German Corona Warn App (CWA)

Backend Infrastructure Architecture Overview

Technical Overview

June 2020

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1 Introduction

1.1 Context

This document complements the “CWA Solution Architecture” document, which is developed and published in parallel to this documentation. It is intended as a technical overview document of Corona Warn App (CWA) and its underlaying infrastructure and network. Please refer to https://github.com/corona-warn-app/cwa-documentation/blob/master/solution_architecture.md

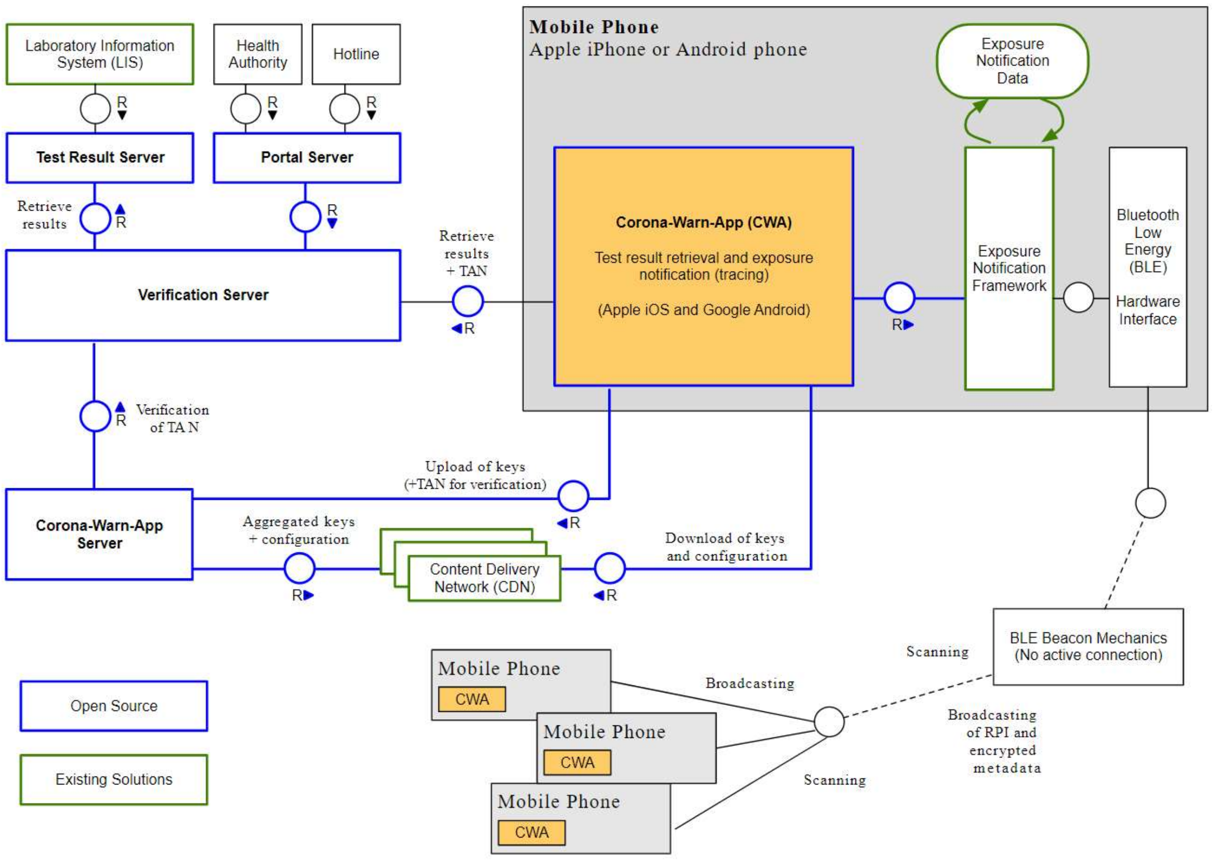
1.2 Scope of Document

This document is part of the overall documentation of the Corona Warn App (CWA, Germany). The documentation is published on github (https://github.com/corona-warn-app/cwa-documentation). The scope of this document is on the infrastructure and network elements, which are necessary to operate the backend services for the CWA.

2 Architecture Outline

2.1 Big Picture

To reduce the spread of COVID-19, it is necessary to inform people about their close proximity to positively tested individuals. So far, health departments and affected individuals have identified possibly infected individuals in personal conversations based on each individuals' memory. This has led to a high number of unknown connections, e.g. when using public transport.



Laboratory Information System (LIS)
Health Authority
Hotline
R
Test Result Server
Portal Server
Retrieve results
R
Verification Server
Retrieve results + TAN
Mobile Phone
Apple iPhone or Android phone
Exposure Notification Data
Corona-Warn-App (CWA)
Test result retrieval and exposure notification (tracing)
(Apple iOS and Google Android)
Exposure Notification Framework
Bluetooth Low Energy (BLE)
Hardware Interface
Verification of TAN
Upload of keys (+TAN for verification)
Corona-Warn-App Server
Aggregated keys + configuration
Content Delivery Network (CDN)
Download of keys and configuration
BLE Beacon Mechanics (No active connection)
Scanning
Broadcasting
Broadcasting of RPI and encrypted metadata
Mobile Phone
CWA
Open Source
Existing Solutions

Figure 1: High level application architecture overview

The Corona-Warn-App, shown centrally in Figure 1, enables individuals to trace their personal exposure risk via their mobile phones. The Corona-Warn-App uses a new framework provided by Apple and Google called Exposure Notification Framework. The framework employs Bluetooth Low Energy (BLE) mechanics. BLE lets the individual mobile phones act as beacons meaning that they constantly broadcast a temporary identifier called Rolling Proximity Identifier (RPI) that is remembered and, at the same time, lets the mobile phone scan for identifiers of other mobile phones. This is shown on the right side of Figure 1. Identifiers are ID numbers sent out by the mobile phones. To ensure privacy and to prevent the tracking of movement patterns of the app user, those broadcasted identifiers are only temporary and change constantly. New identifiers are derived from a Temporary Exposure Key (TEK) that is substituted at midnight (UTC) every day through means of cryptography. Once a TEK is linked to a positive test result, it remains technically the same, but is then called a Diagnosis Key.

