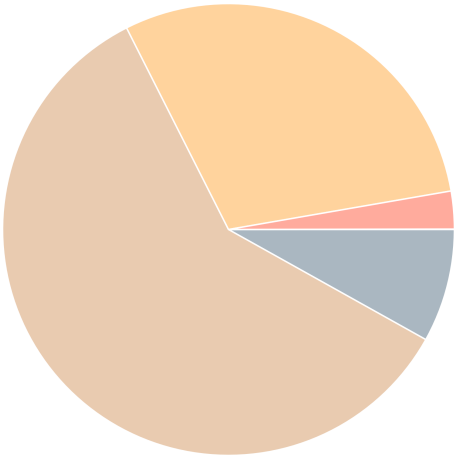
Management Summary

Threagile toolkit was used to model the architecture of "DfE SSPHP Threat Model" 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 "DfE SSPHP Threat Model" whether the mitigation advices have been applied or not.

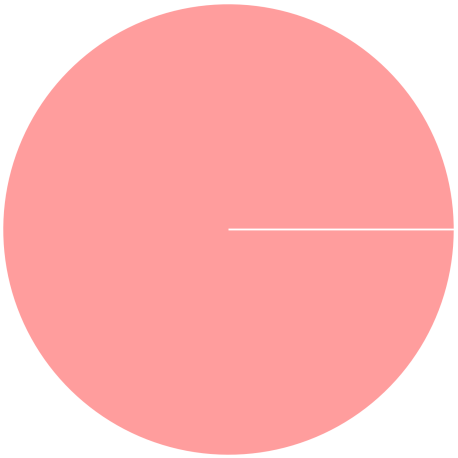
Each risk finding references a chapter of the OWASP ASVS (Application Security Verification Standard) audit checklist. The OWASP ASVS checklist should be considered as an inspiration by architects and developers to further harden the application in a Defense-in-Depth approach. Additionally, for each 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 **37 initial risks** in **11 categories** have been identified during the threat modeling process:

1 critical risk
0 high risk
11 elevated risk
22 medium risk
3 low risk



37 unchecked
0 in discussion
0 accepted
0 in progress
0 *mitigated*
0 *false positive*



Important security data is ingested to show a view of the world in DfE.

Impact Analysis of 37 Initial Risks in 11 Categories

The most prevalent impacts of the **37 initial risks** (distributed over **11 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.

Critical: **Information Dislosure**: 1 Initial Risk - Exploitation likelihood is *Likely* with *High* impact.
Serious reputational damage, potential targeting of sensitive assets.

Elevated: **Missing Cloud Hardening**: 3 Initial Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
If this risk is unmitigated, attackers might access cloud components in an unintended way.

Elevated: **Missing Hardening**: 3 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: **Server-Side Request Forgery (SSRF)**: 9 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**: 8 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**: 1 Initial Risk - Exploitation likelihood is *Unlikely* with *Medium* 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**: 3 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: **Missing Vault Isolation**: 1 Initial Risk - Exploitation likelihood is *Unlikely* with *High* impact.
If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards highly sensitive vault assets and their datastores, as they are not separated by network segmentation.

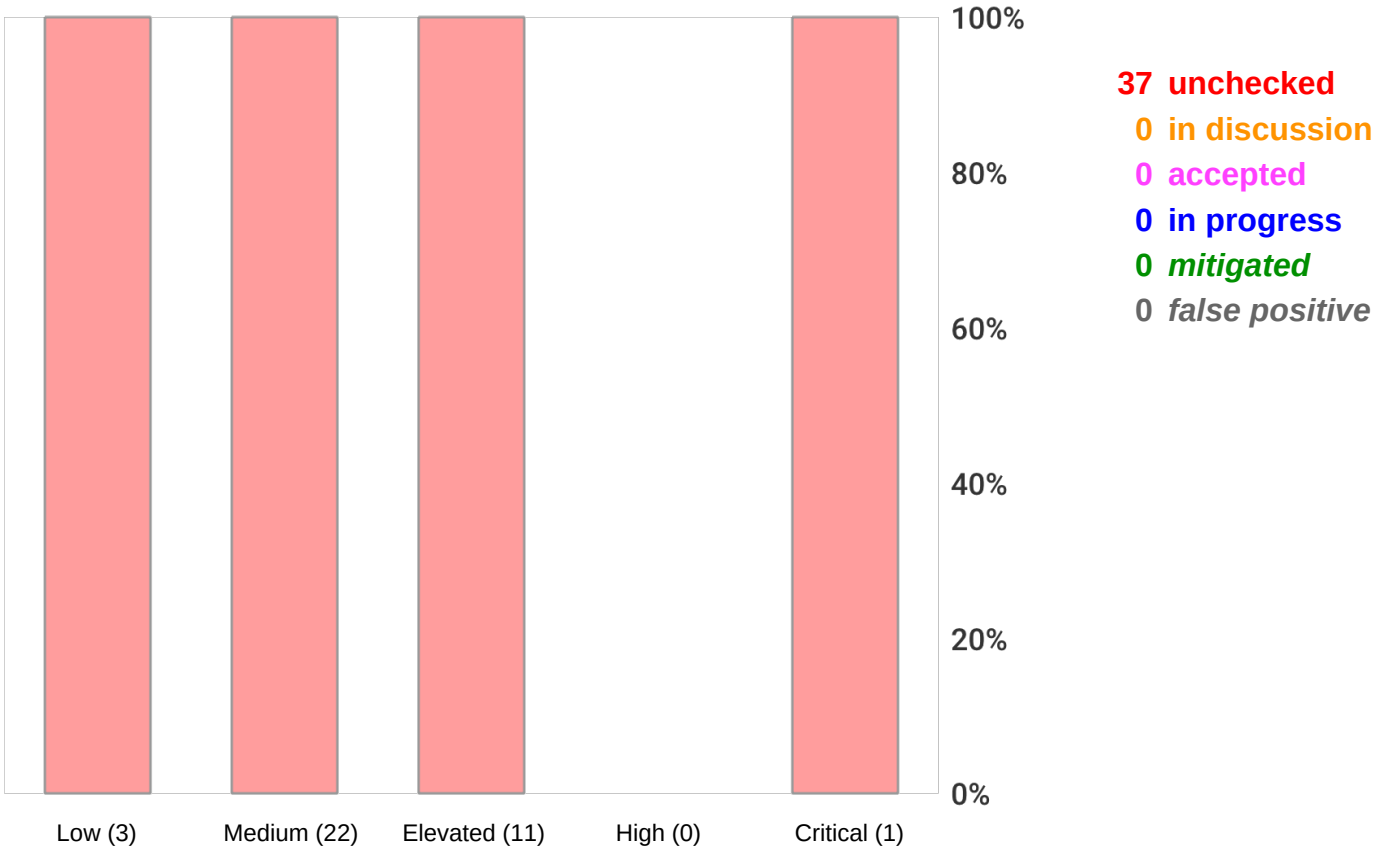
Medium: **Push instead of Pull Deployment**: 1 Initial Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.
If this risk is unmitigated, attackers might have more potential target vectors for attacks, as the overall attack surface is unnecessarily increased.

Medium: **Unchecked Deployment**: 3 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**: 4 Initial Risks - 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.

