OPENCRVS

EVALUATION REPORT

SEPTEMBER 2023

Guardian Project conducted an independent evaluation of OpenCRVS, Version 1.2.0 (and v1.3.0 for Pen Testing).
This task was commissioned by the Digital Center of Excellence (DCoE) at UNICEF.

PREPARED BY
Guardian Project

New Release Statement

July 2024

This assessment was completed and reviewed with the OpenCRVS team in September and October 2023. Since then, the OpenCRVS team has diligently worked on implementing the recommendations from this assessment along with feedback from system integrators involved in the OpenCRVS Implementation Partner Programme and other architecture, infrastructure and deployment considerations. While we cannot confirm full resolution at this time, the responsiveness of the OpenCRVS team and the release notes from versions 1.4 and 1.5 of the software indicate that the recommendations have been taken seriously and likely implemented. Numerous technical enhancements have been made to the platform, and they are now about to release OpenCRVS v1.5.0 (https://documentation.opencrvs.org/general/releases/v1.5.0-release-notes). Furthermore, a number of issues have been addressed or resolved since our assessment. You can find a table of addressed issues in this reports Annex.

OpenCRVS

Evaluation Report

Performed by Guardian Project
Final Report, September 2023
Solution Name and Version: OpenCRVS, Version 1.2.0 (and v1.3.0 for Pen Testing)

About Guardian Project

With 15 years of experience in the Internet Freedom space, Guardian Project is dedicated to building apps and technologies prioritising the safety and protection of those we work with. Our core values include security, privacy, and transparency, reflected in all our developments.

Relevant Expertise

Our work at Oliver+Coady, inc, via Guardian Project, has always been focused on the human rights and humanitarian context, aiming to bend technology to better serve people and communities whose data and digital communications are at higher risk of being exploited and used against them. Over the last 15 years, we have provided security and privacy-focused software architecture, development, and operational deployment services across the human rights and humanitarian technology space. We have experience managing complex multi-year, multi-million dollar technical projects with many stakeholders, and hundreds of thousands to millions of end-users.

For over a decade, we have also been heavily involved in open-source software communities, in particular those focused on privacy-enhancing technology, security-by-design and the minimization of tracking by third-parties. We have worked within projects that are part of Debian, Tor Project, Mozilla, Android, and more. We also lead and nurture our own open-source projects and communities, such as Clean Insights, F-Droid, and ProofMode.

Our DevOps team has worked to define a best-of-breed approach to supporting development and deployment of secure and privacy services protecting high-risk data. We have experience deploying on Amazon Web Services, which is ISO certified for Cloud Security and Data Protection, Microsoft Azure Cloud, Fastly, and other independent hosting providers. For network security purposes, we use private virtual intranets and web application firewalls to secure access to our services. When possible, all content stored is encrypted using cryptographic keys generated using end-to-end encryption protocols, and that are only resident in the user's device or browser. We also take a privacy-preserving approach to measurement - in most cases, no full IP addresses are logged by our servers or analytics services, only country level information. Access logs are stored for the minimal amount of time necessary to operate the service, and are not shared with any third-party. All internal and external communications within our team are encrypted (TLS, VPN, SSH, OpenPGP, Signal, Matrix). All of our services require two-factor authentication access with hardware token, from authorised devices.

Implementation of regular security audits and updates ensure that security standards are upheld. In addition to our own internal auditing and manual and automated testing, we use reputable third-party penetration "Red Team" testing teams to test the security of our services on an annual basis or after major releases.

Table of Contents

Glossary of Terms

Listing of combined terminology from both general project space, along with product developer and evaluation specific terminology

- **Civil Registration and Vital Statistics (CRVS):** A well-functioning civil registration and vital statistics (CRVS) system registers all births and deaths, issues birth and death certificates, and compiles and disseminates vital statistics, including cause of death information. It may also record marriages and divorces.
- **Software Bill of Materials (SBOM):** list of all the open source and third-party components present in a codebase. An SBOM also lists the licences that govern those components, the versions of the components used in the codebase, and their patch status, which allows security teams to quickly identify any associated security or licence risks.
- **Digital Public Good (DPGs):** are public goods in the form of software, data sets, AI models, standards or content that are generally free cultural works and contribute to sustainable national and international digital development. Several international agencies, including UNICEF and UNDP, are exploring DPGs as a possible solution to address the issue of digital inclusion, particularly for children in emerging economies.
- **Tier 2 Data Center:** A Tier 2 data centre has a single path for power and cooling and some redundant and backup components. It has an expected uptime of 99.741% (22 hours of downtime annually).
- **Penetration Test (Pen Test):** an authorised simulated cyberattack on a computer system, performed to evaluate the security of the system. The test is performed to identify weaknesses (also referred to as vulnerabilities), including the potential for unauthorised parties to gain access to the system's features and data, as well as strengths, enabling a full risk assessment to be completed.
- **DevSecOps (Development, Security, Operations):** a practice in application security that involves introducing security earlier in the software development life cycle. It also expands the collaboration between development and operations teams to integrate security teams in the software delivery cycle and workflow of continuous integration and continuous delivery (CI/CD).
- **Static Application Security Testing (SAST):** is a set of technologies designed to analyse application source code, byte code and binaries for coding and design conditions that are indicative of security vulnerabilities. SAST solutions analyse an application from the “inside out” in a non running state. One of the most mature application security testing methods in use, is white-box testing, where source code is analysed from the inside out while components are at rest.
- **Microservices:** an approach to application development in which a large application is built from modular components or services. It enables the continuous delivery/deployment of large, complex applications and consists of loosely coupled services which implement business capabilities.
- **FHIR (Fast Healthcare Interoperability Resources):** is a standard for healthcare data exchange, published by HL7 (Health Level Seven International), a standards development organisation for healthcare IT. It is designed to enable the exchange of healthcare

information between different healthcare IT systems, including electronic health record (EHR) systems, healthcare applications, and mobile devices.

- **Software as a Service (SaaS):** which means software is hosted by a third-party provider and delivered to customers over the internet as a service. It is a software licensing and delivery model in which software is licensed on a subscription basis and centrally hosted. An independent software vendor (ISV) may contract a third-party cloud provider to host the application, which is hosted on remote servers, maintained and updated by the service provider, and made available to customers via web browsers, mobile apps and APIs.
- **Gitflow:** is an alternative Git branching model that involves the use of feature branches and multiple primary branches. Fundamentally, it is a branching strategy aimed at simplifying release management by isolating your work into different types of git branches.
- **Comma Separated Value (CSV):** Data stored in a plain text, spreadsheet-like format, with columns delimited by a comma other defined character token
- **JavaScript Object Notation (JSON):** A lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.
- **Logical Volume Management (LVM):** It is a system of managing logical volumes, or filesystems, that is much more advanced and flexible than the traditional method of partitioning a disk into one or more segments and formatting that partition with a filesystem.
- **Secure Shell (SSH):** A network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

Evaluation Summary

Executive Summary

The following evaluation of OpenCRVS was conducted by Guardian Project over the course of Spring and Summer 2023. The evaluation was designed to provide a comprehensive review of the OpenCRVS solution through the following areas:

- Initial Assessment of Solution, Assets, and Documentation
    - Meet with and interview the product team, receive a typical walkthrough demonstration of the system, gather all available documentation, reports, source code, tools, and complete an overall review of the “fitness” of the solution and readiness for proceeding through the rest of the audit process
- Source Code Security Audit

    - Uncover flaws in the application (bugs, security weaknesses, extensibility, maintainability...), and evaluate the readiness of the source code for being enhanced by a third party
- Application Architecture Audit
    - Review the structure of the application, on how the different components, database, APIs, and third-party libraries interact within the code under the lens of maintainability, performance at scale, re-usability, flexibility, cyber security, and data privacy.
- Penetration Testing Audit
    - Evaluate the holistic approach in terms of cyber security, through active and passive security scanning of vulnerabilities, manual penetration testing, security policies analysis, Analysis of history of public vulnerabilities, analysis of security guidelines/documentation (including resilience and recovery recommendations), and more.
- DevSecOps Analysis
    - Software development operation best practices and from the operations management from a system administration perspective, and provide guidance for keeping in production a solution in a stable, updated, and secure perspective.

Overall we find the OpenCRVS product to be stable and ready for implementation. It has great documentation for implementers, is interoperable with many e-government solutions, focuses on real-world workflows and a team ready to help. The reliance on third-party dependencies are commonplace and the OpenCRVS choices of microservices makes for a flexible and extensible solution.

We found a few areas for improvement, which are outlined in this document, but they do not detract from the overall positive evaluation of OpenCRVS. OpenCRVS is new to the CRVS market (debuted in 2019). It is only a “version 1” generation release, and may not be as flexible or feature rich as other potential solutions. It does benefit from a laser focus on the eCRVS functionality, modern and clean architecture, and an extremely organised and well-run public open-source project. When thinking about how this might be deployed and maintained throughout the world, the idea that a single unified upstream codebase could be improved and updated, and then pushed out to downstream instances, is very attractive and beneficial.

To conduct the OpenCRVS evaluation we read publicly available documentation, visited their website and supporting resources (Github, documentation site, etc), we deployed our own instance and tested for vulnerabilities and dependencies, created a SBOM for the codebase, audited the architecture, deployed the software in AWS and ran multiple variations of vulnerability scans and penetration tests. We had ongoing communications with the OpenCRVS team, coordinated information and asset sharing, discussed openly any issues we found and maintained communication via Slack chat throughout the process. A number of key issues identified have already been addressed by and even resolved by the OpenCRVS team during this process.

Summary of the OpenCRVS Solution

“OpenCRVS is an open-source digital solution for civil registration, designed specifically for low-resource settings and available as a Digital Public Good,” as stated on their website. It is part of the “no one left behind” initiative ensuring all people on the planet are registered. OpenCRVS is intended to be interoperable with a government’s infrastructure, which can be deployed at the community level. The goal is that other vendors (ID contractors, governments) will implement and deploy the application themselves.

First imagined eight years ago by Plan International, OpenCRVS was designed to resolve current issues with Civil Registration and Vital Statistics (CRVS) systems such as being hard to use and administer services, not being accessible for low-resource settings, and often incompatible with other government technologies. The OpenCRVS team set out to develop and design a smoother and more user-friendly product.Recognizing how much of an impact their product will have on people's lives, the team created an open-source, human-centred design solution that fit the needs of both urban and rural communities. Additionally, since any CRVS program collects data that is PII (Personally Identifiable Information) centric, developing a secure and safe system is at the forefront of their decision making.