The collected identifiers from other users as well as the own keys, which can later be used to derive the identifiers, are stored locally on the phone in the secure storage of the framework provided by Apple and Google. The application cannot access this secure storage directly, but only through the interfaces the Exposure Notification Framework provides. To prevent misuse, some of these interfaces are subjected to rate limiting. If app users are tested positively for SARS-CoV-2, they can update their status in the app by providing a verification of their test and select an option to send their recent keys from up to 14 days back. On the Corona-Warn-App back-end server, all keys of positively tested individuals are aggregated and are then made available to all mobile phones that have the app installed. Additionally, the configuration parameters for the framework are available for download, so that adjustments to the risk score calculation can be made, see the Risk Scores section. Once the keys and the exposure detection configuration have been downloaded, the data is handed over to the Exposure Notification Framework, which analyzes whether one of the identifiers collected by the mobile phone matches those of a positively tested individual. Additionally, the metadata that has been broadcasted together with the identifiers such as the transmit power can now be decrypted and used. Based on the collected data, the Exposure Notification Framework provided by Apple and Google calculates a risk score for each individual exposure as well as for the overall situation. Exposures are defined as an aggregation of all encounters with another individual on a single calendar day (UTC time zone). For privacy reasons, it is not possible to track encounters with other individuals across multiple days.

It is important to note that the persons that have been exposed to a positively tested individual are not informed by a central instance, but the risk of an exposure is calculated locally on their phones. The information about the exposure remains on the user’s mobile phone and is not shared.

The Corona-Warn-App pursues two objectives:

1. It supports individuals in finding out whether they have been exposed to a person that has later been tested positively.
2. It receives the result of a SARS-CoV-2 test on a user's mobile phone through an online system. This helps reduce the time until necessary precautions, e.g. a contact reduction and testing, can be taken.

In order to prevent misuse, individuals need to provide proof that they have been tested positively before they can upload their keys. Through this integrated approach, the verification needed for the upload of the diagnosis keys does not require any further action from the users. They only have to confirm in the app and for the Exposure Notification Framework that they agree to share their diagnosis keys. Manual verification is also possible if the lab that performed the testing does not support the direct electronic transmission of test results to the users' mobile phones or if users have decided against the electronic transmission of their test results.

2.1.1 Project Context

The German government has asked SAP and Deutsche Telekom to develop the Corona-Warn-App for Germany as open source software. Deutsche Telekom is providing the network and mobile technology and will operate and run the backend for the app in a safe, scalable and stable manner. SAP is responsible for the app development, its framework and the underlying platform. Therefore, development teams of SAP and Deutsche Telekom are contributing to this project.

2.1.2 Scoping document

For a detailed view on the context for the Corona Warn App, see the scoping document on github (https://github.com/corona-warn-app/cwa-documentation/blob/master/scoping_document.md).

2.2 System Decomposition

Following the high level architecture overview, shown in Figure 1: High level application architecture overview, the system can be split into the following system elements:

- Mobile phone (not in scope for the backend infrastructure)
- CWA Server
- Verification Server
- Test Result Server
- Verification Portal

In addition to the application level components, the following system and technology specific components must be considered:

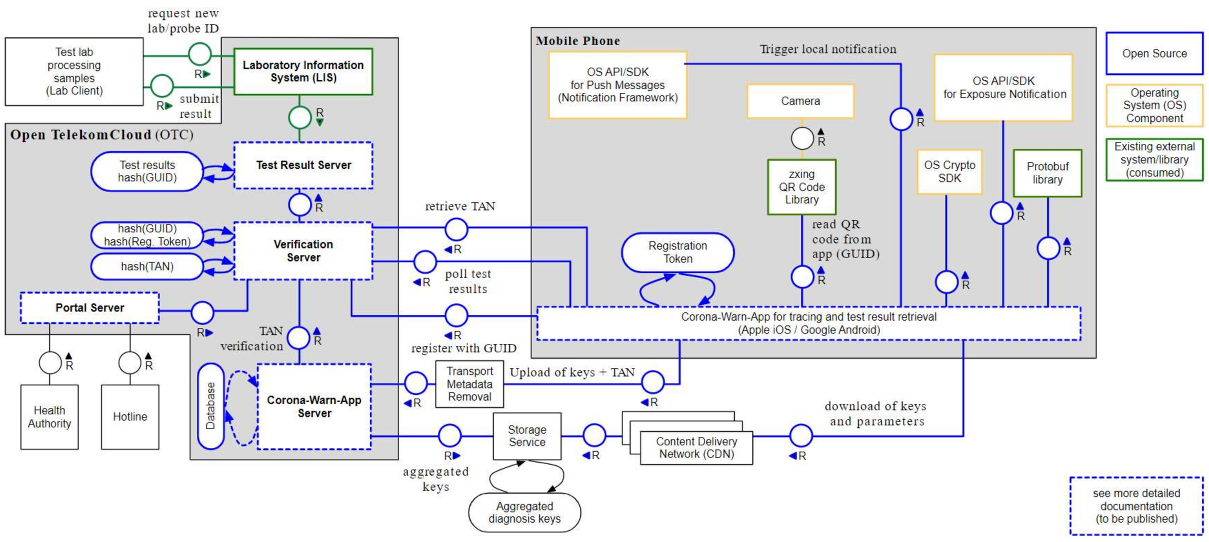
- Content Delivery Network (CDN)
- Network and Security components:
  - DDOS (Distributed-Denial-of-Service)
  - Load Balancer (implementing IP whitelisting and IP masking)
- Infrastructure services (Cloud): Open Telekom Cloud (OTC)
- Platform Services (PaaS): AppAgile (based on OpenShift, a cloud application and orchestration platform)

The next chapters will detail the infrastructure architecture for these components.

3 Architectural Views

3.1 Logical View

For a detailed description of the logical view and the application model of the corona warn app (CWA), a description is available on github (Solution Architecture, https://github.com/corona-warn-app/cwa-documentation/blob/master/solution_architecture.md).



request new lab/probe ID
Test lab processing samples (Lab Client)
submit result
Laboratory Information System (LIS)
Open TelekomCloud (OTC)
Test results hash(GUID)
Test Result Server
hash(GUID) hash(Reg_Token)
hash(TAN)
Verification Server
Portal Server
Health Authority
Hotline
TAN verification
Database
Corona-Warn-App Server
retrieve TAN
poll test results
register with GUID
Transport Metadata Removal
Upload of keys + TAN
Storage Service
aggregated keys
Aggregated diagnosis keys
Content Delivery Network (CDN)
download of keys and parameters
Mobile Phone
Trigger local notification
OS API/SDK for Push Messages (Notification Framework)
Camera
OS API/SDK for Exposure Notification
zxing QR Code Library
read QR code from app (GUID)
Registration Token
OS Crypto SDK
Protobuf library
Corona-Warn-App for tracing and test result retrieval (Apple iOS / Google Android)
R
Open Source
Operating System (OS) Component
Existing external system/library (consumed)
see more detailed documentation (to be published)

Figure 2: CWA logical view

The Corona-Warn-App Server needs to fulfill the following tasks:

- Accept upload requests from clients
  - Verify association with a positive test through the Verification Server and the associated workflow as explained in the “Retrieval of Lab Results and Verification Process” section and shown in the center of the left side of Figure 6.
  - Accept uploaded diagnosis keys and store them (optional) together with the corresponding transmission risk level parameter (integer value of 1-8) into the database. Note that the transport of metadata (e.g. IP) of the incoming connections for the upload of diagnosis keys is removed in a dedicated actor, labeled “Transport Metadata Removal” in Figure 6.
- Provide configuration parameters to the mobile applications
  - Threshold values for attenuation buckets
  - Risk scores for defined values
  - Threshold values for risk categories and alerts
- On a regular schedule (e.g. hourly)
  - Assemble diagnosis keys into chunks for a given time period

- Store chunks as static files (in protocol buffers, compatible with the format specified by Apple and Google)
- Transfer files to a storage service as shown at the bottom of Figure 6 which acts as a source for the Content Delivery Network (CDN)

3.2 Deployment View (Technical Infrastructure)

The following chapters describe the technical infrastructure and the deployment components which were selected by the project partners. It is not necessary to implement the CWA on exact this infrastructure, but the deployment can act as a blueprint for a similar implementation.

3.2.1 Overview

As shown in Figure 3: CWA high level deployment view, the infrastructure for the production environment contains more than only the logical system components, but also a hosting platform (Open Telekom Cloud (OTC) and AppAgile (OpenShift)). The whole environment is hosted in the Open Telekom Cloud, with additional components from the Telekom backbone (Loadbalancer, DDOS). All components are related to one tenant, the CWA OTC Tenant. Tenants within OTC are strictly separated. The hosting and orchestration services within the project is realized with a dedicated OpenShift stack for each project (product name AppAgile@OTC). This allows the usage of all features of OpenShift (based on Kubernetes) for all deployments (container, networking, firewalls, etc.). For each stage (here production stage Prod), two separated dedicated OpenShift stacks are used (like sub-tenants). The first stack contains all backend services, and an additional stack contains all PaaS services which have to be consumed (here OTC RDS, a managed Relational Database Service, in a PostgreSQL flavour). Every OpenShift Stack has its own namespace, which allows a separation of all duties, including addressing, routing and authorization.

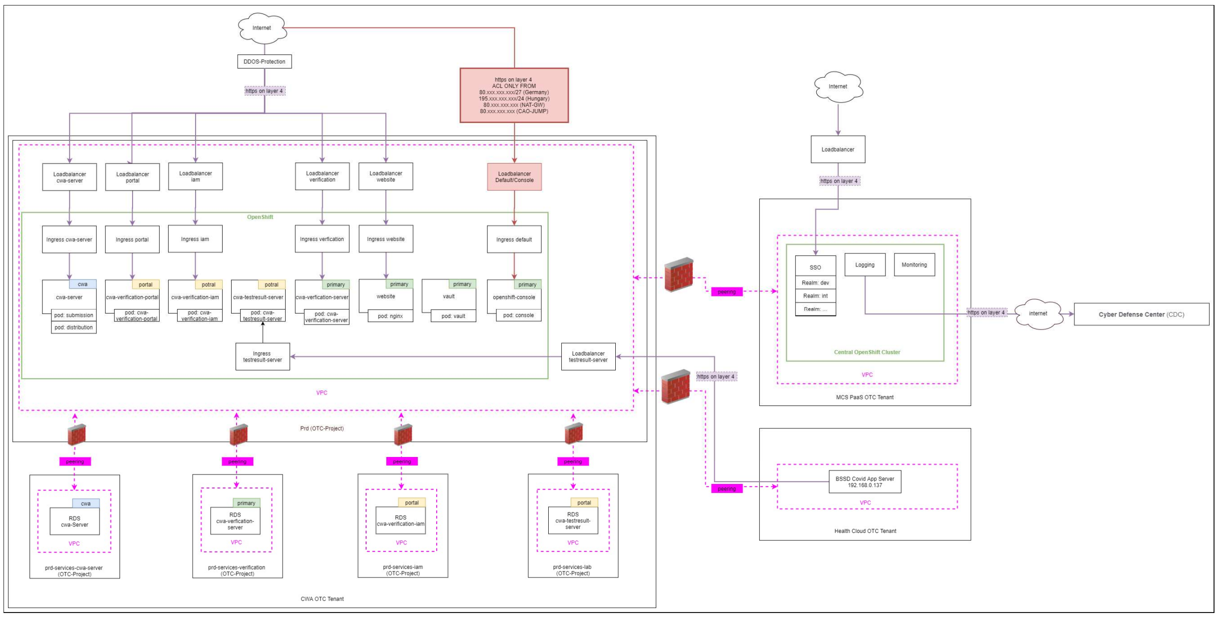


Figure 3: CWA high level deployment view

3.2.2 Staging environments

The following stages are defined:

- DEV – development environment
- INT – integration environment
- WRU – reference environment
- PROD – production environment

To separate all stages (DEV, INT, WRU, PROD), separate OTC sub-projects are used. This allows a strict separation of all traffic, deployments and usage. Also, the user and role definitions can be separated.

The content of all stages is identical, except that DEV and INT contain additional components for the CI/CD pipeline (Jira, Confluence).

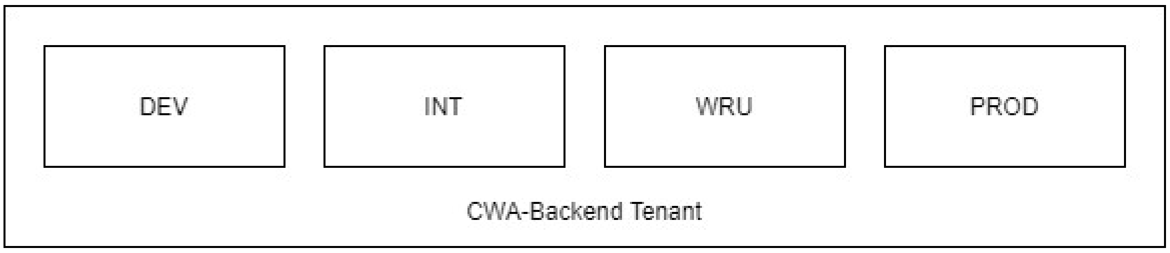


Figure 4: Separated stages withing the CWA OTC Tenant

3.2.3 System building blocks (SBB)

3.2.3.1 Website

This namespace contains a single pod with a nginx container. The nginx stops at https://coronawarn.app and redirects traffic to https://www.coronawarn.app.