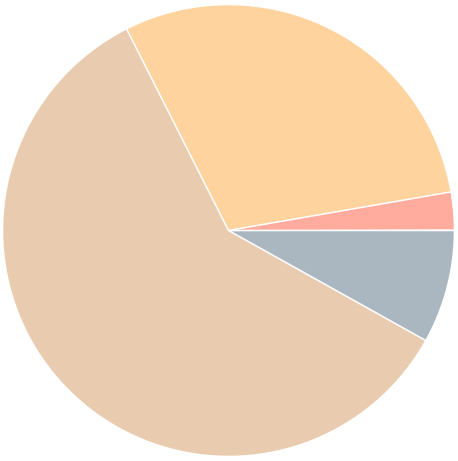
Risk Mitigation

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

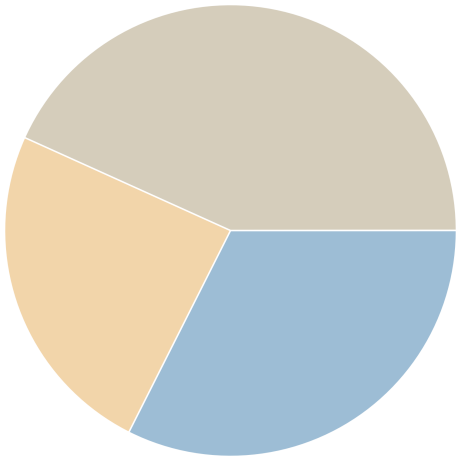


After removal of risks with status *mitigated* and *false positive* the following **37 remain unmitigated**:

1 unmitigated critical risk
0 unmitigated high risk
11 unmitigated elevated risk
22 unmitigated medium risk
3 unmitigated low risk



0 business side related
12 architecture related
9 development related
16 operations related



Impact Analysis of 37 Remaining Risks in 11 Categories

The most prevalent impacts of the **37 remaining risks** (distributed over **11 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.

Critical: **Information Dislosure**: 1 Remaining Risk - Exploitation likelihood is *Likely* with *High* impact.
Serious reputational damage, potential targeting of sensitive assets.

Elevated: **Missing Cloud Hardening**: 3 Remaining Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
If this risk is unmitigated, attackers might access cloud components in an unintended way.

Elevated: **Missing Hardening**: 3 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: **Server-Side Request Forgery (SSRF)**: 9 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**: 8 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**: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* 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**: 3 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: **Missing Vault Isolation**: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *High* impact.
If this risk is unmitigated, attackers successfully attacking other components of the system might have an easy path towards highly sensitive vault assets and their datastores, as they are not separated by network segmentation.

Medium: **Push instead of Pull Deployment**: 1 Remaining Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.
If this risk is unmitigated, attackers might have more potential target vectors for attacks, as the overall attack surface is unnecessarily increased.

Medium: **Unchecked Deployment**: 3 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**: 4 Remaining Risks - 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.

Application Overview

Business Criticality

The overall business criticality of "DfE SSPHP Threat Model" was rated as:

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

Business Overview

A test threagile run based on the DfE SSPHP Metrics continuous assurance platform.

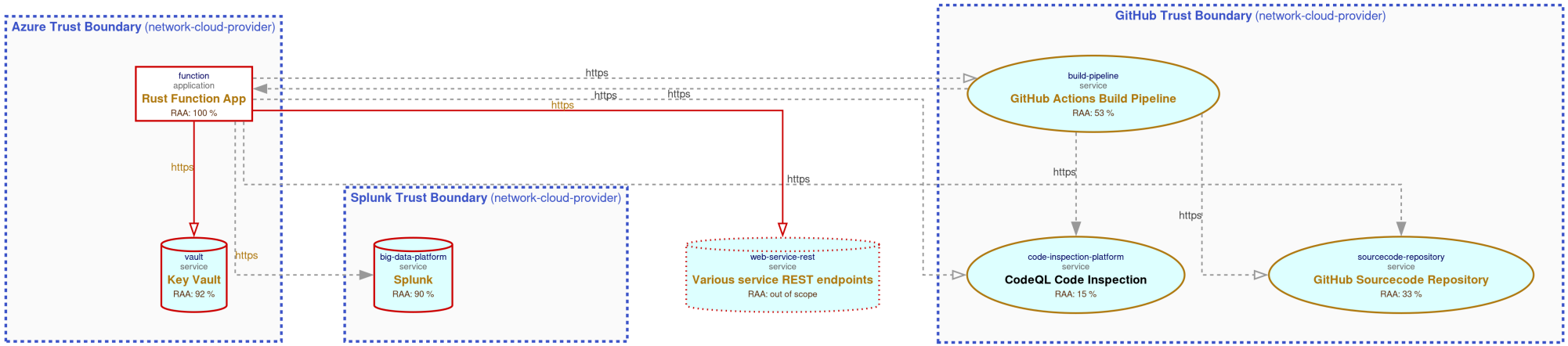
Technical Overview

The application involves a Rust function app that takes care of ingesting data from multiple sources, and then pushes the data to SplunkCloud.

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 - DfE SSPHP Threat Model



Security Requirements

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

Strict key management
Ensure that keys are properly secured and managed to ensure that only the Rust function app can access them.

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.

Data-Poisoning
As an attacker, I'd like to forward data to SplunkCloud to poison the data and cause reputational damage.

Denial-of-Service
As an attacker, I'd like to disrupt the service to prevent data from being ingested.

Information-Dislosure
As an attacker, I'd like to access SplunkCloud to view the dashboards containing the risk and compliance levels of sensitive services.

Sensitive-Key-Compromise
As an attacker, I would like to steal sensitive keys used by the Rust ingester to enable me to get hold of sensitive data regarding the DfE estate.

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.

azure
Retrieve keys, API Calls to Services, Various service REST endpoints, Azure Trust Boundary

azure-function-app
Function App Push, Rust Function App, Azure Trust Boundary

azure-key-vault
Key Vault, Secrets and API Keys, Azure Trust Boundary

codeql
CodeQL Code Inspection, Code Inspection Platform Traffic, Code Inspection Platform Traffic

git
Sourcecode Repository Traffic, Sourcecode Repository Traffic, Sourcecode

github
Sourcecode Repository Traffic, GitHub Sourcecode Repository, Sourcecode Repository Traffic, API Calls to Services, Sourcecode, GitHub Trust Boundary

github-actions
GitHub Actions Build Pipeline, Build Pipeline Traffic

o365
API Calls to Services

splunk
Send data to Splunk, Retrieve keys, Splunk, Splunk data, Splunk Trust Boundary

STRIDE Classification of Identified Risks

This chapter clusters and classifies the risks by STRIDE categories: In total **37 potential risks** have been identified during the threat modeling process of which **0 in the Spoofing** category, **13 in the Tampering** category, **0 in the Repudiation** category, **15 in the Information Disclosure** category, **0 in the Denial of Service** category, and **9 in the Elevation of Privilege** category.

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

Spoofing

n/a

Tampering

Elevated: **Missing Cloud Hardening**: 3 / 3 Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
Cloud components should be hardened according to the cloud vendor best practices. This affects their configuration, auditing, and further areas.

Elevated: **Missing Hardening**: 3 / 3 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**: 3 / 3 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: **Push instead of Pull Deployment**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.
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.

Medium: **Unchecked Deployment**: 3 / 3 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

Critical: **Information Dislosure**: 1 / 1 Risk - Exploitation likelihood is *Likely* with *High* impact.
The biggest risk to SSPHP is the leaking of sensitive system data, this is most likely to happen within Splunk due to the high number of users, including those from a third party supplier.