In 2017, the OpenCRVS prototype was demonstrated globally. In 2018, they implemented their first proof of concept in Bangladesh and 2019, followed with a proof of concept in Zambia. And in 2020 they ran their first pilot program in Bangladesh focusing on both rural and urban deployments. In 2022, they publicly released version 1.1 of OpenCRVS. All of the source code, releases, and documentation are available via their website and Github. This evaluation began by reviewing version 1.2.0 of OpenCRVS and then the subsequent updates and bug fixes implemented in v1.2.1 which is the current release. To date, up to 1.2 of the application has been Penetration tested by an independent third party certified to UK government standards. We will be doing final review and pen testing on v1.3 which is nearing public release as of July 2023.

The OpenCRVS team has been very receptive to the evaluation process, and is open to learning and receiving feedback. They have a team posture acknowledging there is always something to improve upon. And with each deployment they learn and adapt.

Designed for a target audience working in low-resource settings, OpenCRVS is meant to be scalable and configurable to fit the needs of a community. The team seeks to design a solution that speaks to the challenges of people all over the world. They work closely with communities to develop their solution and continue iterations and improvements every month. They hope OpenCRVS responds to real-life scenarios with a UI (user interface) that helps people do their jobs, while balancing local capacity.

When reviewing the publicly available documentation for OpenCRVS, we found the website to be friendly, engaging and informative. At a high-level one can understand the product and its purpose. On the case studies page one can engage with the stories of deployments, and hear from those who have been impacted by the OpenCRVS application.

- Website: https://www.opencrvs.org/
- Documentation link: https://documentation.opencrvs.org/
- Case Studies: https://www.opencrvs.org/about-us/case-studies

Elements of the Solution

OpenCRVS is intended to be part of a nation's foundational identity ecosystem and therefore must account for various deployment environments, implementing partners, technology infrastructure and digital literacy levels. It is designed to be highly accessible and available, to ensure it is functional even in remote areas.

Past CRVS systems were deemed not very easy to implement or use and often didn't account for additional user touch-points to help ease the registration process or procedures. Based on in-depth research and co-design practices, OpenCRVS seeks to consider what real-life scenarios and workflows can be learned from and then implemented into their solution via design, security features, or user flows.

Thus, the OpenCRVS application has accounted for a set of roles representing common actors often involved in civil registration. It also reflects some non-traditional actors which may help improve registration completion and overall service delivery, like trusted community leaders who are approved to record vital events within the community. For instance, one OpenCRVS user representing the *Field Agent* user role is **Maneya Mwansakilwa.** She is a nurse providing child and maternal services within the community, while also working in the Kanyama hospital in Lusaka, Zambia. She states, *"a large number of births occur in the community and these mothers often do not visit the hospital for services for their babies, they wait for community visits, [...] and so if this process can be brought nearer to the people, it will do a lot of good to them" (https://documentation.opencrvs.org/product-specifications/users/examples*).

Other user roles and use cases were found on the website and on the OpenCRVS documentation site (https://documentation.opencrvs.org/product-specifications/users). However, there was not a lot of definition around the 'types' of people implementing these roles and what it would look like for them practically (to use the OpenCRVS system in their daily workflow).

List of user roles from the OpenCRVS documentation site:

Role	**Responsibilities**	**Types**
Field Agent	● Create birth and death notifications	Healthcare Worker Police Office Local Leader Social Worker
Registration Agent	● Create birth and death declarations ● Validate and send declarations for approval ● Issue certificates ● View performance statistics	Civil Registration Authority Employee - Data Entry
Registrar	● Create birth and death declarations ● Approve and register declarations ● Issue certificates ● View performance statistics	Civil Registration Authority Appointee
National Registrar	● Create birth and death declarations ● Approve and register declarations ● Issue certificates ● View performance statistics	
Local System Admin	● Create users	

	● Edit users	
National System Admin	● Config management ● Create users ● Edit users	
Performance Manager	● View performance statistics	

To understand the OpenCRVS product more deeply, the team provided us with a live demo over video conference screen sharing. The demo implementation takes place in an imaginary, low-resource country named Farajaland, where OpenCRVS is deployed at the district level. Farajaland is divided into four districts with multiple provinces, but has one office per district. Within each office there is a registrar assistant & community leader. The deployment is meant to simulate a rural district with low population density who have limited data and mobile phone coverage. In the demo, they highlighted three new service delivery models which help contribute to higher completion levels.

The three service delivery models illustrated were:

1) Declaring a vital event at the community level via a community leader

Declaration being a full form with all the mandatory information necessary to register a vital event and any complementary attachments are sent to the registration office for review. They are not registering events, but rather capturing vital information. This enables capturing at the local level and can be completed by any trusted actor who identifies the birth date (health workers, local leaders, local organisations).

2) Notifications of a vital event from the health facility or hospital through an automated feed from the health system

Usually partial or minimal information collected. Can't often attach the necessary documents needed to 'authenticate registration'.

3) Direct registration at a District Regional office

All the information collected on site makes the process as simple and fast as possible.

During the demo we understood a little more how user roles are defined within the OpenCRVS platform and their workflows. We were introduced to three users: a community leader, registration officer and registrar.

- **Community leader or field agent**–a trusted individual, who can declare the births and deaths that take place in a community. They can complete the form, attach any supporting documents and send along the form to the registration office.

- **Registration officer**- works in the registration office, assists the registrar with reviewing applicants, inputting data, sending for registration and printing certificates *(don't need to use this user, but it helps demonstrate the realities of the actual process. In many cases there is another person, ie the registration agent or officer who actually does the work for the registrar).*

- **Registrar**–formally appointed by the Civil Registration Authority to register vital events into the system.

During the demo, we walked through the service delivery model #1, where declaration of a vital event is done at the community level by a community leader/field agent. The OpenCRVS interface uses iconography, colours, copy, avatars and a ticket tracking system (tracking when a ticket is retrieved, viewed or edited) to assist the user in creating their registration report. Based on community feedback, the OpenCRVS team found it was important to notify mothers (or other informants) more about the registration process (what's happening, what documents are needed for verification and why). So, at the beginning of a report, the field agent is encouraged to send an SMS to the informant. Once a registration report is started, the field agent captures all the mandatory data they can and attach any documents. This report is then ready for review by the Registration Officer and/or Registrar.

A benefit of using OpenCRVS is that you can collect data and upload attachments while offline. When you come back online the system syncs with your collected information. To see how a registration flows through the system, visit the Status Flow diagram on the documentation site, which illustrates all the vital event record statuses in OpenCRVS and how it is possible to move from one to the next (https://documentation.opencrvs.org/product-specifications/status-flow-diagram).

OpenCRVS is receptive to feedback and has integrated solutions and design considerations from proofs of concepts into the various registration workflows; low-resource setting requirements, local capacity and the envisioned goal of making the process of generating a civil registration report as simple as possible.

The OpenCRVS public good is being considered as a Civil Registration and Vital Statistics (CRVS) system that records the details of all major life events, such as births, deaths, marriage and divorce. Currently the OpenCRVS product only allows for registration of births and deaths. The team is aware of their need to include marriages and divorces into their product solution to be CRVS compliant and are working to include these features into their next releases. OpenCRVS issues product releases every 4 months with 6 months of bug fix (hot-fix) support (https://documentation.opencrvs.org/general/releases).

Currently, the team's primary funding is through implementations, which include ongoing support and maintenance, which can consume limited time and resources. A more sustainable long-term model would include more core investment and support. Their product roadmap is published on their documentation site (https://documentation.opencrvs.org/general/product-roadmap). Future product releases will include many new features. Although this set of features are on the OpenCRVS radar, they will need to determine scope and priority in implementation. While the roadmap exists, there are no fixed dates defined.

OpenCRVS have detailed their core, support, admin and data functions of their application on their documentation site:
https://documentation.opencrvs.org/product-specifications/core-functions

Some current features:

- Free and open-source, with no licence fees or ties to specific vendors
    - *However, there are a number of other costs that need to be considered when evaluating the Total Cost of Ownership (TCO) of a digital CRVS system. This includes the upfront costs to develop and rollout the system at a National scale, but also the long term costs that will be incurred year-on-year to operate and maintain the system.*
- Interoperable with other e-government systems via the FHIR standard, a documented JSON API, and webhooks.
    - Support for an open documented interface and data format that another system (such as DHIS2 or other custom application) could write code against to interact with the CRVS data. This could be useful for other public health applications, identity verification for documents, or school registration, and so on. By supporting FHIR and the other interoperability standards, it makes it possible for unnamed, unknown future apps to use this data, as authorised.
- Enables new models of civil registration that can help achieve universal registration
    - The 3 service delivery models captured during the demonstration
- Multilingual-English and French languages supported in default
    - Countries have an official language of operations so depending on where it is implemented localization is possible
    - R to L is not currently supported

- Optimised for mobile-centric communities
- Officially available as a Digital Public Good
- Detailed access controls based on different user roles
- Multi-Layered Security Capabilities through
    - Application firewalls
    - Network security through TLS/SSL protocols and certificate
    - Database firewall
- Scalable
    - The software is designed in a way that is both modular and able to be deployed across multiple virtual or physical servers that each can be increased in size and load capacity as needed. It can be run off a single physical box in a physical government data center, or run in a corporate cloud, that can scale it up or down on demand.
- Quick & Easy to configure (less than 1 week)
    - *Notes on how to set up an instance are detailed on their doc site: https://documentation.opencrvs.org/setup/1.-establish-team*
- Ability to capture data offline, then system syncs when re-connected to the internet
    - This is achieved even without native mobile or desktops apps, through use of Progress Web App architecture available in modern web browsers
- Two Factor Authentication (2FA) required for login
    - *Currently SMS authentication is the only way to authenticate. 2FA codes are sent to the user's mobile device in order to log in. To help prevent brute force attacks, these codes time out after 5 minutes.*
    - *Exploring other alternatives*

Features to be included in the next releases of OpenCRVS:

- Marriage and divorce registration flows
- Foetal death registration flow
- Web portal for direct public services
- Certified copy application flow
- Verify a record (API)
- Verify a certificate (via QR code)
- Vital statistics export
- Printed performance management reports
- Advanced configuration of forms, certificates, application settings and communications (during live operations)
- Advanced deduplication
- Integrated payments
- Delegated authority
- Validation overrides and approval (where outside normal bounds)
- Integrated learning modules
- Legacy data import support (digital / paper)
- Person centric views of vital events data
- Social protection system interoperability

Threat and Risk Assessment

Current understanding of the environment threats and risks that the evaluation is being considered within, with some examples of threats being considered under this evaluation.