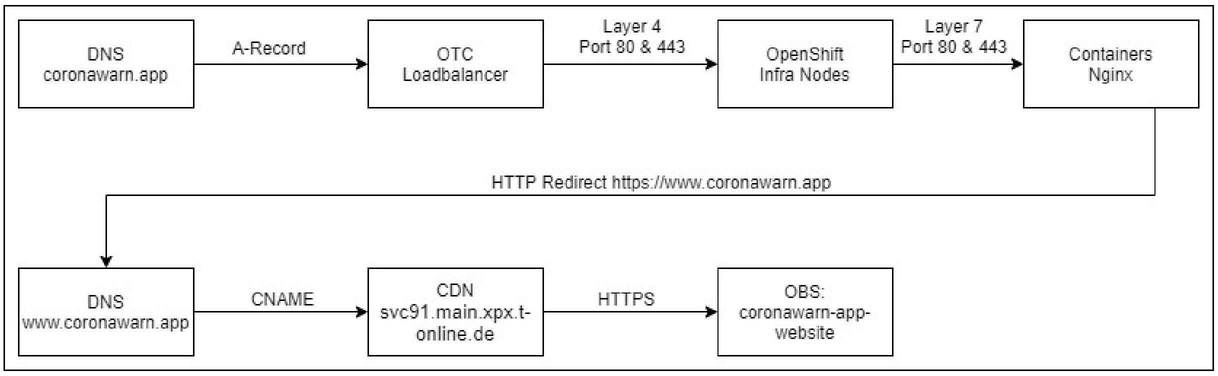


Figure 5: Logical website walkthrough (without DDOS)

3.2.3.2 Open Telekom Cloud (OTC)

Open Telekom Cloud is the public cloud offering from Deutsche Telekom. To learn more about OTC, refer to https://open-telekom-cloud.com/.

3.2.3.3 OpenShift (AppAgile)

AppAgile@OTC is the standardized OpenShift offering from Deutsche Telekom, which is available for all OTC Tenants. It provides a standard OpenShift stack (by RedHat), which is based on Kubernetes.

For a complete description and explanation of the OpenShift service, please refer to https://cloud.telekom.de/en/infrastructure/appagile-paas-big-data.

3.2.3.4 CWA Server

The CWA Backend handles submission and aggregation/distribution of diagnosis keys and configuration files.

For a detailed description of the CWA Backend service, please refer to https://github.com/corona-warn-app/cwa-server/blob/master/docs/architecture-overview.md

3.2.3.5 Verification server

The Verification Server provides proof of a positive SARS-CoV-2 test to other components of the system. In addition, it also provides information about specific SARS-CoV-2 test results. The Verification Server uses several sources to provide such proof and information.

For a complete and actual description of the Verification Server, please have a look on github: https://github.com/corona-warn-app/cwa-verification-server/blob/master/docs/architecture-overview.md

3.2.3.6 Test Result Server

The primary scope of the component is to provide the verification server with information of lab test results of SARS-CoV-2 tested people.

For a detailed description of the Test Result Server, please refer to github: https://github.com/corona-warn-app/cwa-testresult-server/blob/master/docs/architecture-overview.md

3.2.3.7 Verification Portal

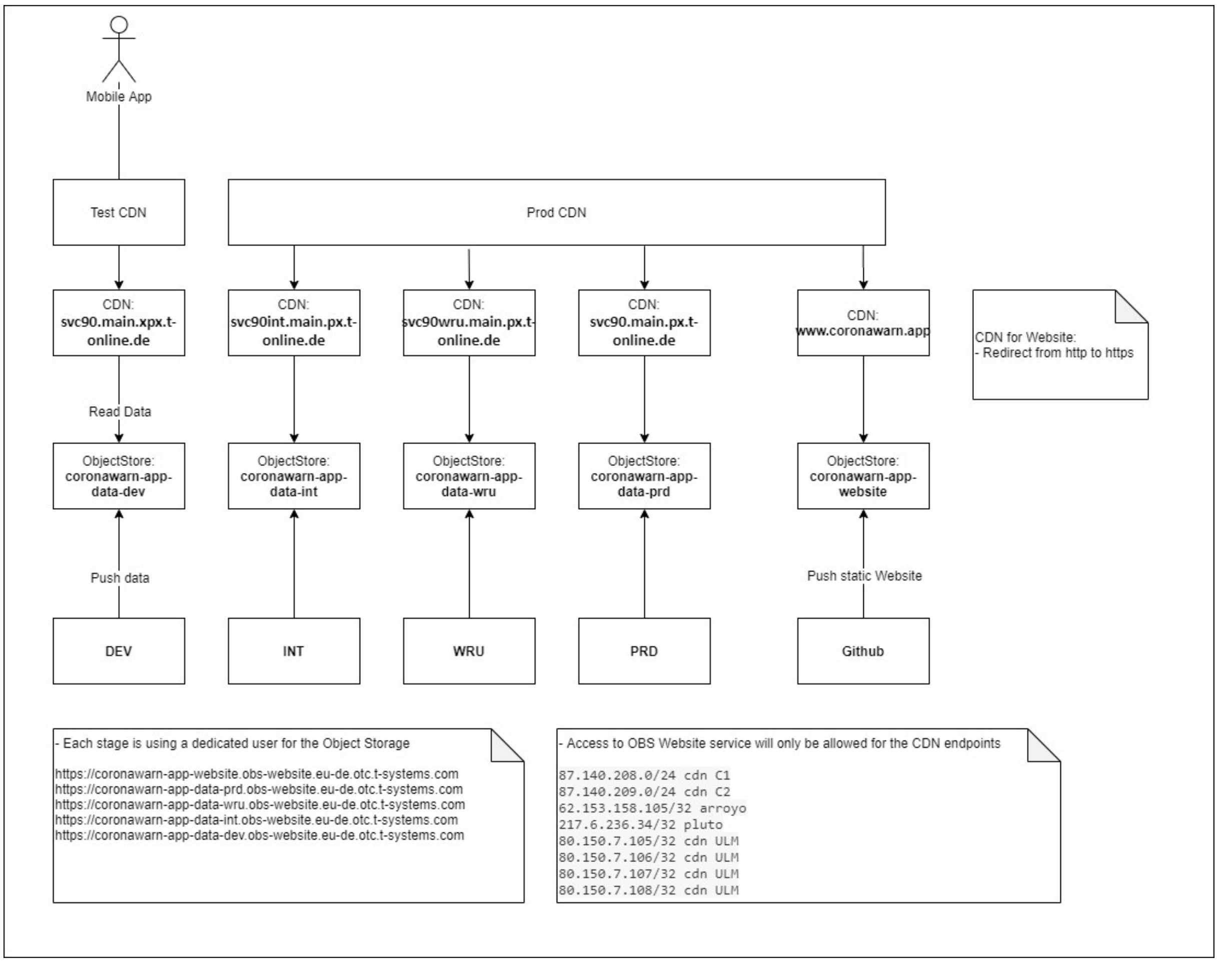
The primary scope of the component is to create a proof certificate (teleTAN) whether users were positive tested for SARS-CoV-2 or not. The proof is generated by hotline employees based on the information they can obtain. The scope of the Verification Portal is just the creation of the proof certificate - the teleTAN. Not in scope is the support of validating the request of people calling the hotline.