Elevated: **Server-Side Request Forgery (SSRF)**: 9 / 9 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**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* 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: **Unencrypted Technical Assets**: 4 / 4 Risks - 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: **Unguarded Access From Internet**: 8 / 8 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 Vault Isolation**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *High* impact.
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).

Assignment by Function

This chapter clusters and assigns the risks by functions which are most likely able to check and mitigate them: In total **37 potential risks** have been identified during the threat modeling process of which **0 should be checked by Business Side**, **12 should be checked by Architecture**, **9 should be checked by Development**, and **16 should be checked by Operations**.

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

Business Side

n/a

Architecture

Elevated: **Unguarded Access From Internet**: 8 / 8 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: **Push instead of Pull Deployment**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

Try to prefer pull-based deployments (like GitOps scenarios offer) over push-based deployments to reduce the attack surface of the production system.

Medium: **Unchecked Deployment**: 3 / 3 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.

Development

Elevated: **Server-Side Request Forgery (SSRF)**: 9 / 9 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

Critical: **Information Dislosure**: 1 / 1 Risk - Exploitation likelihood is *Likely* with *High* impact.
Restrict number of users, continuously audit users, implement robust leavers processes, audit and alert on sensitive actions, hold audit logs for 90+ days, follow principle of least privilege, enforce SSO where possible, where SSO not possible - enforce MFA.

Elevated: **Missing Cloud Hardening**: 3 / 3 Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
Apply hardening of all cloud components and services, taking special care to follow the individual risk descriptions (which depend on the cloud provider tags in the model).

Elevated: **Missing Hardening**: 3 / 3 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).

Medium: **Accidental Secret Leak**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* 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**: 3 / 3 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: **Missing Vault Isolation**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *High* impact.
Apply a network segmentation trust-boundary around the highly sensitive vault assets and their datastores.

Medium: **Unencrypted Technical Assets**: 4 / 4 Risks - 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.

Rust Function App: RAA 100%
The Rust Function App collects data from Azure, O365, AWS, GitHub.

Key Vault: RAA 92%
The key vault containing keys for sensitive systems, and the Splunk data ingestion HEC token.

Splunk: RAA 90%
All data from the function app ends up in Splunk for analysis.

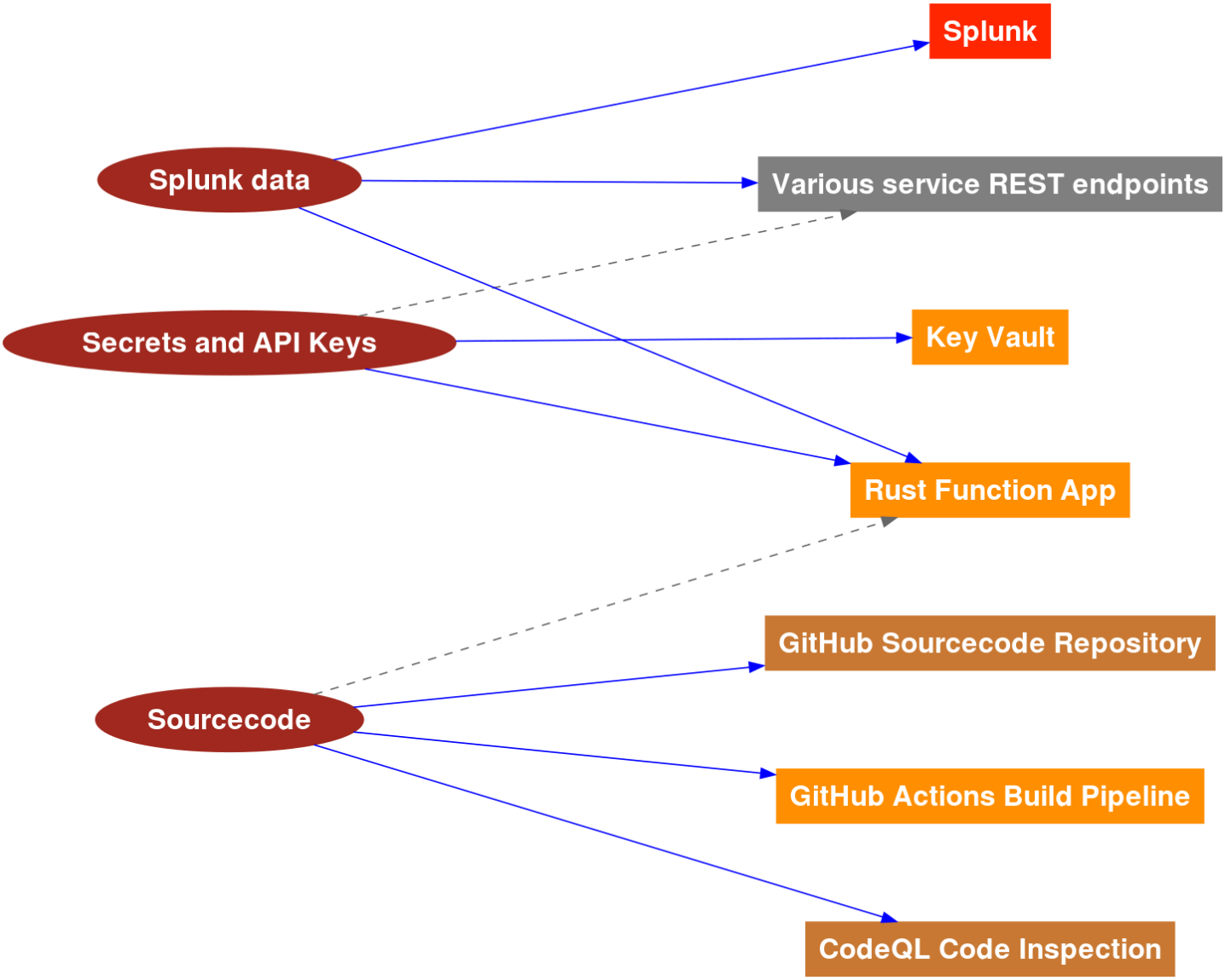
GitHub Actions Build Pipeline: RAA 53%
GitHub Actions Build Pipeline

GitHub Sourcecode Repository: RAA 33%
github Sourcecode Repository

CodeQL Code Inspection: RAA 15%
CodeQL Code Inspection

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: 1 Asset

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.

Various service REST endpoints: out-of-scope

Not part of the overall system, or managed by the team.

Potential Model Failures: 1 / 1 Risk

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: **Push instead of Pull Deployment**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.

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.

Questions: 0 / 0 Questions

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

No custom questions arose during the threat modeling process.

Identified Risks by Vulnerability Category

In total **37 potential risks** have been identified during the threat modeling process of which **1 are rated as critical**, **0 as high**, **11 as elevated**, **22 as medium**, and **3 as low**.

These risks are distributed across **11 vulnerability categories**. The following sub-chapters of this section describe each identified risk category.

Information Dislosure: 1 / 1 Risk

Description (Information Disclosure): CWE 200

The biggest risk to SSPHP is the leaking of sensitive system data, this is most likely to happen within Splunk due to the high number of users, including those from a third party supplier.

Impact

Serious reputational damage, potential targeting of sensitive assets.

Detection Logic

User logs in after X amount of inactivity, User exfiltrates data from Splunk, User does not have MFA, User doesn't log in for X amount of time.

Risk Rating

The likelihood of this happening is medium as users are strictly trusted staff, the impact is high due to the sensitivity of data.

False Positives

Third party company doesn't keep DfE updated with staff churn.

Mitigation (Operations): Gaining unauthorised access to Splunk and retrieving sensitive vulnerability and compliance data.