Threat	**Likelihood**	**Impact**	**Severity**	**Mitigations**
Describe the potential threat, attack vector, bad actor	*How likely is it that this could happen?*	*What will happen if the threat/attack is successful?*	*How severely will the solution instance be impacted?*	*How does the solution reduce the risk, impact, and severity of the attack?*
Identity theft or fraud	Likely	Personal data, including that of children, is increasingly in demand by identity thieves	Moderate	-Multiple roles that limit the scope of access to data and capabilities. -Use of 2 factor authentication and other measures to defeat brute force attacks
Privacy Violation	Moderately likely	Digital transmission, networked storage and increased sharing of birth data may expose personal information to individuals and uses that are against the wishes of families participating in registration	Minor	-Encryption of data on the network and at rest -Multiple roles that limit the scope of access to data and capabilities.

Targeting based on personal characteristics	Very unlikely	The ability to rapidly gather and process large amounts of population data could contribute to targeted advertising, other forms of exploitation, and targeted physical threats and violence.	Severe	-Multiple roles that limit the scope of access to data and capabilities - Limit in user experience for mass search and export
Personal security violation or exploitation	Moderately likely	Registration happening outside a controlled institutional environment, such as a hospital or registrar's office, could place families at risk of physical violence and economic or other exploitation by registration agents.	Severe	-Easy access to mobile interface, even with limited connectivity, keeps as much data reporting "in the system" and private as possible - Focus on defined user roles, control who has access to accounts can help fight corruption and exploitation
Incorrect or Insecure Deployment	Moderately likely	Deployment of services by unqualified staff or into unvetted or untested environments could lead to data exfiltration, watering hole attacks, and other harms to the users and administrators	Critical	- Availability of deployment docs, training, and support - "Platform" approach creates potential for an ecosystem of certified/trusted partners for deployment - Transparency of open-source and iterative development means vulnerabilities and updates can be fixed and deployed quickly

Initial Assessment Results

Initial Thoughts

General Impressions

- We have had very positive interactions and open communication with the OpenCRVS staff and team.
- We were able to quickly get access to source code, installation information, product documentation and more, enabling us to begin our work without delay.
- Security Audits
    - Based on the additional detail received in the "OpenCRVS Security Assessment Reports" #1 and #2 documents (links are provided in the Review Assets portion of this evaluation), the OpenCRVS team has engaged in effective security audit practices with external auditors. These reports show that while there were medium and low-level vulnerabilities discovered (a common occurrence in audits!), the OpenCRVS team was able to resolve and mitigate the issues within a reasonable amount of time. From the reports:
        - *"The security assessment was conducted in two rounds, first to identify and report vulnerabilities, and then reassessed to ensure reported vulnerabilities were resolved.*
          Already from the results of the first assessment, it was evident that the OpenCRVS web application had a good security posture. The web application security fundamentals were sound. However, there were
        - *medium and low level security vulnerabilities identified and reported, as well as informational improvements. In the second assessment these were verified as resolved."*
- Open-Source Components
    - A number of open source components are used within the application, including Traefik, MongoDB, ElasticSearch and InfluxDB. These are all well-known components, actively maintained and with a good security track record. Beyond their core developers they have an extensive community and it will not be too difficult to find knowledgeable people to work on resolving any issues with these components.
- Data Centers and Uptime Considerations
    - OpenCRVS is designed to be deployed in a Tier 2 datacenter due to the unavailability of higher tier data centres in the target regions. These data centres will provide rackspace, may provide access to transit providers or provide a managed uplink, and will have partially redundant power and cooling. Uptime will typically be better than 99.5%, however the application is designed for

intermittent connectivity in mind and so interruptions to connectivity should not impact the operation of the system. Interruptions to power and cooling would be more likely in this environment than in higher tier datacenters and so the application should use a robust approach to data transactions that can maintain consistency through power events. These considerations are not unique to OpenCRVS and will be applicable to all the product solutions assessed in this project.

Key Team Members and Roles

We have been connected with and met a variety of team members from OpenCRVS. We are in contact via both email and Slack group chat. We are satisfied that the sets of skills and knowledge represented by these contacts will be sufficient for us to complete our evaluation.

- Director of Product Strategy and Sustainability
- Technical Architect
- Director of Community Development & Engagement
- Marketing and Engagement Team

Summary of Initial Interactions

- Feb 28, 2023- Coordinator made introduction to product solution and evaluator
    - We followed up with a calendar invite for a kickoff meeting and shared the evaluation timeline with request for evaluation assets/inputs
- March 7, 2023- Kick-Off Meeting with Product Solution Team
    - Followed up with Slack group chat connections, sharing of additional assets/inputs as requested
- March 14, 2023 - Live Product Demo over Zoom (Raw Notes here: https://docs.google.com/document/d/17CWjDPqyJZ6s4IBHlLxXgCRratHvSgR-sKimZ9UedXE/edit?usp=sharing)
- March 29, 2023 - Check-in and API / Component Review with OpenCRVS team

Concerns and Blockers

- We currently do not have any concerns or blockers regarding our ability to move forward and complete the evaluation process.
- We have begun to form a list of specific questions and lines of inquiry that we have developed based on our time reviewing the assets (below) and from the live demo we received. These are presented later in this section.

Review of Assets

- ☐ Overall project website https://www.opencrvs.org/
- ☐ Functional architecture shows the logical components of which the system is comprised: https://documentation.opencrvs.org/product-specifications/functional-architecture
- ☐ Technical Architecture: https://documentation.opencrvs.org/technology/architecture
    - ☐ The technical architecture of OpenCRVS was designed to conform to the Open Health Information Exchange (OpenHIE) architectural standard and interoperate using HL7 (Fast Healthcare Interoperability Resources) or FHIR. FHIR is a global standard application programming interface or (API) for exchanging electronic health records.
- ☐ The status flow diagram shows all the vital event record statuses in OpenCRVS and how it is possible to move from one to the next https://documentation.opencrvs.org/product-specifications/status-flow-diagram
- ☐ The different user roles in the system, are defined below, reflect common actors involved in civil registration services around the world as well as non-traditional actors that may help improve service delivery https://documentation.opencrvs.org/product-specifications/users
    - ☐ Example Users: https://documentation.opencrvs.org/product-specifications/users/examples
- ☐ Demo Roles / Logins for Farajaland demo instance: https://github.com/opencrvs/opencrvs-core#log-into-opencrvs
- ☐ Plan, Intl, Identifying and addressing risks to children in digitised birth registration systems: https://www.ohchr.org/sites/default/files/Documents/Issues/Children/BirthRegistrationMarginalized/PlanInternationalGeneva_4.pdf
    - ☐ *"This document provides guidance on identifying and mitigating these risks for implementing government agencies and their partners operating in low- and middle-income countries. It expands on the model of DBR developed by Plan International as part of its Count Every Child initiative and within the context of strengthening civil registration and vital statistics (CRVS) systems more broadly."*
- ☐ Open-source project hosted on Github: https://github.com/opencrvs/
- ☐ opencrvs-core Public
    - ☐ A global solution to civil registration; A digital public good for civil registration
- ☐ opencrvs-farajaland Public

- ☐ An example configuration module for OpenCRVS using a fictional country
- ☐ opencrvs-mec Public
    - ☐ An example configuration for OpenCRVS using a fictional country
- ☐ performance-tests Public
    - ☐ Performance tests for OpenCRVS
- ☐ mosip-mediator Public
    - ☐ This is an example microservice mediator that subscribes to an OpenCRVS BIRTH_REGISTRATION Webhook.
- ☐ hearth Public
    - ☐ A fast FHIR-compliant server focused on longitudinal data stores.
- ☐ Github Issue list for core OpenCRVS codebase https://github.com/opencrvs/opencrvs-core/issues
- ☐ Community Forum: https://community.opencrvs.org/
    - ☐ "This is a place for everyone interested in CRVS system strengthening to engage in discussion and healthy debate to explore how we can move the dial on civil registration systems globally."
- ☐ Case Studies: https://www.opencrvs.org/about-us/case-studies
- ☐ Product Council discussion "Requirements for Registering a Marriage and Divorce in OpenCRVS" https://community.opencrvs.org/t/requirements-for-registering-a-marriage-and-divorce-in-opencrvs/167
    - ☐ "We welcome additional contributions to the questions on this Mural board MURAL 2"
- ☐ OpenCRVS Interoperability Hackathon https://www.opencrvs.org/resources/connect/news-and-events/opencrvs-interoperability-hackathon
    - ☐ "We hosted a briefing session on March 9th for system integrators, ID vendors, Digital Public Goods and other organisation s interested in participating in the OpenCRVS Interoperability Hackathon"

- ☐ https://documentation.opencrvs.org/technology/security
    - ☐ Every release of the OpenCRVS application and infrastructure has been security penetration tested by an independent, CREST and CyberEssentials certified 3rd party to UK government standards.
    - ☐ The latest penetration test of OpenCRVS was performed by GoFore - NORAD's preferred security testing provider.

    - ☐ "Already from the results of the first assessment, it was evident that the OpenCRVS web application had a good security posture. The web application security fundamentals were sound."
- ☐ Security Assessment Report #1 https://drive.google.com/file/d/1TmaH06vpqpTnLJaZzTPYGJE9DOPD0kXn/view?usp=sharing
    - ☐ Security Assessment Report #2 for verification of mitigation: https://drive.google.com/file/d/1FhhllHkxy-8HncxgB5aoABlTJ7iD3gRx/view?usp=sharing
- ☐ Bangladesh deployment case study: https://www.opencrvs.org/about-us/case-studies/bangladesh
    - ☐ "The Bangladesh Computer Council found the OpenCRVS implementation to be highly secure and effective against cyber security threats."
- ☐ "Explore our implementation resources to understand how OpenCRVS can be implemented in different country contexts and the associated costs" https://www.opencrvs.org/resources/implementation-guidance
- ☐ Information on a "Proof of Concept" deployment: https://www.opencrvs.org/resources/implementation-guidance/what-is-a-poc
- ☐ Guide on "Total Cost of Ownership" for OpenCRVS implementation: https://www.opencrvs.org/uploads/images/Understanding-the-costs-of-an-OpenCRVS-Implementation.pdf
    - ☐ "OpenCRVS can now be configured within just 1 week, making digital civil registration more accessible than ever before"
- ☐ "This section introduces the installation steps that your Technical System Administrator will be required to run" https://documentation.opencrvs.org/setup/3.-installation
- ☐ Monitoring: https://documentation.opencrvs.org/setup/7.-monitoring

Source Code Security Audit

Processes and Tools

Research and document the complete "Software Bill of Materials" (SBOM)

We used the open-source tool 'syft' to create a Software Bill of Materials for the entire opencrvs-core repository and also for each Docker image produced from that repository. The advantage of the former is that we get an overview of all NPM packages used in any of the OpenCRVS microservices. With the latter we can look at only the packages that appear in the final Docker image for a given microservice. The SBOMs in JSON format are available here: https://drive.google.com/drive/folders/1gmNYA2DeaKXzV_f7lDmdJBF1bw_n2bTH

Open-Source Software (OSS) vulnerability scanning

We used the open-source tool 'grype' to check the source and Docker images for vulnerabilities. This tool demonstrated several vulnerabilities, though most were not critical or were awaiting a fix at the operating system level. One that appears to be an important and relatively easy fix is to update the parse-url NPM package to the latest version. This would solve several medium-level vulnerabilities and one critical vulnerability.