For a complete description and explanation of the OpenShift service, please refer to https://github.com/corona-warn-app/cwa-verification-portal/blob/master/docs/architecture-overview.md

3.2.3.8 Content Delivery Network (CDN)

The content delivery network is designed to deliver static content (in this context the list of keys) from the nearest point of the network, where the end-user (consumer) resides. Telekom uses the Magenta CDN to deliver such static content. The source of the content is registered within the CDN. At the first web request for this content, the content is pulled from the source and cached on the nearest endpoint in the CDN. The content is then delivery from this endpoint.

For the 4 different stages, separated CDN entries are necessary. Figure 6: CDN implementation for different stages shows the registration of these different URLs within the Magenta CDN.


Mobile App
Test CDN
Prod CDN
CDN: svc90.main.xpx.t-online.de
CDN: svc90int.main.px.t-online.de
CDN: svc90wru.main.px.t-online.de
CDN: svc90.main.px.t-online.de
CDN: www.coronawarn.app
CDN for Website:
- Redirect from http to https
Read Data
ObjectStore: coronawarn-app-data-dev
ObjectStore: coronawarn-app-data-int
ObjectStore: coronawarn-app-data-wru
ObjectStore: coronawarn-app-data-prd
ObjectStore: coronawarn-app-website
Push data
Push static Website
DEV
INT
WRU
PRD
Github

- Each stage is using a dedicated user for the Object Storage

https://coronawarn-app-website.obs-website.eu-de.otc.t-systems.com
https://coronawarn-app-data-prd.obs-website.eu-de.otc.t-systems.com
https://coronawarn-app-data-wru.obs-website.eu-de.otc.t-systems.com
https://coronawarn-app-data-int.obs-website.eu-de.otc.t-systems.com
https://coronawarn-app-data-dev.obs-website.eu-de.otc.t-systems.com

- Access to OBS Website service will only be allowed for the CDN endpoints

```
87.140.208.0/24 cdn C1
87.140.209.0/24 cdn C2
62.153.158.105/32 arroyo
217.6.236.34/32 pluto
80.150.7.105/32 cdn ULM
80.150.7.106/32 cdn ULM
80.150.7.107/32 cdn ULM
80.150.7.108/32 cdn ULM
```

Figure 6: CDN implementation for different stages

Note: please keep in mind, that the URLs and the implementation is still under development. Changes may occur and may not reflect the actual implementation.

3.2.3.9 Vault

This namespace contains a Hashicorp Vault, to manage the used secrets. Here the TLS certificates for the individual services are provided so that the final termination does not have to take place at the ingress controllers. It also contains the key for signing the "Diagnosis Keys", which are sent to the Apple/Google framework. The key must be in ECDSA-SHA256 format with the prime256v1 curve.

The root token has been removed from the Vault.

The Vault is locked for the first time and must be unlocked by known processes. Subsequently, the components receive access to their certificates, keys, etc. according to the need-to-know principle.

3.2.3.10 RDS

The consumed Relational Database Services (RDS) of OTC are each in a tenant project and are thus completely separated from the other components. In the RDS, PostgreSQL version 11.5 is used and the databases are encrypted (@REST, as it is the data transport to them). Every service that needs a database gets a dedicated RDS (thus runs in a separate namespace).

3.2.3.11 Denial of Service Protection (DDOS)

The DDOS protection in place is a standard feature, available at the Telekom backbone. It protects many services against DDOS attacks. One of the standard services is the Open Telekom Cloud (OTC).

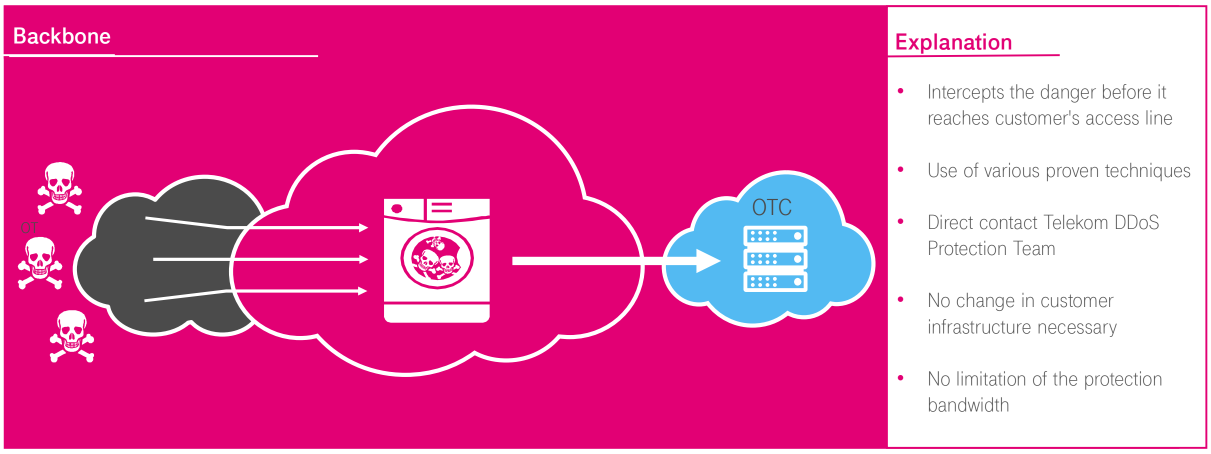

Backbone
OTC
Explanation
Intercepts the danger before it reaches customer's access line
Use of various proven techniques
Direct contact Telekom DDoS Protection Team
No change in customer infrastructure necessary
No limitation of the protection bandwidth

Figure 7: DDOS protection within the Telekom backbone

There are different concepts for DDOS protection in place, which can be used to protect against different attacks:

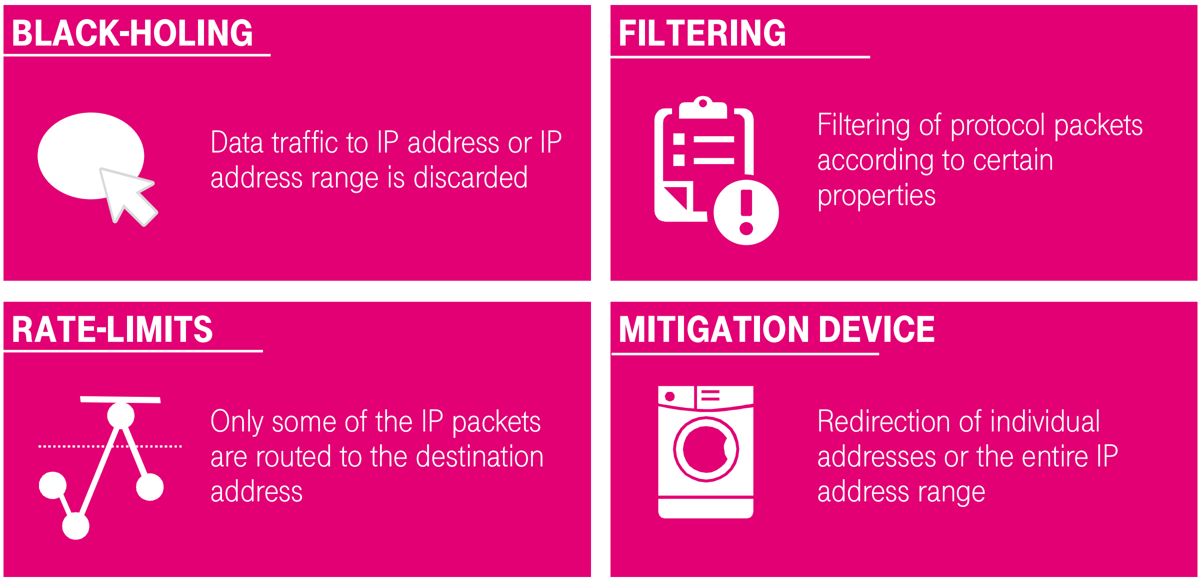

BLACK-HOLING
Data traffic to IP address or IP address range is discarded
FILTERING
Filtering of protocol packets according to certain properties
RATE-LIMITS
Only some of the IP packets are routed to the destination address
MITIGATION DEVICE
Redirection of individual addresses or the entire IP address range

Figure 8: DDOS protection concepts

An overview of DDOS protection functions available by Deutsche Telekom is shown in Table 1: Some available DDOS functions. Some of them are permanently active, others are passive and can be activated when needed.

IP Location Blocking	Fragmentation Prevention	Multi-Attack Prevention	HTTP Basic Botnet Prevention	DNS Malformed Prevention
IP Location Policing	Comprehensive IP/FCAP, DNS & HTTP Filter Lists	URL Blocking	HTTP Regular Expression Filter	DNS Domain Blacklisting
TCP Connection Verification	Payload Filter	HTTP Malformed Prevention	DNS Authentication	DNS Regular Expression Filter
IP Black/White Listing	ATLAS Intelligence Feed (AIF) Prevention	HTTP Authentication	DNS Request Limiting	Multiple SIP Preventions
SYN Authentication	SSL/TLS Protocol	HTTP Flood Prevention	DNS NXDOMAIN Rate Limiting	ICMP Flood Prevention
Traffic Shaping				

Table 1: Some available DDOS functions

3.2.4 Interaction of components

This chapter describes the interactions of the solution building blocks. Please refer to Figure 3: CWA high level deployment view for an overview of relevant components.

3.2.4.1 TLS certificates

The termination of the TLS connection to the CWA server takes place in the namespace. The ingress controller cannot record the traffic.

Connections between servers within the OTC are secured with TLS 1.3 with mutual authentication (with TLS client certificate). The required TLS certificates are generated by a CA provided by OpenShift for each cluster. The web application provided by the Portal Server for hotline staff only enforces TLS 1.2 for interoperability reasons.

The public interfaces (App to CWA server and App to Verification server as well as browser to Verification Portal) each use a server certificate issued by TeleSec whose root certificate is generally known. The App uses certificate pinning to an Intermediate-CA. Since TeleSec belongs to the Deutsche Telekom AG and we trust it, we do not use an own Intermediate CA certificate. Instead an Intermediate CA certificate from TeleSec is used, under which the server certificates are attached. PROD and WRU each have their own different Intermediate CA certificates.

3.2.4.2 IP Masking

The communication of endpoints from the Internet (mobile phone, browser of the hotline staff) is encrypted end-to-end up to the backend systems. The load balancer masks the source IP address so that no IP addresses reach the backend systems which could appear in log files there and enable a re-personalization of the pseudonymized data flows. For this purpose, IP masking is performed on the load balancers.

3.2.4.3 Namespaces

The CWA_Server namespace is assigned a dedicated pool of VMs (cwa pool). The Portal Server and the TestResult Server receive a second, dedicated pool of VMs. The remaining servers/pods share the default pool.

The separation between VMs is more robust than the separation between containers. In addition, the VM Pools are also assigned their own operating teams.

3.2.4.4 VPC Peering

The RDS Tenant projects are assigned to the respective namespaces via VPC peering; VPC stands for Virtual Private Cloud.

The existing laboratory gateway developed by Bucher Software, which collects test results from the laboratories and provides them to the Test Result Server, is also connected via VPC peering. The laboratory gateway is located in the Telekom Healthcare Cloud Tenant, which is being implemented as a special tenant in the OTC. Since private IP addresses are used within the OTC, peering was carried out to ensure that the IP address ranges are disjunctive.

The setup of VPC peering is described here: https://docs.otc.t-systems.com/usermanual/vpc/vpc_peering_0000.html.

3.2.4.5 *Management*

The management interface of the OpenShift platform is accessible from the Internet. However, only IP addresses from the DTAG Corporate Network are allowed. In addition, 2-factor-authorization (2FA) will be enforced for each (administration) account.