Restrict number of users, continuously audit users, implement robust leavers processes, audit and alert on sensitive actions, hold audit logs for 90+ days, follow principle of least privilege, enforce SSO where possible, where SSO not possible - enforce MFA.

ASVS Chapter: V4.1 - General Access Control, V2.2 General Authenticator Security
Cheat Sheet: n/a

Check

Risk Findings

The risk **Information Dislosure** 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.

Critical Risk Severity

Example Individual Risk at : Exploitation likelihood is *Likely* with *High* impact.

information-disclosure@splunk@splunk-network@splunk-data

Unchecked

Missing Cloud Hardening: 3 / 3 Risks

Description (Tampering): CWE 1008

Cloud components should be hardened according to the cloud vendor best practices. This affects their configuration, auditing, and further areas.

Impact

If this risk is unmitigated, attackers might access cloud components in an unintended way.

Detection Logic

In-scope cloud components (either residing in cloud trust boundaries or more specifically tagged with cloud provider types).

Risk Rating

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

False Positives

Cloud components not running parts of the target architecture can be considered as false positives after individual review.

Mitigation (Operations): Cloud Hardening

Apply hardening of all cloud components and services, taking special care to follow the individual risk descriptions (which depend on the cloud provider tags in the model).

For **Amazon Web Services (AWS)**: Follow the *CIS Benchmark for Amazon Web Services* (see also the automated checks of cloud audit tools like *"PacBot", "CloudSploit", "CloudMapper", "ScoutSuite", or "Prowler AWS CIS Benchmark Tool"*).
For EC2 and other servers running Amazon Linux, follow the *CIS Benchmark for Amazon Linux* and switch to IMDSv2.
For S3 buckets follow the *Security Best Practices for Amazon S3* at https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html to avoid accidental leakage.
Also take a look at some of these tools: https://github.com/toniblyx/my-arsenal-of-aws-security-tools

For **Microsoft Azure**: Follow the *CIS Benchmark for Microsoft Azure* (see also the automated checks of cloud audit tools like *"CloudSploit" or "ScoutSuite"*).

For **Google Cloud Platform**: Follow the *CIS Benchmark for Google Cloud Computing Platform* (see also the automated checks of cloud audit tools like *"CloudSploit" or "ScoutSuite"*).

For **Oracle Cloud Platform**: Follow the hardening best practices (see also the automated checks of cloud audit tools like *"CloudSploit"*).

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 Cloud Hardening** 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

Missing Cloud Hardening (Azure) risk at **Azure Trust Boundary**: CIS Benchmark for Microsoft Azure: Exploitation likelihood is *Unlikely* with *Very High* impact.

missing-cloud-hardening@azure-network

Unchecked

Missing Cloud Hardening risk at **Splunk Trust Boundary**: Exploitation likelihood is *Unlikely* with *Very High* impact.

missing-cloud-hardening@splunk-network

Unchecked

Medium Risk Severity

Missing Cloud Hardening risk at **GitHub Trust Boundary**: Exploitation likelihood is *Unlikely* with *High* impact.

missing-cloud-hardening@github-network

Unchecked

Missing Hardening: 3 / 3 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 **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

Missing Hardening risk at **Key Vault**: Exploitation likelihood is *Likely* with *Medium* impact.

missing-hardening@key-vault

Unchecked

Missing Hardening risk at **Rust Function App**: Exploitation likelihood is *Likely* with *Medium* impact.

missing-hardening@rust-function-app

Unchecked

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

missing-hardening@splunk

Unchecked

Server-Side Request Forgery (SSRF): 9 / 9 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 **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.

Elevated Risk Severity

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Key Vault** via **Retrieve keys**: Exploitation likelihood is *Likely* with *Medium* impact.

server-side-request-forgery@rust-function-app@key-vault@rust-function-app>retrieve-keys

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Various service REST endpoints** via **API Calls to Services**: Exploitation likelihood is *Likely* with *Medium* impact.

server-side-request-forgery@rust-function-app@various-service-rest-endpoints@rust-function-app>api-calls-to-services

Unchecked

Medium Risk Severity

Server-Side Request Forgery (SSRF) risk at **GitHub Actions Build Pipeline** server-side web-requesting the target **CodeQL Code Inspection** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@github-actions-build-pipeline@codeql-code-inspection@github-actions-build-pipeline>code-inspection-platform-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **GitHub Actions Build Pipeline** server-side web-requesting the target **GitHub Sourcecode Repository** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@github-actions-build-pipeline@github-sourcecode-repository@github-actions-build-pipeline>sourcecode-repository-traffic

Unchecked

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

server-side-request-forgery@github-actions-build-pipeline@rust-function-app@github-actions-build-pipeline>function-app-push

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **CodeQL Code Inspection** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@codeql-code-inspection@rust-function-app>code-inspection-platform-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **GitHub Actions Build Pipeline** via **Build Pipeline Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@github-actions-build-pipeline@rust-function-app>build-pipeline-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **GitHub Sourcecode Repository** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@github-sourcecode-repository@rust-function-app>sourcecode-repository-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Splunk** via **Send data to Splunk**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@splunk@rust-function-app>send-data-to-splunk

Unchecked

Unguarded Access From Internet: 8 / 8 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 **8 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 **GitHub Actions Build Pipeline** by **Rust Function App** via **Build Pipeline Traffic**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@github-actions-build-pipeline@rust-function-app@rust-function-app>build-pipeline-traffic

Unchecked

Unguarded Access from Internet of **Key Vault** by **Rust Function App** via **Retrieve keys**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@key-vault@rust-function-app@rust-function-app>retrieve-keys

Unchecked

Unguarded Access from Internet of **Rust Function App** by **GitHub Actions Build Pipeline** via **Function App Push**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@rust-function-app@github-actions-build-pipeline@github-actions-build-pipeline>function-app-push

Unchecked

Unguarded Access from Internet of **Splunk** by **Rust Function App** via **Send data to Splunk**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@splunk@rust-function-app@rust-function-app>send-data-to-splunk

Unchecked

Medium Risk Severity

Unguarded Access from Internet of **CodeQL Code Inspection** by **GitHub Actions Build Pipeline** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@codeql-code-inspection@github-actions-build-pipeline@github-actions-build-pipeline>code-inspection-platform-traffic

Unchecked

Unguarded Access from Internet of **CodeQL Code Inspection** by **Rust Function App** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@codeql-code-inspection@rust-function-app@rust-function-app>code-inspection-platform-traffic

Unchecked

Unguarded Access from Internet of **GitHub Sourcecode Repository** by **GitHub Actions Build Pipeline** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@github-sourcecode-repository@github-actions-build-pipeline@github-actions-build-pipeline>sourcecode-repository-traffic

Unchecked

Unguarded Access from Internet of **GitHub Sourcecode Repository** by **Rust Function App** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@github-sourcecode-repository@rust-function-app@rust-function-app>sourcecode-repository-traffic

Unchecked

Accidental Secret Leak: 1 / 1 Risk

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 **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

Accidental Secret Leak risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Medium* impact.

accidental-secret-leak@github-sourcecode-repository

Unchecked

Code Backdooring: 3 / 3 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 **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

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

code-backdooring@github-actions-build-pipeline

Unchecked

Code Backdooring risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *High* impact.

code-backdooring@github-sourcecode-repository

Unchecked

Low Risk Severity

Code Backdooring risk at **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Low* impact.

code-backdooring@codeql-code-inspection

Unchecked

Missing Vault Isolation: 1 / 1 Risk

Description (Elevation of Privilege): CWE 1008

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).