The vulnerabilities discovered were reported to the OpenCRVS team, with additional data provided in the issue filed in their bug tracking system.

Static application security testing (SAST) scanning

We used the open source SAST scanner 'semgrep' to perform an automated analysis of the Typescript/Javascript codebase. This analysis turned up three vulnerabilities deemed 'critical', but one appears to be a false positive.

The vulnerabilities discovered were reported to the OpenCRVS team, with additional data provided in the issue filed in their bug tracking system.

High-level concerns, issues

Our overall impressions of the OpenCRVS Core codebase and its security posture are positive.

1. The codebase is primarily written in Typescript, a typed superset of Javascript that is the industry-standard for frontend web development and also very common for backend applications. Typescript makes large Javascript codebases easier to maintain and is well-suited to OpenCRVS's requirements.
2. The frontend framework is React (also an industry-standard) and GraphQL is used for data-fetching. JWT is used for authorization and Docker is used for deployment. The ubiquity of these choices make the codebase approachable by outside developers seeking to contribute to the project or extend it.

3. The project has good automated test coverage and the development team follows best practices when it comes to implementing and testing new features.
4. There is evidence in our direct interactions with OpenCRVS developers and historically in the project's issue tracker that security concerns are handled in a timely, serious and open manner.

Summary of test environment setup, steps taken to complete analysis

In order to complete this assessment we read application source code, deployed a development instance as described in the project's documentation and ran standard open source tools for automated dependency and vulnerability checks. We also produced a Software Bill of Materials (SBOM) for the OpenCRVS codebase as a whole and for all of its constituent Docker images as a snapshot of the dependencies as of version 1.2.0.

Summary of communication with vendor related to disclosures and direct feedback

In the course of our audit we found a small number of issues that we felt should be addressed by the OpenCRVS developers. We shared our concerns during a video call and in an internal Slack chat and are satisfied that these issues have received appropriate attention.

Status of any mitigations, patches, updated releases

We are currently tracking two Github issues related to our suggestions:

- Potential password and SMS code vulnerabilities: https://github.com/opencrvs/opencrvs-core/issues/4979
- NodeJS version nearly EOL: https://github.com/opencrvs/opencrvs-core/issues/5010
  - They will update to Node 16 in their new release v1.3 https://github.com/opencrvs/opencrvs-core/pull/5055

Outcomes

Below are outcomes of the use of the various manual and automated source code audit processes and tools. This includes disclosures of any vulnerabilities, bugs, typos, threats, etc, which were all shared with the OpenCRVS team via our communication channel and the public Github project, as needed.

These are the issues we discovered by manually reviewing the source code (version 1.2):

- The NodeJS version is nearly at the end of its long-term support (LTS). OpenCRVS requires NodeJS version 14 which will reach end-of-life (EOL) at the end of April 2023. This means that no further security fixes will be provided for newly discovered vulnerabilities. As a second order effect, many NPM packages remove support for EOL versions of Node. This may lead to a situation where critical vulnerabilities have been patched in a required dependency, but the updated code can't be installed. We recommend trying to stay on the newest LTS version of Node, which at this time is version 18 (supported until April 30, 2025). The related Github issue is here: https://github.com/opencrvs/opencrvs-core/issues/5010

  Update: *On April 28, 2023 OpenCRVS informed us that they will migrate to NodeJS 16 as part of their upcoming version 1.3 release and provided a link to the updated code: https://github.com/opencrvs/opencrvs-core/pull/5055*

- Use of insecure random source in SMS code generation which could result in predictable 2FA. It should use a cryptographically secure source of randomness. The code in question is here: https://github.com/opencrvs/opencrvs-core/blob/v1.2.0/packages/auth/src/features/verifyCode/service.ts#L43 and the related Github issue is here: https://github.com/opencrvs/opencrvs-core/issues/4979

- SHA512 is used for password hashing. While SHA512 is cryptographically secure against collisions, it is fast to brute force on modern hardware. A key derivation function specifically designed for password hashing would be a much more secure choice. The recommended standards here are PBKDF2, bcrypt, or scrypt with an appropriate amount of iterations. The code in question is here: https://github.com/opencrvs/opencrvs-core/blob/v1.2.0/packages/user-mgnt/src/utils/hash.ts and the Github issue is here: https://github.com/opencrvs/opencrvs-core/issues/4979

- We have an open question about how JWTs are revoked when users are deactivated. There is evidence in the codebase that a list of revoked tokens is stored in Redis, but when we look at places in the code where tokens are validated, we don't see tokens checked against this revoked token list.

    - From OpenCRVS team: “If you look at this function here: https://github.com/opencrvs/opencrvs-core/blob/ocrvs-5562/packages/commons/src/token-verifier.ts#L34, it checks for invalid tokens using the "verifyToken" endpoint that we have in our auth microservice. And this function is used in the auth strategy of our user facing services (i.e. config, gateway & webhooks)”

- Another open question: it appears that given the UUID of an attachment (which could be a birth or death certificate containing personally identifiable information) it is possible to retrieve the document without a JWT check. Is this correct and are there plans to change it?
    - From OpenCRVS team: “We have addressed this issue here: https://github.com/opencrvs/opencrvs-core/pull/4970. Now clients will need a presigned URL to access any attachments which requires a valid JWT”

Readiness of the source code for being enhanced by a third party

OpenCRVS is built on a microservice architecture where each element of the larger application (authentication, document handling, user management) runs as small independent services. This architecture is generally a positive for future extensibility by third parties, with some caveats.

At first glance the proliferation of mini-applications can be overwhelming when compared with a more traditional monolithic application where all code works together as part of a single large application. Monoliths are usually considered easier to deploy and test, but they come with some downsides for extensibility. In a monolithic application, changes in one section of the app can have negative consequences in other seemingly-unrelated parts of the application. Often this means that a monolith can require a developer’s understanding of the larger codebase before they can confidently extend and modify it.

By contrast, the microservice architecture is friendlier to extension despite its logistical complexity. If a developer wanted to create, for example, an alternate API to query the data held by OpenCRVS, they could create a microservice to enable that use without necessarily understanding how all of the other microservices work. They could also confidently add this service and have the expectation that it will not interfere with other parts of other applications.

The one area where it truly is essential to understand and emulate the existing OpenCRVS codebase is in the area of authorization. The microservices uniformly use verified JWTs for authorization and unless the system to be integrated specifically requires a different form of authorization, we recommend adopting the same standard.

The OpenCRVS team also expressed an interest in accepting patches and possibly even additional features from developers outside their organisation. This is another positive point for extensibility and the future viability of the platform.

Application Architecture Audit

High-level concerns, issues

Our overall evaluation of the OpenCRVS application architecture is **positive**.

While there are some areas where documentation could be improved, the overall architecture is sound from both an application developer's and operational engineer's perspective. The microservices architecture is flexible and scalable, and the use of third-party software packages and services provides a solid foundation for the system. However, the distributed nature of the architecture and the use of many third-party dependencies mean that the system is complex and will require specialised expertise to manage and maintain effectively. This complexity of the system architecture itself is not a negative point, as any scalable and reliable production-level system will inherently have some level of complexity, but the requirement for capable and experienced operational expertise should not be ignored.

Summary of test environment setup, steps taken to complete analysis

In order to complete the architectural analysis assessment we read the application source code, the infrastructure source code, and deployed a development environment as described in the documentation. We also reviewed the existing architectural documentation.

Relevant source code and documentation sources:

- Primary application and infrastructure codebase: https://github.com/opencrvs/opencrvs-core
- High level description of the application and deployment architecture https://documentation.opencrvs.org/technology/architecture
- Illustration of the components in the functional architecture https://documentation.opencrvs.org/product-specifications/functional-architecture

Summary of communication with vendor related to disclosures and direct feedback

There were no issues requiring disclosure or mitigation discovered during this portion of the assessment. However it should be noted that under scope for this assessment was the latest stable version of OpenCRVS, version 1.2. In the coming weeks version 1.3 will be released along with functional and architectural changes. We chose to limit the assessment to version 1.2 as it comprises the latest recommended and stable release of the project. We recommend that a followup assessment be conducted against version 1.3 when it is released.

Outcomes

Document a shared, holistic view of the structure of the application

Microservices Architecture

OpenCRVS is a modular and scalable web-based application developed and distributed as independent microservices. Each microservice component and every OpenCRVS component is independently scalable in private or public cloud, in large or small data centres.