3.2.5 Communication matrix and endpoints

The following table shows the public web interfaces:

Interface	DNS-Name	Component	Stage
https	verification-iam-6d99852d.coronawarn.app	verification-iam	WRU
https	Verification-2554e68b.coronawarn.app	Cwa-verification-server	WRU
https	testresult-5c253f79.coronawarn.app	Cwa-testresult-server	WRU
https	verification-portal-0cebc62a.coronawarn.app	Cwa-verification-portal	WRU
https	submission-cff4f7147260.coronawarn.app	Cwa-server	WRU
https	verification-iam.coronawarn.app	verification-iam	PROD
https	verification.coronawarn.app	Cwa-verification-server	PROD
https	testresult.coronawarn.app	Cwa-testresult-server	PROD
https	verification-portal.coronawarn.app	Cwa-verification-portal	PROD
https	submission coronawarn.app	Cwa-server	PROD

Table 2: public web interfaces

The list of API endpoints is shown in the following table:

Component	Endpoint	public	TLS Cert	Connection Path
CWA-Verification-IAM	/auth/**	yes	TeleSec	from Browser, Portal, Verifiaction-Server

CWA-Testresult-Server	/api/v1/lab/results	non	OpenShift	from Bucher Gateway (Health Cloud OTC)
	/api/v1/app/result	no	OpenShift	From Verification-Server
CWA-Verification-Portal	/cwa/**	yes	TeleSec	Browser (redirect to keycloak), (Hotline, Healthservice)
CWA-Verification-Server	/api/v1/testresult	yes	TeleSec	from Mobile
	/api/v1/registrationToken	yes	TeleSec	from Mobile
	/api/tan/verify	no	TeleSec	from CWA-Server
	/api/tan	yes	TeleSec	from Mobile
	/api/tan/teletan	no	OpenShift	from CWA-Verification-Portal

Table 3: API endpoints

3.3 Operational Runtime View (Considerations)

The following list of requirements according the operation of the software has been derived from the given architecture. All of these must be covered by the operation team in order to fulfill all nonfunctional requirements.

- 24/7 operation
- Seamless deployments, minimizing downtimes
- No central data observation
- Segregation of duties: following groups of components cannot be operated from the same persons at any point of time (this includes monitoring with log data access):
  - security components (DDOS),
  - backend services (Backend Server, Verification Server, Portal Server, Test Result Server) and
  - the other infrastructure components (Load Balancer)
- Certificate Management & Governance

4 Cross-Cutting System Properties

4.1 Scalability and Availability

The Corona Warning App backend (CWA backend) is operated in the Open Telekom Cloud (OTC) on the EU-DE region and is therefore subject to strict German data protection guidelines. Availability is guaranteed by three different Availability Zones (AZ). The AZ are spread over two different locations in Magdeburg and Biere. The CWA backend is based on a PaaS from AppAgile (OpenShift). This also ensures the scalability within the three availability zones. The platform is provided in a dedicated OTC tenant (as a managed platform service).

4.2 Security and Data Privacy

There is a data protection requirement that the CWA server and the Verification Portal server must be separated because pseudonymized or even personal data is processed here. For this reason, these are not only operated by different operating teams, but also have dedicated node pools. Access to the databases (RDS) is also very restrictive. Operating members only see the components to which they have been granted access. Furthermore, it is required that an employee is only assigned to one operating team.

Operation takes place on three layers:

1. Application operation
2. PaaS operation (AppAgile)
3. OTC operation

However, this segregation of duties can only be implemented at application level. The platform itself (AppAgile), theoretically has access to all components, as they assign rights and roles. This assignment of rights and roles is to take place under the principle of dual control and is additionally logged. A detailed description of how the operation takes place, which roles are assigned and from where the whole infrastructure and application is operated, is recorded in the operating documentation.

The operation may only take place within the EU, thus following EU-GDPR.

For a list of security requirements assessed during the project, you can have a look on github: https://github.com/corona-warn-app/cwa-documentation/blob/master/overview-security.md

4.3 Integration/Communication with Other IT Systems

Currently, integration and communication with other external IT Systems is not necessary. To ensure interoperability between solutions built by other European countries, an EU Interoperability Concept is developed by T-Systems/SAP.

4.4 Monitoring

Monitoring and Logging is performed from a separate OTC Tenant. This Tenant is used to perform managed services for the CWA OTC Tenant. Connected to the Management Tenant (MCS PaaS) is the Telekom Cyber Defense Center (CDC).

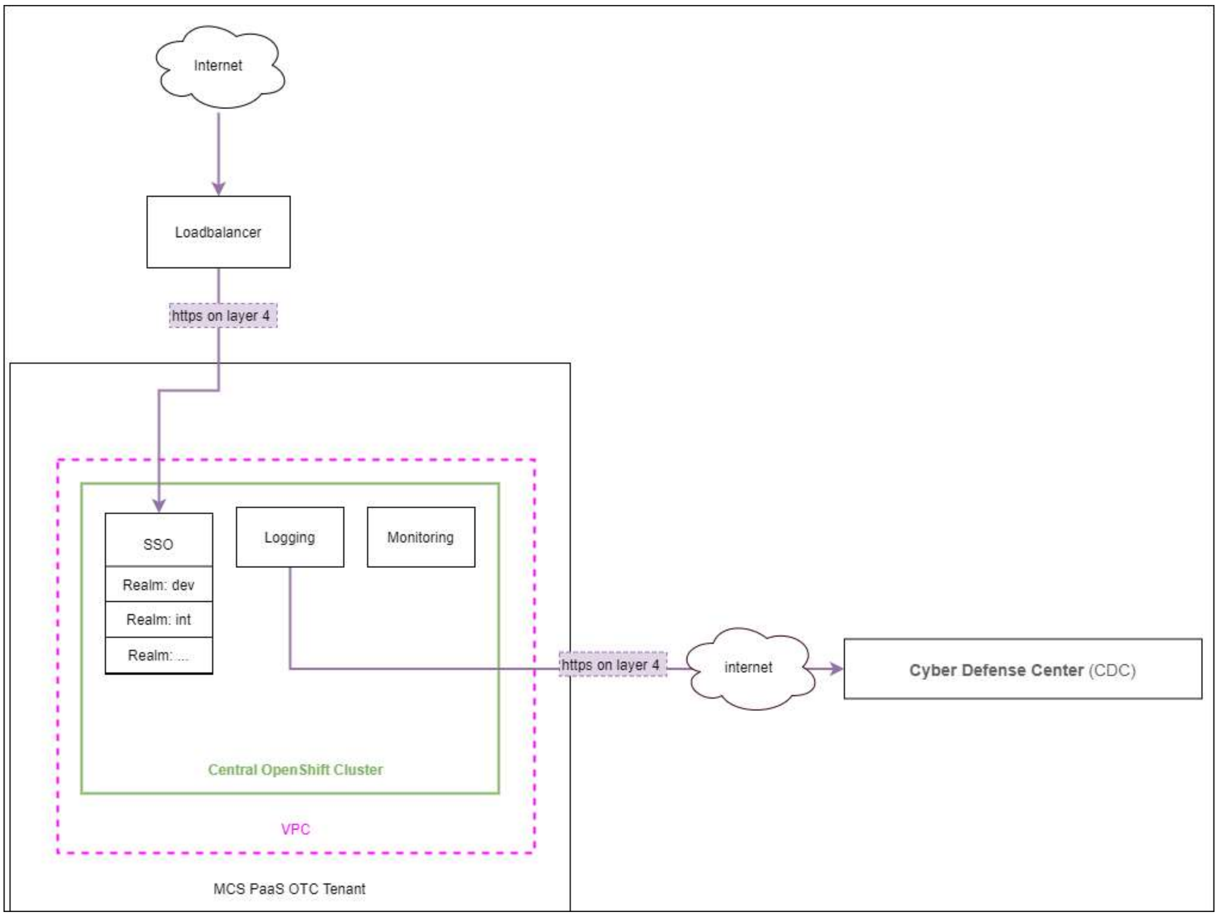


Figure 9: Management OTC Tenant

4.5 Tracing and Logging

The logging of the infrastructure of the CWA backend is explained in the AppAgile Customer System Description under the section "Logging & Monitoring".