Impact

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

Detection Logic

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).

Risk 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.

False Positives

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

Mitigation (Operations): Network Segmentation

Apply a network segmentation trust-boundary around the highly sensitive vault assets and their 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 Vault Isolation** 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 Isolation to further encapsulate and protect vault-related asset **Key Vault** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *High* impact.

missing-vault-isolation@key-vault

Unchecked

Push instead of Pull Deployment: 1 / 1 Risk

Description (Tampering): CWE 1127

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.

Impact

If this risk is unmitigated, attackers might have more potential target vectors for attacks, as the overall attack surface is unnecessarily increased.

Detection Logic

Models with build pipeline components accessing in-scope targets of deployment (in a non-readonly way) which are not build-related components themselves.

Risk Rating

The risk rating depends on the highest sensitivity of the deployment targets running custom-developed parts.

False Positives

Communication links that are not deployment paths can be considered as false positives after individual review.

Mitigation (Architecture): Build Pipeline Hardening

Try to prefer pull-based deployments (like GitOps scenarios offer) over push-based deployments to reduce the attack surface of the production system.

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 **Push instead of Pull Deployment** 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

Push instead of Pull Deployment at **Rust Function App** via build pipeline asset **GitHub Actions Build Pipeline**: Exploitation likelihood is *Unlikely* with *Medium* impact.

push-instead-of-pull-deployment@github-actions-build-pipeline

Unchecked

Unchecked Deployment: 3 / 3 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 **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

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

unchecked-deployment@github-actions-build-pipeline

Unchecked

Low Risk Severity

Unchecked Deployment risk at **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@codeql-code-inspection

Unchecked

Unchecked Deployment risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@github-sourcecode-repository

Unchecked

Unencrypted Technical Assets: 4 / 4 Risks

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 **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

Unencrypted Technical Asset named **Rust Function App**: Exploitation likelihood is *Unlikely* with *High* impact.

unencrypted-asset@rust-function-app

Unchecked

Unencrypted Technical Asset named **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@codeql-code-inspection

Unchecked

Unencrypted Technical Asset named **GitHub Actions Build Pipeline**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@github-actions-build-pipeline

Unchecked

Unencrypted Technical Asset named **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@github-sourcecode-repository

Unchecked

Identified Risks by Technical Asset

In total **37 potential risks** have been identified during the threat modeling process of which **1 are rated as critical**, **0 as high**, **11 as elevated**, **22 as medium**, and **3 as low**.

These risks are distributed across **6 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.

Splunk: 3 / 3 Risks

Description

All data from the function app ends up in Splunk for analysis.

Identified Risks of Asset

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

Critical Risk Severity

Example Individual Risk at : Exploitation likelihood is *Likely* with *High* impact.

information-disclosure@splunk@splunk-network@splunk-data

Unchecked

Elevated Risk Severity

Unguarded Access from Internet of **Splunk** by **Rust Function App** via **Send data to Splunk**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@splunk@rust-function-app@rust-function-app>send-data-to-splunk

Unchecked

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

missing-hardening@splunk

Unchecked

Asset Information

ID:	splunk
Type:	datastore
Usage:	business
RAA:	90 %
Size:	service
Technology:	big-data-platform
Tags:	splunk
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false

Client by Human:	false
Data Processed:	Splunk data
Data Stored:	Splunk data
Formats Accepted:	JSON

Asset Rating

Owner:	dfe	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	important	(rated 3 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	KeyVault should be properly secured due to the increasing attack surface that would result from leaking the secrets within.	

Incoming Communication Links: 1

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

Send data to Splunk (incoming)

Data collected and sent to SplunkCloud for analysis, via a HEC token.

Source:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	splunk
VPN:	false
IP-Filtered:	true
Data Received:	Splunk data
Data Sent:	none

GitHub Actions Build Pipeline: 7 / 7 Risks

Description

GitHub Actions Build Pipeline

Identified Risks of Asset

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

Elevated Risk Severity

Unguarded Access from Internet of **GitHub Actions Build Pipeline** by **Rust Function App** via **Build Pipeline Traffic**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@github-actions-build-pipeline@rust-function-app@rust-function-app>build-pipeline-traffic

Unchecked

Medium Risk Severity

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

code-backdooring@github-actions-build-pipeline

Unchecked

Server-Side Request Forgery (SSRF) risk at **GitHub Actions Build Pipeline** server-side web-requesting the target **CodeQL Code Inspection** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@github-actions-build-pipeline@codeql-code-inspection@github-actions-build-pipeline>code-inspection-platform-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **GitHub Actions Build Pipeline** server-side web-requesting the target **GitHub Sourcecode Repository** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@github-actions-build-pipeline@github-sourcecode-repository@github-actions-build-pipeline>sourcecode-repository-traffic

Unchecked

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

server-side-request-forgery@github-actions-build-pipeline@rust-function-app@github-actions-build-pipeline>function-app-push

Unchecked

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

unchecked-deployment@github-actions-build-pipeline

Unchecked

Unencrypted Technical Asset named **GitHub Actions Build Pipeline**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@github-actions-build-pipeline

Unchecked

Asset Information

ID:	github-actions-build-pipeline
Type:	process
Usage:	devops
RAA:	53 %
Size:	service
Technology:	build-pipeline
Tags:	github-actions
Internet:	true
Machine:	virtual
Encryption:	none
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Sourcecode
Data Stored:	Sourcecode
Formats Accepted:	File

Asset Rating

Owner:	dfe	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	Build pipeline components are at least rated as 'critical' in terms of integrity, because any malicious modification of it might lead to a backdoored production system.	

Outgoing Communication Links: 3
Target technical asset names are clickable and link to the corresponding chapter.

Sourcecode Repository Traffic (outgoing)
Sourcecode Repository Traffic

Target:	GitHub Sourcecode Repository
Protocol:	https
Encrypted:	true
Authentication:	credentials
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	git, github
VPN:	false
IP-Filtered:	false
Data Sent:	none
Data Received:	Sourcecode

Function App Push (outgoing)
Deployment of the Rust function app and infrastructure

Target:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	azure-function-app
VPN:	false
IP-Filtered:	false
Data Sent:	Sourcecode
Data Received:	Sourcecode

Code Inspection Platform Traffic (outgoing)
Code Inspection Platform Traffic

Target:	CodeQL Code Inspection
Protocol:	https

Encrypted:	true
Authentication:	credentials
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	codeql
VPN:	false
IP-Filtered:	false
Data Sent:	Sourcecode
Data Received:	none

Incoming Communication Links: 1

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

Build Pipeline Traffic (incoming)

Build Pipeline Traffic

Source:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	github-actions
VPN:	false
IP-Filtered:	false
Data Received:	Sourcecode
Data Sent:	Sourcecode

Key Vault: 3 / 3 Risks

Description

The key vault containing keys for sensitive systems, and the Splunk data ingestion HEC token.