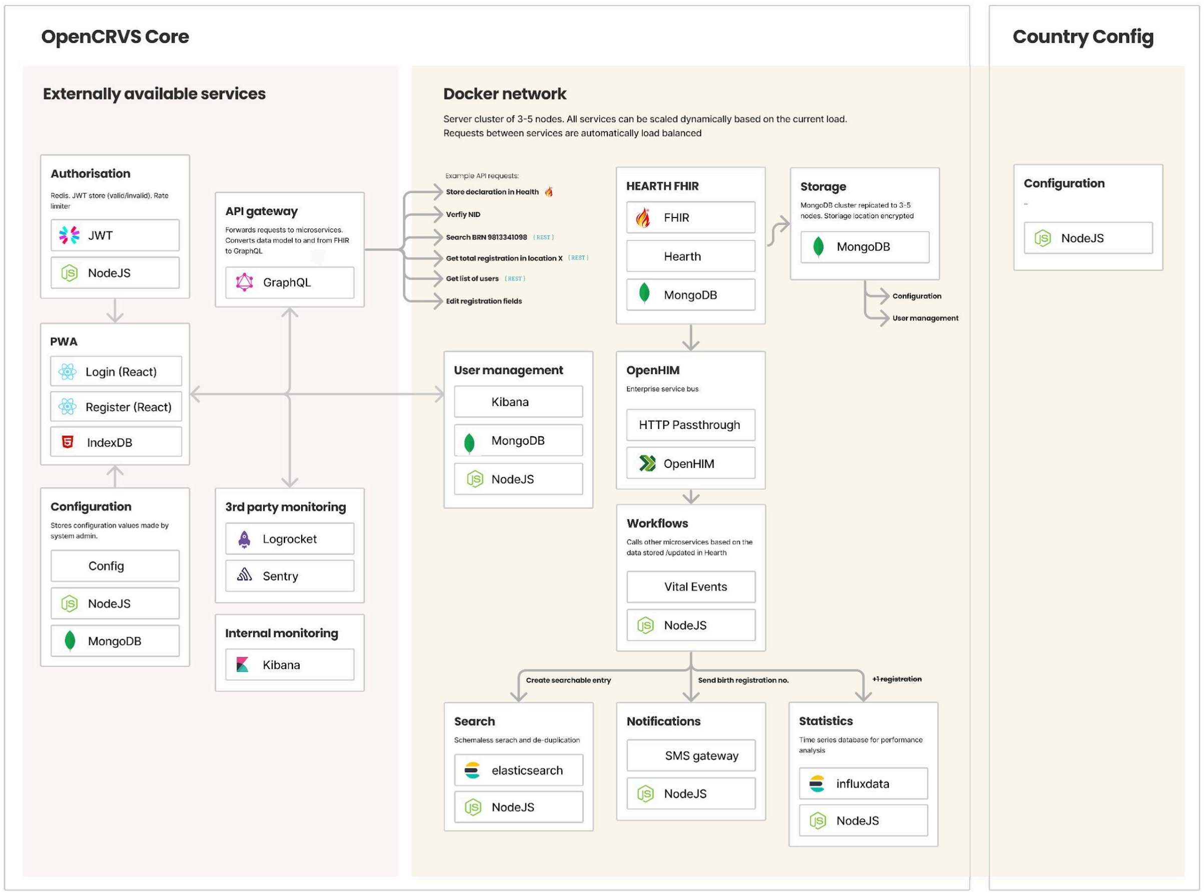
OpenCRVS Core
Country Config
Externally available services
Docker network
Server cluster of 3-5 nodes. All services can be scaled dynamically based on the current load.
Requests between services are automatically load balanced
Authorisation
Redis. JWT store (valid/invalid). Rate limiter
JWT
NodeJS
API gateway
Forwards requests to microservices. Converts data model to and from FHIR to GraphQL
GraphQL
Example API requests:
Store declaration in Health
Verfiy NID
Search BRN 9813341098
REST
Get total registration in location X
REST
Get list of users
REST
Edit registration fields
HEARTH FHIR
FHIR
Hearth
MongoDB
Storage
MongoDB cluster replicated to 3-5 nodes. Storage location encrypted
MongoDB
Configuration
User management
Configuration
-
NodeJS
PWA
Login (React)
Register (React)
IndexDB
User management
Kibana
MongoDB
NodeJS
OpenHIM
Enterprise service bus
HTTP Passthrough
OpenHIM
Configuration
Stores configuration values made by system admin.
Config
NodeJS
MongoDB
3rd party monitoring
Logrocket
Sentry
Internal monitoring
Kibana
Workflows
Calls other microservices based on the data stored /updated in Hearth
Vital Events
NodeJS
Create searchable entry
Send birth registration no.
+1 registration
Search
Schemaless serach and de-duplication
elasticsearch
NodeJS
Notifications
SMS gateway
NodeJS
Statistics
Time series database for performance analysis
influxdata
NodeJS

Fig: OpenCRVS Application Architecture. Retrieved 2023-04-29.
Source: https://documentation.opencrvs.org/technology/architecture

Microservices are a way of designing software applications by breaking them down into smaller, independent services that work together to provide the overall functionality. Each service can be developed and deployed independently, which makes the application easier to scale, maintain and update over time.

The use of microservices in OpenCRVS is an architectural decision that provides several benefits and disadvantages for the system. The modular, event-driven approach allows for greater flexibility and agility in the system, as each microservice can be updated and deployed independently of the others.

A microservice architecture is particularly well-suited to large-scale applications serving many users due to its ability to scale horizontally, allowing developers to distribute the workload across multiple services and servers. However, for applications serving only thousands of users, the benefits of scalability may not be as relevant, as the application may be able to function effectively on a single large server.

From an application architect's perspective, the use of microservices comes with some challenges that need to be carefully considered. Microservices increase system complexity, making it harder to develop, test, and maintain due to the additional effort required to ensure correct communication between services. Each service must handle its specific function while interacting with other services, leading to complex testing and debugging processes. Coordination and testing are also required when making changes to one service that may affect other services.

Notably, OpenCRVS has a strong end-to-end test suite. An end-to-end test suite is a set of automated tests that verify the correct functioning of an entire system or application, from the user interface to the back-end components. It is useful and important when developing applications consisting of microservices because it ensures that all services work together correctly and communicate seamlessly, reducing the risk of errors and failures. End-to-end testing helps identify potential issues and bottlenecks in the system, and ensures that the overall system meets the functional and non-functional requirements. Moreover, it allows developers to test the system in a more realistic and comprehensive way, mimicking user behaviour and real-world scenarios.

From an operational engineer’s perspective, that is, the IT engineers responsible for day-to-day operations of the application, the use of microservices requires a strong focus on infrastructure management, as each microservice needs to be deployed, monitored, and managed independently. Microservices can make life difficult for operational engineers when trying to troubleshoot problems in a running application due to the distributed nature of the services. Operational engineers must navigate the complexities of the interactions between microservices and must have a deep understanding of the system's architecture to pinpoint the source of the problem, leading to longer resolution times and increased operational overhead. This can add significant overhead to the development and deployment process, and will require additional operational and infrastructure expertise.

<u>The use of microservices overall in OpenCRVS is a solid architectural decision. However, it also requires careful consideration of the potential challenges and a strong focus on infrastructure management when managing a deployment of the application.</u>

Research and document the complete “Software Bill of Materials” (SBOM) regarding components, database, APIs, and third-party libraries

The “Software Bill of Materials” (SBOM) was delivered as part of the source code security assessment, so it will not be mentioned here.

Under scope for the architecture assessment are the various external third-party service dependencies and database systems that comprise a functional deployment of OpenCRVS.

Third Party Technology Choices and Implications on Architecture

In addition to the opencrvs-core application containing the microservices, OpenCRVS leverages many open source software packages to provide functional features. By and large, the chosen open source technologies are reliable and proven projects that allow OpenCRVS to develop and deliver a solid CRVS system with a small development team. These architectural dependencies are outlined with rationale at https://documentation.opencrvs.org/technology/architecture#open-source-dependencies

Open source dependencies which are automatically provisioned alongside the OpenCRVS Core

The following dependencies are used in docker containers in a Docker Swarm on Ubuntu.

Dependency	OpenCRVS Justification
Docker Swarm	Docker Swarm was chosen by our architects in 2018 for its lack of requirement of other essential dependencies, its close alignment with Docker and its simplicity in terms of installation and monitoring on a Tier 2 private data centre, on bare metal servers with headless Ubuntu OS. Why not use AWS, public cloud SaaS or serverless you might be thinking? ● Many countries may be located far from a high-quality data-centre above Tier 2. ● Many countries may not legally support international data storage of citizen data on a public cloud. Getting the legal approval for external storage of citizen data requires regulatory change which can take a considerable

	amount of time. ● Because some countries may not be able to maintain complex software independently, we are considering a SaaS solution, provided enough countries get regulatory approval. Over time, this situation appears to be slowly evolving and we are monitoring it closely. Previously unskilled system administrators can quickly up-skill in the techniques of private cloud infrastructure management using Docker Swarm. We wanted to democratise containerisation benefits for all countries.
Kubernetes	We are working on a Kubernetes migration now that Kubernetes has become a more mature, easier to use and configure solution, thanks to dependencies like Helm and other plugins increasing in popularity since 2018. In the meantime, Docker Swarm makes it easy to commence containerised microservice package distribution privately, automatically configures a "round robin" load balanced cluster, and provides Service Discovery out-the-box.
ElasticSearch	OpenCRVS uses ElasticSearch, an industry standard, NoSQL document oriented, real-time de-duplication & search engine. Lightning fast, intelligent civil registration record returns are possible, even with imprecise “fuzzy” search parameters. De-duplication management to ensure data integrity is essential to any civil registration system. A fast search engine lowers operational costs and improves the user experience for frontline staff. ElasticSearch is also used with Kibana for application and server health monitoring.
Hearth MongoDB Database Layer	In order to support configuration for limitless

	country scale, OpenCRVS was designed for NoSQL, built on MongoDB, and aligned to a globally recognised healthcare standard. Massively scalable and extensible, Hearth is an OpenSource NoSQL database server originally built by the OpenCRVS founding member Jembi Health Systems, using interoperable Health Level 7 FHIR v4 (ANSI Accredited, Fast Healthcare Interoperability Resources) as standard. We extended FHIR to support the civil registration context. Our civil registration FHIR standard is described here.
InfluxData	The hyper-efficient Influx "time series database" is used in our stack for "big data" performance insights. Millisecond level query times facilitate civil registration statistical queries over years of data, disaggregated by gender, location and configurable operational and statistical parameters.
OpenHIM enterprise service bus, interoperability Layer	The OpenHIM (Health Information Mediator) is a NodeJS enterprise service bus designed to ease interoperability between OpenCRVS and external systems such as Health & National ID. It provides external access to the system via secure APIs. OpenHIM channels and governs internal transactions, routing, orchestrating and translating requests into FHIR between services and the database layer.

Conclusions

The choice of Traefik for ingress, MongoDB as a database, ElasticSearch for search, Minio for object storage, and InfluxDB for statistics storage are solid choices. However, the use of multiple dependencies also increases operational complexity, as each tool requires specific expertise and management for operation at scale. Despite the added complexity, the use of these tools provides essential functionality for OpenCRVS, including secure and reliable data storage, fast and flexible search capabilities, and comprehensive statistics tracking.