OpenShift, AppAgile and OTC logs that occur in the CWA tenant are logged to the central graylog of AppAgile. Additionally, there is a graylog in every stage of the CWA-Tenant. There you will find all logs, which also go to the central graylog and additionally the application logs of the services. These logs are also transmitted to the Cyber Defense Center (CDC) via TLS (logs sent to CDC do not include any personal data like IP addresses of the users of CWA).

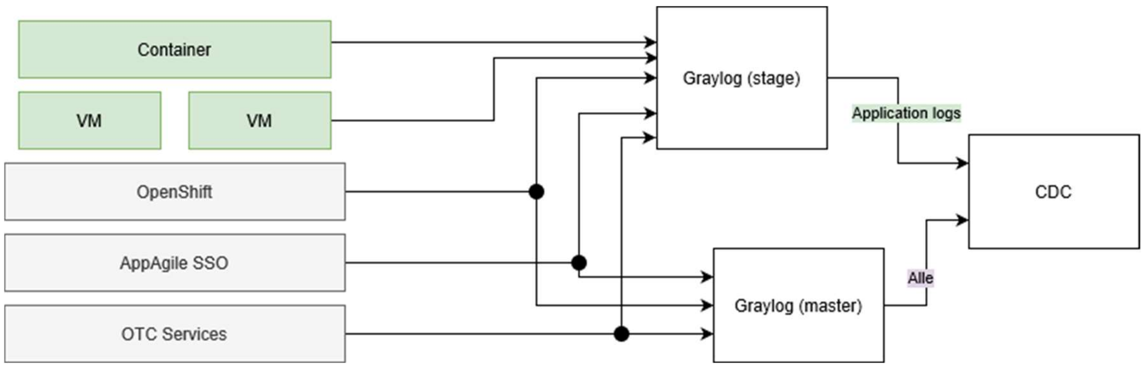


Figure 10: Overview of logging paths

If an incident occurs, the CDC reports it to the application operations team.

The CWA backend logs contain (not exhaustive)

- Relevant events of the submission service
- TAN checks
- CDN uploads
- Signing of the "Diagnosis Keys"
- Status transmission between lab gateway and test result server

4.6 Security monitoring by the Cyber Defense Center (CDC)

4.6.1 Objective of the Security Monitoring

To increase the security of the overall solution during the operating phase, T-Systems monitors key components using the Managed Cyber Defense Service from Telekom Security in Bonn.

The service combines automated and manual analyses of security-relevant logs from the IT systems. In addition, daily updated threat intelligence information from T-Systems is used to improve the quality of the analyses.

The Managed Cyber Defense Services from T-Systems make it possible to identify attempted attacks and, if necessary, successful compromises of systems and identities and to initiate countermeasures.

4.6.2 Implementation of the connection

In order to detect attempted attacks, security-relevant log files are forwarded to the Magenta Security Information and Event Management System (SIEM) for alerting of anomalies or investigation of potential incidents.

For real-time alerting and long-term analysis, rule sets are configured in the system. These alerts are based on threat scenarios that affect the platform and are modeled by experts at the Telekom Cyber Defense Center. Typical alerting scenarios include attempts to break passwords by trial and error or the comparison of log data against indicators that are known to indicate malicious activity.

Log data generated is directly forwarded to the SIEM from the log servers in the AppAgile environment. This is done using a Transport Layer Security (TLS) secured connection and the HTTP method POST. The TLS connection guarantees on the one hand the encryption of the connection, on the other hand we additionally use client and server certificates, so that sender and receiver are authenticated.

On the SIEM platform the data is processed and read in directly. In this first step, our real-time alerting uses configured rule sets. The data is stored for seven days in order to carry out analyses in the past over this period. This can be done by means of automated rules or incident related, for example if automated investigations give reason to carry out further analyses.

4.6.3 Security Monitoring and Incident Response

The SIEM platform is designed for high availability and operated in a 24/7 model.

The detection scenarios (use cases) are defined in the project phase after risk assessment and availability of corresponding security logs and are regularly optimized. In the event of a detection, the incident is processed according to the specified runbook.

If the analyses provide indications of security incidents, an in-depth analysis is carried out by experienced Telekom Security specialists as part of incident response activities.

4.6.4 Transmitted data

In particular, the following log data is transmitted to the Cyber Defense Center (in each case relating to the components used by the Corona Warn App):

- Audit logs of the AppAgile platform
- Virtual Machine System Logs
- Logs of the applications (especially Verification Server, Laboratory Server, Portal Server, CWA Backend), which are written to "stdout" or "stderr"
- Cloud Trace Logs from the OTC clients used by the Corona Warn App

Since one of the main objectives of the application is to maintain user anonymity, no log data containing users' personally identifiable information (e.g., their IP addresses, data from their mobile devices, and request data from users) is transmitted to the SIEM.

REFERENCES

References to documents on github:

- Corona-Warn-App - Scoping Document (https://github.com/corona-warn-app/cwa-documentation/blob/master/scoping_document.md)
- Corona-Warn-App - User Interface Screens (https://github.com/corona-warn-app/cwa-documentation/blob/master/ui_screens.md)
- Corona-Warn-App - Solution Architecture (https://github.com/corona-warn-app/cwa-documentation/blob/master/solution_architecture.md)
- Corona-Warn-App Server Architecture (https://github.com/corona-warn-app/cwa-server/blob/master/docs/architecture-overview.md)
- Corona-Warn-App Verification Server Software Design (https://github.com/corona-warn-app/cwa-verification-server/blob/master/docs/architecture-overview.md)
- Corona-Warn-App Verification Portal Server Software Design (https://github.com/corona-warn-app/cwa-verification-portal/blob/master/docs/architecture-overview.md)
- Corona-Warn-App Test Result Server Software Design (https://github.com/corona-warn-app/cwa-testresult-server/blob/master/docs/architecture-overview.md)
- Corona-Warn-App Security Overview (https://github.com/corona-warn-app/cwa-documentation/blob/master/overview-security.md)