Identified Risks of Asset

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

Elevated Risk Severity

Unguarded Access from Internet of **Key Vault** by **Rust Function App** via **Retrieve keys**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@key-vault@rust-function-app@rust-function-app>retrieve-keys

Unchecked

Missing Hardening risk at **Key Vault**: Exploitation likelihood is *Likely* with *Medium* impact.

missing-hardening@key-vault

Unchecked

Medium Risk Severity

Missing Vault Isolation to further encapsulate and protect vault-related asset **Key Vault** against unrelated lower protected assets **in the same network segment**, which might be easier to compromise by attackers: Exploitation likelihood is *Unlikely* with *High* impact.

missing-vault-isolation@key-vault

Unchecked

Asset Information

ID:	key-vault
Type:	datastore
Usage:	business
RAA:	92 %
Size:	service
Technology:	vault
Tags:	azure-key-vault
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	false
Redundant:	false

Custom-Developed:	false
Client by Human:	false
Data Processed:	Secrets and API Keys
Data Stored:	Secrets and API Keys
Formats Accepted:	JSON

Asset Rating

Owner:	dfe	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	important	(rated 3 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	KeyVault should be properly secured due to the increasing attack surface that would result from leaking the secrets within.	

Incoming Communication Links: 1

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

Retrieve keys (incoming)

The function must retrieve keys from Key Vault in order to call APIs and send data to Splunk.

Source:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	business
Tags:	azure, splunk
VPN:	false
IP-Filtered:	false
Data Received:	Secrets and API Keys
Data Sent:	Secrets and API Keys

Rust Function App: 10 / 10 Risks

Description

The Rust Function App collects data from Azure, O365, AWS, GitHub.

Identified Risks of Asset

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

Elevated Risk Severity

Unguarded Access from Internet of **Rust Function App** by **GitHub Actions Build Pipeline** via **Function App Push**: Exploitation likelihood is *Very Likely* with *Medium* impact.

unguarded-access-from-internet@rust-function-app@github-actions-build-pipeline@github-actions-build-pipeline>function-app-push

Unchecked

Missing Hardening risk at **Rust Function App**: Exploitation likelihood is *Likely* with *Medium* impact.

missing-hardening@rust-function-app

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Key Vault** via **Retrieve keys**: Exploitation likelihood is *Likely* with *Medium* impact.

server-side-request-forgery@rust-function-app@key-vault@rust-function-app>retrieve-keys

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Various service REST endpoints** via **API Calls to Services**: Exploitation likelihood is *Likely* with *Medium* impact.

server-side-request-forgery@rust-function-app@various-service-rest-endpoints@rust-function-app>api-calls-to-services

Unchecked

Medium Risk Severity

Unencrypted Technical Asset named **Rust Function App**: Exploitation likelihood is *Unlikely* with *High* impact.

unencrypted-asset@rust-function-app

Unchecked

Push instead of Pull Deployment at **Rust Function App** via build pipeline asset **GitHub Actions Build Pipeline**: Exploitation likelihood is *Unlikely* with *Medium* impact.

push-instead-of-pull-deployment@github-actions-build-pipeline

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **CodeQL Code Inspection** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@codeql-code-inspection@rust-function-app>code-inspection-platform-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **GitHub Actions Build Pipeline** via **Build Pipeline Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@github-actions-build-pipeline@rust-function-app>build-pipeline-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **GitHub Sourcecode Repository** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@github-sourcecode-repository@rust-function-app>sourcecode-repository-traffic

Unchecked

Server-Side Request Forgery (SSRF) risk at **Rust Function App** server-side web-requesting the target **Splunk** via **Send data to Splunk**: Exploitation likelihood is *Unlikely* with *Medium* impact.

server-side-request-forgery@rust-function-app@splunk@rust-function-app>send-data-to-splunk

Unchecked

Asset Information

ID:	rust-function-app
Type:	external-entity
Usage:	business
RAA:	100 %
Size:	application
Technology:	function
Tags:	azure-function-app
Internet:	true
Machine:	serverless
Encryption:	none
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Secrets and API Keys, Sourcecode, Splunk data
Data Stored:	Secrets and API Keys, Splunk data
Formats Accepted:	JSON

Asset Rating

Owner:	dfe	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	The function app handles sensitive keys and processess sensitive system infromation from across the estate.	

Outgoing Communication Links: 6

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

Sourcecode Repository Traffic (outgoing)

Sourcecode Repository Traffic

Target:	GitHub Sourcecode Repository
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	git, github
VPN:	false
IP-Filtered:	false
Data Sent:	Sourcecode
Data Received:	Sourcecode

Send data to Splunk (outgoing)

Data collected and sent to SplunkCloud for analysis, via a HEC token.

Target:	Splunk
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	splunk

VPN:	false
IP-Filtered:	true
Data Sent:	Splunk data
Data Received:	none

Retrieve keys (outgoing)

The function must retrieve keys from Key Vault in order to call APIs and send data to Splunk.

Target:	Key Vault
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	business
Tags:	azure, splunk
VPN:	false
IP-Filtered:	false
Data Sent:	Secrets and API Keys
Data Received:	Secrets and API Keys

Code Inspection Platform Traffic (outgoing)

Code Inspection Platform Traffic

Target:	CodeQL Code Inspection
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	codeql
VPN:	false
IP-Filtered:	false
Data Sent:	none
Data Received:	Sourcecode

Build Pipeline Traffic (outgoing)

Build Pipeline Traffic

Target:	GitHub Actions Build Pipeline
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	github-actions
VPN:	false
IP-Filtered:	false
Data Sent:	Sourcecode
Data Received:	Sourcecode

API Calls to Services (outgoing)

The function makes multiple API calls to Azure, AWS, O365 and GitHub.

Target:	Various service REST endpoints
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	business
Tags:	azure, github, o365
VPN:	false
IP-Filtered:	false
Data Sent:	Secrets and API Keys, Sourcecode
Data Received:	Splunk data

Incoming Communication Links: 1

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

Function App Push (incoming)

Deployment of the Rust function app and infrastructure

Source:	GitHub Actions Build Pipeline
Protocol:	https
Encrypted:	true
Authentication:	token

Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	azure-function-app
VPN:	false
IP-Filtered:	false
Data Received:	Sourcecode
Data Sent:	Sourcecode

CodeQL Code Inspection: 5 / 5 Risks

Description

CodeQL Code Inspection

Identified Risks of Asset

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

Medium Risk Severity

Unencrypted Technical Asset named **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@codeql-code-inspection

Unchecked

Unguarded Access from Internet of **CodeQL Code Inspection** by **GitHub Actions Build Pipeline** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@codeql-code-inspection@github-actions-build-pipeline@github-actions-build-pipeline>code-inspection-platform-traffic

Unchecked

Unguarded Access from Internet of **CodeQL Code Inspection** by **Rust Function App** via **Code Inspection Platform Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@codeql-code-inspection@rust-function-app@rust-function-app>code-inspection-platform-traffic

Unchecked

Low Risk Severity

Code Backdooring risk at **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Low* impact.

code-backdooring@codeql-code-inspection

Unchecked

Unchecked Deployment risk at **CodeQL Code Inspection**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@codeql-code-inspection

Unchecked

Asset Information

ID:	codeql-code-inspection
Type:	process
Usage:	devops

RAA:	15 %
Size:	service
Technology:	code-inspection-platform
Tags:	codeql
Internet:	true
Machine:	virtual
Encryption:	none
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Sourcecode
Data Stored:	Sourcecode
Formats Accepted:	File

Asset Rating

Owner:	dfe	
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:	Sourcecode inspection platforms are rated at least 'important' in terms of integrity, because any malicious modification of it might lead to vulnerabilities found by the scanner engine not being shown.	