One notable third-party technology dependency used by OpenCRVS is the Hearth database. Hearth is a fast FHIR-compliant server used as a datastore for data in OpenCRVS.

FHIR (Fast Healthcare Interoperability Resources) is a standard for healthcare data exchange, published by HL7 (Health Level Seven International), a standards development organisation for healthcare IT. It is designed to enable the exchange of healthcare information between different healthcare IT systems, including electronic health record (EHR) systems, healthcare applications, and mobile devices. FHIR is built on modern web technologies, making it easy to implement and use in a variety of different settings.
Hearth is a server implementation of the FHIR standard built on top of MongoDB. It was originally developed by Jembi Health Systems, but they have stopped maintaining and developing the project (https://github.com/opencrvs/opencrvs-core/issues/3704). The OpenCRVS team has taken over maintenance of Hearth for their own usage, applying security patches as needed. The OpenCRVS team has expressed a roadmap for migrating away from Hearth to another solution, but this is still in progress with a target of version 1.5.

As Hearth is a niche third-party dependency focused on FHIR compatibility, it poses a moderate risk to the health of the OpenCRVS project. Migrating a datastore out from underneath a functioning application is a non-trivial task, but after discussions with the OpenCRVS team, we believe that they will be able to accomplish this without disruption to implementers of OpenCRVS. Despite the risks involved, it's important to note that Hearth remains a critical component of OpenCRVS at present.

External Third Party Services

OpenCRVS relies on several third party services for functional requirements and operational requirements.

On the functional side, OpenCRVS recommends and assumes that you will be using **Contentful** (https://contentful.com) to manage the user-readable text in a deployment of OpenCRVS.

Contentful is a headless content management system (CMS) that provides a content infrastructure for digital products and services. It allows developers and content creators to manage and deliver content across multiple channels and platforms, providing a flexible and scalable solution for content management. It is a proprietary (not open source nor free software) subscription based service which implementers will need to subscribe to and train on.

This dependency is not explicitly stated in the high-level architecture documentation, and instead buried inside the deployment manual (https://documentation.opencrvs.org/setup/3.-installation/3.2-set-up-your-own-country-configuration/3.2.7-set-up-language-content). We would like to see the OpenCRVS team bring the Contentful dependency into the high-level architecture discussion with a discussion of trade-offs, price and how it affects total-cost-of-ownership.

Importantly, the linked documentation states that there is not a hard dependency on Contentful, but "it requires professional expertise in NodeJS if you want to set up a content management system other than Contentful." How exactly to set up such a system is left as an exercise for the reader.

Operationally OpenCRVS depends on **Logrocket**, **PaperTrail**, and **Sentry** for application monitoring and logging. Tools that fulfil this role are essential for application monitoring and debugging, because they provide operational engineers and developers real-time visibility into how their applications are performing and where errors are occurring.

LogRocket is a session replay and error logging tool that provides detailed insights into user behaviour and errors in web applications. It allows developers to see exactly what users are doing on the website and what errors they encounter, enabling them to quickly identify and fix issues. LogRocket is a proprietary (not open source nor free software) subscription service that implementers will need to subscribe to.

Sentry is an open-source error tracking platform that provides real-time error reporting, alerting, and performance monitoring for web and mobile applications. It helps developers identify and fix errors before they affect users, reducing the risk of downtime and improving user experience. Sentry is open-source and can be deployed on premise with no software licensing fee, however this will require additional operational expertise and increase total-cost-of-ownership (TCO). Sentry is also available as a paid subscription service.

PaperTrail is a cloud-based log management and monitoring tool that provides real-time visibility into application logs and system events. In OpenCRVS, PaperTrail is used to manage and monitor the logs generated by the microservices architecture. By centralising logs in a single location, PaperTrail enables developers and operational engineers to quickly identify and troubleshoot issues that arise within the system. PaperTrail is a proprietary (not open source nor free software) subscription service that implementers will need to subscribe to. It's worth noting that PaperTrail is not deeply integrated into the OpenCRVS architecture and could be swapped out for an alternative log aggregator.

PaperTrail, Sentry and LogRocket are critical for maintaining the stability, security, and performance of web applications, and are used by organisations of all sizes. They are known trusted players in their space. It was a sound decision by the OpenCRVS team to leverage these tools, but the role they play in the TCO and regulatory compliance and data privacy should not be ignored.

When using SaaS-based tools like Contentful, LogRocket, PaperTrail, and Sentry, there are implications to consider regarding regulatory compliance and data privacy. As these tools are hosted by third-party providers, it is important to ensure that the service providers are compliant with applicable regulations, such as GDPR in the EU or HIPAA in the US, and that appropriate measures are in place to protect sensitive data. Implementers must also ensure that they are using the tools in accordance with their own internal policies and procedures to maintain data privacy and security. Failure to properly manage data privacy and security in the context of SaaS tools could lead to compliance violations, data breaches, or other legal and reputational risks.

OpenCRVS's dependence on these proprietary systems is significant due to the essential functions they perform for application monitoring, logging, and error logging. While there are open-source alternatives (or even other proprietary alternatives) available, these would not be supported by OpenCRVS out-of-the-box. Transitioning to these alternatives would necessitate custom software development to ensure integration and functionality, representing a substantial investment of time, resources, and operational adjustments.

If regulatory constraints or IT infrastructure policies prevent the use of these external proprietary dependencies, deploying OpenCRVS would still be possible, but it would involve some substantial changes and challenges, particularly concerning operations and information security.

In terms of operations, the loss of these tools would mean finding and implementing alternative solutions that fulfil the same roles, and these replacements would need to comply with the given regulations or policies. This could involve utilising open-source tools, as mentioned in the previous response, or building custom solutions in-house. Both of these options would involve significant time and resources, particularly the latter. Moreover, there would be a learning curve associated with using new tools, and an adjustment period as developers and engineers adapt their workflows.

From an information security perspective, in-house solutions or on-premises deployments of open-source tools could potentially offer more control over data, which could be beneficial in terms of meeting strict data privacy and security regulations. However, this also means the organisation would assume full responsibility for securing that data, as opposed to relying on the security measures provided by a third-party SaaS provider. This could necessitate hiring or training security experts, implementing additional security measures, and staying up-to-date with the latest security threats and best practices.

It's also important to consider the cost implications. While in-house or open-source solutions might avoid SaaS subscription costs, they could increase costs in other areas. For example, in-house solutions would require development and maintenance resources, while on-premises deployments would necessitate investing in hardware and infrastructure, as well as the personnel to manage and maintain them.

In summary, while it is possible to deploy OpenCRVS without the mentioned external dependencies, it would involve significant operational adjustments, potential security implications, and possibly increased costs. A thorough analysis and understanding of the particular constraints and requirements would be necessary to plan for and mitigate these challenges.

Finally, for two-factor authentication OpenCRVS relias on an external SMS provider. The security implications of this will be discussed in the DevSecOps section later in this evaluation. However, for the purposes of application architecture it should be noted that OpenCRVS recommends the use of clickatell (https://www.clickatell.com/) and infobip (https://www.infobip.com/) as SMS delivery providers. Using an external third-party provider for SMS is a reasonable choice, it makes implementation easier and in many cases might be the only option. Thankfully the choice of SMS delivery provider is delegated to the specific country/implementer's configuration module (https://github.com/opencrvs/opencrvs-farajaland/blob/v1.2.1/src/features/notification/service.ts), so it will be straightforward for an implementer to use a provider of their choice.

Ease of user interface for setting roles and status visibility

Managing users and roles in OpenCRVS is straightforward. The only notable item here is that the names of the roles in OpenCRVS (Field Agent, Registration Agent, Registrar, etc) do not always map directly onto the roles that countries/implementers actually use. But this is a known issue and will be addressed by the team.

Relevant documentation:

- https://documentation.opencrvs.org/product-specifications/users
- https://documentation.opencrvs.org/default-configuration/opencrvs-configuration-in-farajaland/user-role-mapping
- https://documentation.opencrvs.org/setup/4.-functional-configuration/4.5-create-system-users#before-you-start
- https://github.com/opencrvs/opencrvs-core/issues/4376

Evaluation of maintainability, performance at scale, re-usability, flexibility.

Further Remarks on Application Extensibility

Building on the remarks from the “Readiness of the source code for being enhanced by a third party” section in the Source Code Security Audit, we note that the application is readily extensible by implementers of the system, which the microservice architecture is well suited for.

OpenCRVS has created a user interface component library to maintain a consistent user experience in all parts of the application. The component library is built on Storybook, the industry-standard for React component management. In addition to being useful for maintaining uniformity within the OpenCRVS application, these components would also be useful for implementers who may want to extend the application while keeping a familiar user experience.

Given the vast technical knowledge to develop on OpenCRVS however, we suggest to the OpenCRVS team that they update their documentation (which is otherwise fantastic) to include examples of customization and extensibility to provide implementers with a spring-board from which to start their customizations.

Maintainability and Performance

As discussed, the use of a microservices architecture allows for modularity and scalability, enabling each component to be updated and deployed independently of the others. The existing end-to-end testing suite gives confidence in the OpenCRVS developer’s ability to manage the complexities of a microservice architecture while delivering a reliable and performant product.

OpenCRVS is not a turn-key system and requires operational expertise to manage and maintain effectively. Any implementers of OpenCRVS will require a strong operational IT background and be familiar with (or able to learn) the various dependencies discussed above. However the same could be said for nearly any software application operating within the same functional space. OpenCRVS's documentation is extensive and includes deployment artefacts in the form of docker-compose files, which greatly reduces the burden of setup and maintenance for implementers

Penetration Testing Audit

Details of process, setup, tools utilised

We utilised cloud-based testing suites and services, alongside human teams, that offer a variety of capabilities and options for one-time and ongoing scanning and testing. While fully bespoke and custom security audits are always a valuable service, they come at a very high cost in both money and time. Our approach for this evaluation was to use tools and techniques that are both within the realm of the available budget, and provided a more dynamic, ongoing approach for uncovering vulnerabilities. We recommend this approach for use not only in the evaluation stage, but also as part of the ongoing monitoring in future eCRVS production deployments.