Incoming Communication Links: 2

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

Code Inspection Platform Traffic (incoming)

Code Inspection Platform Traffic

Source:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	devops

Tags:	codeql
VPN:	false
IP-Filtered:	false
Data Received:	none
Data Sent:	Sourcecode

Code Inspection Platform Traffic (incoming)

Code Inspection Platform Traffic

Source:	GitHub Actions Build Pipeline
Protocol:	https
Encrypted:	true
Authentication:	credentials
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	codeql
VPN:	false
IP-Filtered:	false
Data Received:	Sourcecode
Data Sent:	none

GitHub Sourcecode Repository: 6 / 6 Risks

Description

github Sourcecode Repository

Identified Risks of Asset

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

Medium Risk Severity

Code Backdooring risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *High* impact.

code-backdooring@github-sourcecode-repository

Unchecked

Accidental Secret Leak risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Medium* impact.

accidental-secret-leak@github-sourcecode-repository

Unchecked

Unencrypted Technical Asset named **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Medium* impact.

unencrypted-asset@github-sourcecode-repository

Unchecked

Unguarded Access from Internet of **GitHub Sourcecode Repository** by **GitHub Actions Build Pipeline** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@github-sourcecode-repository@github-actions-build-pipeline@github-actions-build-pipeline>sourcecode-repository-traffic

Unchecked

Unguarded Access from Internet of **GitHub Sourcecode Repository** by **Rust Function App** via **Sourcecode Repository Traffic**: Exploitation likelihood is *Very Likely* with *Low* impact.

unguarded-access-from-internet@github-sourcecode-repository@rust-function-app@rust-function-app>sourcecode-repository-traffic

Unchecked

Low Risk Severity

Unchecked Deployment risk at **GitHub Sourcecode Repository**: Exploitation likelihood is *Unlikely* with *Low* impact.

unchecked-deployment@github-sourcecode-repository

Unchecked

Asset Information

ID:	github-sourcecode-repository
Type:	process
Usage:	devops
RAA:	33 %
Size:	service
Technology:	sourcecode-repository
Tags:	github
Internet:	true
Machine:	virtual
Encryption:	none
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Sourcecode
Data Stored:	Sourcecode
Formats Accepted:	File

Asset Rating

Owner:	dfe	
Confidentiality:	confidential	(rated 4 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	Sourcecode processing components are at least rated as 'critical' in terms of integrity, because any malicious modification of it might lead to a backdoored production system.	

Incoming Communication Links: 2

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

Sourcecode Repository Traffic (incoming)

Sourcecode Repository Traffic

Source:	Rust Function App
Protocol:	https

Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	false
Usage:	devops
Tags:	git, github
VPN:	false
IP-Filtered:	false
Data Received:	Sourcecode
Data Sent:	Sourcecode

Sourcecode Repository Traffic (incoming)
Sourcecode Repository Traffic

Source:	GitHub Actions Build Pipeline
Protocol:	https
Encrypted:	true
Authentication:	credentials
Authorization:	technical-user
Read-Only:	true
Usage:	devops
Tags:	git, github
VPN:	false
IP-Filtered:	false
Data Received:	none
Data Sent:	Sourcecode

Various service REST endpoints: out-of-scope

Description

Various service REST endpoints we collect data from.

Identified Risks of Asset

Asset was defined as out-of-scope.

Asset Information

ID:	various-service-rest-endpoints
Type:	datastore
Usage:	business
RAA:	out-of-scope
Size:	service
Technology:	web-service-rest
Tags:	azure
Internet:	true
Machine:	virtual
Encryption:	data-with-symmetric-shared-key
Multi-Tenant:	false
Redundant:	false
Custom-Developed:	false
Client by Human:	false
Data Processed:	Secrets and API Keys, Splunk data
Data Stored:	Splunk data
Formats Accepted:	JSON

Asset Rating

Owner:	azure and dfe share responsibility	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	important	(rated 3 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	We are not in control of these assets, we simply use them to retrieve data.	

Asset Out-of-Scope Justification

Not part of the overall system, or managed by the team.

Incoming Communication Links: 1

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

API Calls to Services (incoming)

The function makes multiple API calls to Azure, AWS, O365 and GitHub.

Source:	Rust Function App
Protocol:	https
Encrypted:	true
Authentication:	token
Authorization:	technical-user
Read-Only:	true
Usage:	business
Tags:	azure, github, o365
VPN:	false
IP-Filtered:	false
Data Received:	Secrets and API Keys, Sourcecode
Data Sent:	Splunk data

Identified Data Breach Probabilities by Data Asset

In total **37 potential risks** have been identified during the threat modeling process of which **1 are rated as critical**, **0 as high**, **11 as elevated**, **22 as medium**, and **3 as low**.

These risks are distributed across **3 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.

Secrets and API Keys: 16 / 16 Risks

There are sensitive secrets and API keys being held in key vault and used by the Rust Function.

ID:	secrets-and-api-keys	
Usage:	business	
Quantity:	very-few	
Tags:	azure-key-vault	
Origin:	DfE	
Owner:	DfE	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	The Rust Function App secrets are at least rated as 'critical' in terms of integrity, because any access to the keys or modification of the code would reveal sensitive system information across the whole estate.	
Processed by:	Key Vault, Rust Function App, Various service REST endpoints	
Stored by:	Key Vault, Rust Function App	
Sent via:	Retrieve keys, API Calls to Services	
Received via:	Retrieve keys	
Data Breach:	**probable**	
Data Breach Risks:	This data asset has data breach potential because of 16 remaining risks:	

Probable: code-backdooring@github-actions-build-pipeline

Probable: missing-cloud-hardening@azure-network

Possible: server-side-request-forgery@rust-function-app@codeql-code-inspection@rust-function-app>code-inspection-platform-traffic

Possible: server-side-request-forgery@rust-function-app@github-actions-build-pipeline@rust-function-app>build-pipeline-traffic

Possible: server-side-request-forgery@rust-function-app@github-sourcecode-repository@rust-function-app>sourcecode-repository-traffic

Possible: server-side-request-forgery@rust-function-app@key-vault@rust-function-app>retrieve-keys

Possible: server-side-request-forgery@rust-function-app@splunk@rust-function-app>send-data-to-splunk

Possible: server-side-request-forgery@rust-function-app@various-service-rest-endpoints@rust-function-app>api-calls-to-services

Possible: unchecked-deployment@github-actions-build-pipeline

Possible: unguarded-access-from-internet@key-vault@rust-function-app@rust-function-app>retrieve-keys

Possible: unguarded-access-from-internet@rust-function-app@github-actions-build-pipeline@github-actions-build-pipeline>function-app-push

Improbable: missing-hardening@key-vault

Improbable: missing-hardening@rust-function-app

Improbable: missing-vault-isolation@key-vault

Improbable: push-instead-of-pull-deployment@github-actions-build-pipeline

Improbable: unencrypted-asset@rust-function-app

Sourcecode: 30 / 30 Risks

Sourcecode to build the application components from.