- Intruder.io: fast, cheap automated vulnerability scanning service, with multiple vantage points; Less feature rich, but still a good tool for initial "smoke test" results
    - **Emergent Threats** performs automated, nearly daily additional ongoing, focused scans based on newly identify threats and vulnerabilities added to the Astra database
    - **Nessus Agents** extend scanning to run within server-infrastructure from the "inside out" uncovering vulnerabilities and configuration issues that an attacker may take advantage of if they compromise a network

- Astra: Powerful tool+service providing Automated, Vetted, and Emergent Threats vulnerability scanning
    - **Automated** is machine-only scripted testing of a comprehensive set of known vulnerabilities
    - **Vetted** builds on the Automated result, then adds human review and verification of identified potential vulnerabilities to add more detail, and identity and label "false positives"

- Manual Penetration Testing
    - Building on results of automated and vetted scanning, a manual penetration test utilises the same approach, techniques, attack vectors, and known vulnerabilities, but with added creativity and skills of a human-based attacker.
    - Very few "false positive" outputs come from this step, due to the human operator understanding if they have been able to achieve a valuable

The Manual Penetration Testing of the test OpenCRVS instance occurred during the week of 26th July 2023. The testing was performed from a remote attacker's perspective with the following goals:

- To identify security loopholes, business logic errors and evaluate effectiveness of existing security controls in the application that pose a risk to the systems, infrastructure, or data.
- Recommend technical security best practices to improve security posture of the target applications audited.
- Explain the potential impact of the identified vulnerabilities, such as the extent of data exposure, potential financial losses, or reputational damage that could occur if they were exploited by malicious actors.
- Provide clear and actionable recommendations for addressing the identified vulnerabilities.

From the Manual Penetration Testing, a total of 18 vulnerabilities/recommendations were reported, with 1 High, 9 Medium, and the remaining Low or "Info".

Out of a score of 10, the highest risk score assigned to a vulnerability was 7.1, the lowest was 3, and the average score was 4.6.

Full PDF reports of findings from the Manual Penetration Testing is available here: https://drive.google.com/file/d/1ABSzJFUyjLXVljipszjqTnfClIn-e_Ci/view?usp=drive_link

Additional vulnerabilities were identified and logged in the automated scanning, with the full PDF reports of finding available here: https://drive.google.com/drive/folders/1VfeO4JZOqyEUm6CcWnG7qeSY6UPdWVyA?usp=sharing

High-level concerns, issues
Here is a summary of the top vulnerabilities discovered and their current status.

Vulnerability	Description	Proposed Fix	Status
None Hashing Algorithm Attack Critical	JWT library accepts none hashing algorithm. none hashing algorithm is used by the JWT in case the integrity of the token is already verified. So an attacker	Not allowing none hashing algorithm	Under review, and considering if a false positive or not; "If you manipulated the token to insert any claim, all you could do is access an empty client application as a different user type to see

	can alter the token claims and the token will be trusted by the application.		what UIs look like for different user types. You couldn't download or manipulate any data.
Insecure password change mechanism may lead to full account takeover High	A current password should be required to change the email address or password so that an attacker who can perform actions on behalf of a victim user (e.g., using XSS, CSRF) cannot take over the user's account by changing the e-mail address or password.	Validate old password during email change.	Reported to OpenCRVS
Cross Site Scripting (Reflected) High	Cross site scripting (XSS) is a common attack vector that injects malicious code into a vulnerable web application. XSS differs from other web attack vectors (e.g., SQL injections), in that it does not directly target the application itself. Instead, the users of the web application are the ones at risk.	Web application firewalls (WAFs) can be used to mitigate reflected XSS attacks. Also, deployment behind "Gov't VPN" can also mitigate impact	The script doesn't run in testing and an error JSON response is created. I think the objection here is that the app returns a 200 status code for errors. Re-run test in manual/vetted state to determine if true. Otherwise, ensure deployment guides have recommended info regarding WAF and VPN requirements
SQL Injection - SQLite High	SQL injection is a technique used to exploit user data through web page inputs by injecting SQL commands as statements	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side	Waiting for response from OpenCRVS team. It may be an issue in the "metabase" component, and not the core OpenCRVS
Content Security Policy	Some aspects of the defined CSP may be	The unsafe-inline Content Security	Tracking on ticket with OpenCRVS team:

(CSP) Moderate	exploitable, specifically “unsafe inline” https://content-security-policy.com/unsafe-inline/	Policy (CSP) keyword allows the execution of inline scripts or styles.	https://github.com/opencrvs/opencrvs-core/issues/5607 Working together to identity instances of unsafe CSP in their configuration
Server Leaks Version Information Moderate	Web servers often include the version information in the HTTP response header field Server. This information can be valuable to attackers	Remove the version information from the Server HTTP response header field. Configure the web server to not include version information in the Server HTTP response header field.	Tracking here: https://github.com/opencrvs/opencrvs-core/issues/5608
No Rate Limiting; Brute Force Attacks are Possible Moderate	Having no rate limiting can allow attackers to send a large number of requests to perform brute force attacks, disrupt normal service, or spam users.	Implement rate-limiting.	Reported to OpenCRVS
No Account Lockout Policy; Password can be brute forced on Login page Moderate	The application does not have a lockout policy on the login page to prevent user account compromise via brute force attacks. We were able to successfully send 100+ requests without getting blocked	implement an Account Lockout policy which will cause temporary lockout of a user account after 10 or 15 failed login attempts and after that, the user should be made to contact the Administrator to regain access	Reported to OpenCRVS

Username Enumeration Moderate	We found that the application displays sensitive information which allows attackers to guess valid registered email addresses of customers.	It is recommended to show a response that does not show variation leading to username enumeration when brute forced. For example, "If an account associated with the provided email address exists, an email containing instructions on how to reset the password would be sent."	Reported to OpenCRVS

Some of the issues discovered were determined to be irrelevant in the context of the OpenCRVS application. For example, the “None Hashing Algorithm Attack” listed as Critical above is a result of OpenCRVS’s split client/server architecture. In consultation with OpenCRVS we determined that no privileged data could be accessed by exploiting this avenue. It would merely allow a user to see the empty user interface of a higher-privileged user, something they could also do by downloading the source code from GitHub.

Summary of test environment setup, steps taken to complete analysis

The test setup was deployed by the OpenCRVS organisation at the public endpoint of https://guardian-project.opencrvs.org/

This is version 1.3 of the application, and should be considered a development or staging server. It has no actual data on it. It was also configured with a static 2FA code of 000000 to enable automated testing without requiring human intervention.

Summary of communication with vendor related to disclosures and direct feedback

After reviewing the potential vulnerabilities suggested by the Astra tool, we were able to rule out several as false positives, either because the technology that would enable a specific vulnerability was not present in the OpenCRVS stack or because sites not a part of the main OpenCRVS application were being tested. What remained were primarily issues related to specifics of the web server deployment. We met with OpenCRVS developers and came up with a set of vulnerabilities that we agreed needed to be addressed. They are:

- The TLS minimum version was too low, which presented potential security issues with encrypted web traffic. This problem was tracked in this GitHub issue: https://github.com/opencrvs/opencrvs-core/issues/5606 and was resolved with this commit: https://github.com/opencrvs/opencrvs-farajaland/pull/607
- The Nginx web server was reporting too much information about the specifics of its version and configuration. This was tracked in this GitHub issue: https://github.com/opencrvs/opencrvs-core/issues/5608 and resolved with this commit: https://github.com/opencrvs/opencrvs-core/pull/5625
- The Content Security Policy which determines how scripts may run inside the application while loaded in a browser was too permissive. This was tracked in this GitHub issue: https://github.com/opencrvs/opencrvs-core/issues/5607 and resolved in this commit: https://github.com/opencrvs/opencrvs-core/pull/5627

Status of any mitigations, patches, updated releases

As of 5 July 2023, a number of fixes and mitigations have been made by the OpenCRVS team:

- TLS minimum version is now >=1.2
- Nginx version doesn't leak from response headers anymore
- Content-Security-Policy is now stricter and doesn't include 'unsafe-inline'

We will continue to rescan and review fixes, and update the audit vulnerability report.

Outcomes

Evaluation of the holistic approach in terms of cyber security

Security policies analysis and Analysis of security guidelines/documentation

Other areas of this document have covered our perspective regarding the following of the variety of policies around deployment, configuration, and architecture security.

In the Application Architecture section, it was stated that: *"...the requirement for complex, coordinated testing, with capable and experienced operational expertise should not be ignored. Be mindful of compliance, regulations, policies regarding protection of sensitive data."*

In the DevSecOps review, it was stated:*"it is important to ensure that the service providers are compliant with applicable regulations, such as GDPR in the EU or HIPAA in the US, and that appropriate measures are in place to protect sensitive data. Implementers must also ensure that they are using the tools in accordance with their own internal policies and procedures to maintain data privacy and security. Failure to properly manage data privacy and security in the context of SaaS tools could lead to compliance violations, data breaches, or other legal and reputational risks. "*

We believe OpenCRVS does take security seriously, and has documentation, curriculum, and technology for implementing compliant deployments. However, as stated before, we do feel there are some gaps in their documentation for production deployment that need to be improved. In addition, continued work to understand the threat model of the places where this solution will be deployed is highly recommended.

Analysis of history of public vulnerabilities
***(and* CRVS product developer & maintainer response/management and communication to users)**

Being a relatively new codebase, there is not a large history of public vulnerabilities. What we do know however, is that OpenCRVS has engaged in multiple previous security audits, and that as with our own experience, they were very engaged and rapidly addressed critical issues. Their team, process, codebase, and entire organisation is a well-run open-source project that can benefit from having "many eyes" on the solution, addressing them in a unified way, and iterating through improved releases that can be deployed into an ecosystem of instances. This is an ideal way to address the "security is a process, not an endpoint" mindset.

Remaining Sections

- These two areas of interest were sufficiently covered in the other areas of our audit. We do have some initial notes below:
    - Analysis of the security of data at rest and in motion
        - In Motion: The penetration test did find issues regarding configuration and version of TLS, as well as some areas where TLS/SSL could be bypassed entirely. Again, this is primarily a deployment config issue, and not a fundamental problem in the codebase.

        - At Rest: Continued vigilance around threats related to unauthorised extraction of sensitive data through physical access to servers must be part of the threat model considered. Ensuring data encryption at rest is available as an option is important. See our recommendation about “Encrypted LVM” elsewhere in this document.

Feedback

- We will remain engaged with the OpenCRVS team as they continue to respond to and resolve issues found in the audit
    - This includes one final round of manual penetration testing once the primary high and critical issues are resolved.
- We will discuss the use of ongoing vulnerability scanning services that we can provide for free for the next 12 months as part of this service.
- We will discuss with the OpenCRVS teams what discovered vulnerabilities could be mitigated by improving deployment documentation, scripts, tools, etc.