ID:	sourcecode	
Usage:	devops	
Quantity:	few	
Tags:	git, github	
Origin:	dfe	
Owner:	dfe	
Confidentiality:	public	(rated 1 in scale of 5)
Integrity:	important	(rated 3 in scale of 5)
Availability:	important	(rated 3 in scale of 5)
CIA-Justification:	The source code is continuously changing, deployed by CI/CD and is public due to the nature of Government's coding in the open policy.	
Processed by:	CodeQL Code Inspection, GitHub Actions Build Pipeline, GitHub Sourcecode Repository, Rust Function App	
Stored by:	CodeQL Code Inspection, GitHub Actions Build Pipeline, GitHub Sourcecode Repository	
Sent via:	Sourcecode Repository Traffic, Function App Push, Code Inspection Platform Traffic, Build Pipeline Traffic, API Calls to Services	
Received via:	Sourcecode Repository Traffic, Sourcecode Repository Traffic, Function App Push, Code Inspection Platform Traffic, Build Pipeline Traffic	
Data Breach:	**probable**	
Data Breach Risks:	This data asset has data breach potential because of 30 remaining risks:	

Probable: accidental-secret-leak@github-sourcecode-repository

Probable: code-backdooring@codeql-code-inspection

Probable: code-backdooring@github-actions-build-pipeline

Probable: code-backdooring@github-sourcecode-repository

Probable: missing-cloud-hardening@azure-network

Probable: missing-cloud-hardening@github-network

Possible: server-side-request-forgery@github-actions-build-pipeline@codeql-code-inspection@github-actions-build-pipeline>code-inspection-platform-traffic

Possible: server-side-request-forgery@github-actions-build-pipeline@github-sourcecode-repository@github-actions-build-pipeline>sourcecode-repository-traffic

Possible: server-side-request-forgery@github-actions-build-pipeline@rust-function-app@github-actions-build-pipeline>function-app-push

Possible: server-side-request-forgery@rust-function-app@codeql-code-inspection@rust-function-app>code-inspection-platform-traffic

Possible: server-side-request-forgery@rust-function-app@github-actions-build-pipeline@rust-function-app>build-pipeline-traffic

Possible: server-side-request-forgery@rust-function-app@github-sourcecode-repository@rust-function-app>sourcecode-repository-traffic

Possible: server-side-request-forgery@rust-function-app@key-vault@rust-function-app>retrieve-keys

Possible: server-side-request-forgery@rust-function-app@splunk@rust-function-app>send-data-to-splunk

Possible: server-side-request-forgery@rust-function-app@various-service-rest-endpoints@rust-function-app>api-calls-to-services

Possible: unchecked-deployment@codeql-code-inspection

Possible: unchecked-deployment@github-actions-build-pipeline

Possible: unchecked-deployment@github-sourcecode-repository

Possible: unguarded-access-from-internet@codeql-code-inspection@github-actions-build-pipeline@github-actions-build-pipeline>code-inspection-platform-traffic

Possible: unguarded-access-from-internet@codeql-code-inspection@rust-function-app@rust-function-app>code-inspection-platform-traffic

Possible: unguarded-access-from-internet@github-actions-build-pipeline@rust-function-app@rust-function-app>build-pipeline-traffic

Possible: unguarded-access-from-internet@github-sourcecode-repository@github-actions-build-pipeline@github-actions-build-pipeline>sourcecode-repository-traffic

Possible: unguarded-access-from-internet@github-sourcecode-repository@rust-function-app@rust-function-app>sourcecode-repository-traffic

Possible: unguarded-access-from-internet@rust-function-app@github-actions-build-pipeline@github-actions-build-pipeline>function-app-push

Improbable: missing-hardening@rust-function-app

Improbable: push-instead-of-pull-deployment@github-actions-build-pipeline

Improbable: unencrypted-asset@codeql-code-inspection

Improbable: unencrypted-asset@github-actions-build-pipeline

Improbable: unencrypted-asset@github-sourcecode-repository

Improbable: unencrypted-asset@rust-function-app

Splunk data: 17 / 17 Risks

Splunk data contains all the vulnerability and compliance data for each system we collect from (Azure, AWS, GitHub, O365).

ID:	splunk-data	
Usage:	business	
Quantity:	many	
Tags:	splunk	
Origin:	DfE	
Owner:	DfE	
Confidentiality:	strictly-confidential	(rated 5 in scale of 5)
Integrity:	critical	(rated 4 in scale of 5)
Availability:	operational	(rated 2 in scale of 5)
CIA-Justification:	The splunk data is confidential due to the amount of vulnerability and compliance data for systems across the org.	
Processed by:	Rust Function App, Splunk, Various service REST endpoints	
Stored by:	Rust Function App, Splunk, Various service REST endpoints	
Sent via:	Send data to Splunk	
Received via:	API Calls to Services	
Data Breach:	**probable**	
Data Breach Risks:	This data asset has data breach potential because of 17 remaining risks:	

Probable: code-backdooring@github-actions-build-pipeline

Probable: information-disclosure@splunk@splunk-network@splunk-data

Probable: missing-cloud-hardening@azure-network

Probable: missing-cloud-hardening@splunk-network

Possible: server-side-request-forgery@rust-function-app@codeql-code-inspection@rust-function-app>code-inspection-platform-traffic

Possible: server-side-request-forgery@rust-function-app@github-actions-build-pipeline@rust-function-app>build-pipeline-traffic

Possible: server-side-request-forgery@rust-function-app@github-sourcecode-repository@rust-function-app>sourcecode-repository-traffic

Possible: server-side-request-forgery@rust-function-app@key-vault@rust-function-app>retrieve-keys

Possible: server-side-request-forgery@rust-function-app@splunk@rust-function-app>send-data-to-splunk

Possible: server-side-request-forgery@rust-function-app@various-service-rest-endpoints@rust-function-app>api-calls-to-services

Possible: unchecked-deployment@github-actions-build-pipeline

Possible: unguarded-access-from-internet@rust-function-app@github-actions-build-pipeline@github-actions-build-pipeline>function-app-push

Possible: unguarded-access-from-internet@splunk@rust-function-app@rust-function-app>send-data-to-splunk

Improbable: missing-hardening@rust-function-app

Improbable: missing-hardening@splunk

Improbable: push-instead-of-pull-deployment@github-actions-build-pipeline

Improbable: unencrypted-asset@rust-function-app

Trust Boundaries

In total **3 trust boundaries** have been modeled during the threat modeling process.

Azure Trust Boundary
The Azure ecosystem

ID:	azure-network
Type:	network-cloud-provider
Tags:	azure, azure-function-app, azure-key-vault
Assets inside:	Key Vault, Rust Function App
Boundaries nested:	none

GitHub Trust Boundary
The GitHub ecosystem

ID:	github-network
Type:	network-cloud-provider
Tags:	github
Assets inside:	CodeQL Code Inspection, GitHub Actions Build Pipeline, GitHub Sourcecode Repository
Boundaries nested:	none

Splunk Trust Boundary
The Splunk ecosystem

ID:	splunk-network
Type:	network-cloud-provider
Tags:	splunk
Assets inside:	Splunk
Boundaries nested:	none

Shared Runtimes

In total **0 shared runtime** has been modeled during the threat modeling process.

Risk Rules Checked by Threagile

Threagile Version: 1.0.0
Threagile Build Timestamp: 20231104141112
Threagile Execution Timestamp: 20240502140519
Model Filename: /app/work/threagile-SSPHP-model.yaml
Model Hash (SHA256): 876f42ad2f4ce5de287011f1cbbb6a2e2b94186d2bacdca99c02dbcd9f33dc5e

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:

Information Dislosure
information-disclosure

Individual Risk Category

STRIDE: Information Disclosure
Description: The biggest risk to SSPHP is the leaking of sensitive system data, this is most likely to happen within Splunk due to the high number of users, including those from a third party supplier.
Detection: User logs in after X amount of inactivity, User exfiltrates data from Splunk, User does not have MFA, User doesn't log in for X amount of time.
Rating: The likelihood of this happening is medium as users are strictly trusted staff, the impact is high due to the sensitivity of data.

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 webservers 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

Sam Pritchard 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. Sam Pritchard 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 Sam Pritchard 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.