DevSecOps Analysis

Summary of test environment setup, steps taken to complete analysis

OpenCRVS is designed to be deployed in a Tier 2 datacenter due to the unavailability of higher tier datacenters in the target regions. These data centres will provide rackspace, may provide access to transit providers or provide a managed uplink, and will have partially redundant power and cooling. Uptime will typically be better than 99.5%, however the application is designed for intermittent connectivity in mind and so interruptions to connectivity should not impact the operation of the system. Interruptions to power and cooling would be more likely in this environment than in higher tier datacenters and so the application should use a robust approach to data transactions that can maintain consistency through power events. These considerations are not unique to OpenCRVS and will be applicable to all the solutions assessed in this project.

In order to perform this analysis, the documentation was used to guide a deployment of the software in AWS but utilising AWS Lightsail virtual machines and making no use of the additional networking or storage features offered by the AWS platform. This approach was taken over a cloud-native service like EC2 due to the deployment environments that would be available in the target regions.

The deployment process uses Ansible, an industry-standard tool that employs declarative configuration files to describe the expected final state of a server, including all installed applications, patches and network setup. Ansible files (usually referred to as “playbooks”) may be version-controlled like any other code which makes them auditable and encourages iterative development.

The test deployment followed the documentation at:

- https://documentation.opencrvs.org/setup/3.-installation

and made use of the *opencrvs-core* and *opencrvs-farajaland* repositories, using the default Farajaland country configuration.

Outcomes

Software development operation best practices

There is a strong reliance on GitHub and CI tools throughout, which should be generalised to support options that may be required to host data and build processes in-country to comply with local regulations.

Review of operations management from a system administrator perspective

The production deployment documentation contains a number of details that may not be appropriate to a production deployment. The high-level issues relate to:

- Network security considerations
- Key and secret management
- Deployment maintainability

Network Security Considerations

The deployment guide starts from a point where you already have deployed 3 or 5 servers running Ubuntu LTS. This assessment assumes that these servers will be deployed in a co-location facility within the datacenter.

The choice of Ubuntu LTS is appropriate for this kind of deployment as security updates should be provided for 5 years since the release. It's important to remember, however, that Ubuntu LTS support is provided by Ubuntu community members and is not guaranteed without a commercial support contract from Canonical.

Ubuntu LTS will have good compatibility with server hardware from popular vendors such as HP, Dell, Supermicro, etc. It should be possible to acquire hardware in the target regions, and no specialist hardware would be required to run the software.

Before continuing with the deployment of software to the servers, first the network architecture must be considered. This aspect is not discussed in the OpenCRVS documentation. There will be a requirement for communication between the servers in the cluster and this communication should not be exposed to the Internet.

The following TCP ports were discovered open on the demonstration deployment of OpenCRVS (maintained by the OpenCRVS team):

Port	Description
22	OpenSSH Server
80	HTTP (redirect to HTTPS)
443	HTTPS (Application)
8200	HTTP (APM Server)
9000	HTTP (Redirects to HTTP MinIO console)
9001	HTTP - MinIO Console
9200	HTTP - ElasticSearch

Of these open ports, the HTTP and HTTPS ports (80 and 443) are expected to be open. It is not known if this is a single-server deployment, or a multi-server deployment, but notably the Docker Swarm port is not accessible via the Internet as had been previously identified by the Gofore report. There are however still the SSH port, and 4 plain HTTP listeners, for which there is no clear use case for access from the general Internet.

The Google Analytics cookie that is served by the main website is not restricted to HTTPS only pages and will be sent with any request to those 4 unsecured listeners. Other cookies may also be similarly unrestricted. No attempts were made to login to any of the services on the HTTP listeners.

Key and Secret Management

Deployment and management of the servers running the application happens via Secure Shell (SSH). While there is advice to use key authentication for this, the documentation assumes that the operations will be performed using the same SSH keys as may be used with a GitHub user. There is no discussion about the use of hardware tokens for SSH key storage in the documentation.

Using hardware tokens such as YubiKeys for SSH keys is a highly secure approach to access management. Without the use of a hardware token, the keys may be freely accessed, duplicated and potentially extracted from the environment. Being uncertain about the locations in which the key exists and is usable makes it difficult to reason about threats posed by the key.

The documentation only refers to adding the key to the server and does not refer to the users or groups that should be used for privilege separation. The Ansible configuration assumes that the SSH user would be root, however it would not be advised to allow SSH access by the root user directly. The Ansible configuration also uses the become module for privilege escalation from an unprivileged user, which is the arrangement used for the assessment. That user has passwordless sudo access in order to allow use of the become module. It is also required that the master node must be able to SSH into each of the other nodes, including the backup node.

Recommendation: There is no guidance offered for how to manage key security credentials (like an SSH key). We would recommend that this should be done using a hardware token.

Encrypted storage on the server is only used for the data volumes, and is provided via cryptfs, a built-in disk encryption system in the Linux kernel. During the installation attempt, cryptfs was not used despite it being enabled. There may be an error in the playbook where as the cryptfs file does not exist, the `st` variable registers that, and after the creation of the file nothing will update that variable. Future tests of `st.stat.exists and encrypt_data` will still resolve to false.

A swap file is created and enabled, however the swap file is not encrypted. This means that data stored in memory at runtime may still be committed unencrypted to the disk. This pattern reflects one that may be used during development using virtual machines with fixed disk layouts.

Recommendation: We would recommend that the system be deployed without these specific post-installation modifications, using encrypted LVM from the start.

Encrypted LVM is supported by Ubuntu 20.04 LTS as part of the standard installation. On platforms with a Trusted Platform Module (TPM), this can hold the key to decrypt the disk on boot and various checks can be used to automatically destroy the TPM's copy of the key on suspected compromise (e.g. chassis entry sensor, or extended power interruption).

Deployment maintainability

The Ansible playbooks are written as single playbooks with no external roles or plugins. It would be advisable to break down the playbook into roles, even if these are maintained within the same repository, to provide individually testable and documentable units. It is easier to review a number of smaller components with well defined interfaces than it is to review a large monolithic component with greater internal complexity.

There are issues with idempotency in the playbooks. A key strength of Ansible is the ability to use playbooks to monitor and correct drift in the system configurations over time. If changes are made directly on the production server then a future run of the playbook would be able to detect and correct those changes. It would appear that these playbooks are intended to be run only once at the start to the deployment, but this does not take full advantage of Ansible as a powerful tool for managing software deployments.

An example of this is in the deployment playbook within the opencrvs-farajaland repository: `playbook-3.yml`. The task *Create MongoDB replicate key file locally*[1] will run regardless of the file already existing. Multiple runs of the playbook will discard the created key rather than reusing the existing key. Further, the keyfile created by this task stores the result in the storage of the machine running Ansible and does not otherwise persist it.

The documentation does not explicitly mention the deviation from the usual way in which Ansible is used and so this could create confusion for system administrators that expect to be able to run the playbook multiple times without side effects.

Using multiple playbooks for the different deployment scenarios has already led to divergence in the repository between the playbooks. The only differentiating factor should be the number of worker nodes that are deployed but our analysis has found that other aspects have changed over time where updates have not been uniformly applied across the three variants of the playbook. Different software package repositories are used to install Docker between the playbooks, a file mode is missing from the replication key installed, and a comment regarding MOSIP integration is missing from one file. These differences evidence a need to consolidate the playbooks and reduce maintenance burden as minor issues have crept in.

[1]https://github.com/opencrvs/opencrvs-farajaland/blob/702c3596b847ada66f000ed534ae8e2bda423514/infrastructure/server-setup/playbook-3.yml#L13

Recommendation: We would recommend that these playbooks be merged together and the true and intentional differences are extracted as variables or controlled with conditional statements.

The documentation recommends Gitflow (isolating your work into different types of git branches) for development and expects that country-specific configurations will regularly rebase on the upstream Farjaland country configuration. This will involve rebasing multiple branches and these operations are not familiar for those that do not work with Git regularly. Either additional documentation is required here, or a script to perform the automations automatically and remove the risk of human error.

The documentation contains a note[2] regarding the import of CSV files that is concerning. It requires that the CSV file imported contains no empty lines, and that there are no commas within data fields. This suggests that a naive implementation of a CSV reader is being used that may cause data integrity errors on import, and being a naive implementation may not raise warnings about those errors. There exist in almost every programming language robust parsers for CSV files that will correctly handle the escaped and quoted strings produced by common CSV writers, e.g. Microsoft Excel.

While a "Humdata" standard is referred to, a search did not turn up any documentation on the standard and so implementers are left to reverse engineer the specification of the file from a number of examples.

Translated strings can be used and standard tools like Transifex can export to the JSON format that is used.

Guidance on production deployment

Recommendation: We recommend that at least a stateful firewall is used to protect the IP subnet on which the cluster is deployed and that that subnet should be isolated at layer 2, either by VLAN or physical layer separation.

[2] https://documentation.opencrvs.org/setup/3.-installation/3.2-set-up-your-own-country-configuration/3.2.3-set-up-cr-offices-and-health-facilities/3.2.3.1-prepare-source-file-for-crvs-office-facilities (Archive: https://archive.is/MF5Yy)

For internal communication inside the cluster, a separate cluster subnet should be utilised if possible, without the ability to route to and from the global Internet from that subnet. In addition, the firewall may provide VPN access to authenticated users to access the servers by SSH and to protect the communication between the server cluster and the backup server.

Security appliances, such as the Juniper SRX345, may be deployed in pairs to provide fault tolerance. These appliances would combine both the stateful firewall and VPN in a single 1U form factor. If deployed in a datacenter where other services are already deployed, these features may be provided by an existing security appliance.

It is suggested that secrets are maintained in a password manager and passed to the Ansible playbook via the command line at runtime. This approach would not be advisable as it will result in the secrets being stored in the command history of the shell, which is freely accessible to any application running on the machine. There also has not been any discussion around the security of the machine that is running Ansible so it's not clear if this machine would have encrypted storage.

Recommendation: Mozilla's sops may be used to encrypt the relevant secrets and store these alongside the rest of the IaC.

An Ansible vars plugin (community.sops.sops_vars) is available to load the variables and decrypt them at the time that the playbook is run. This would also reduce possibilities of human error when manually copying and pasting secrets from the password manager.

There is an acknowledgement in the documentation that the current approach would not be production-ready, which is good to see. There is a link to the MOSIP documentation relating to the requirements for a Hardware Security Module (HSM) used to securely store the secrets, however this HSM would be greatly over-specified if all it would be used for is providing access to runtime secrets. If the goal is to use an HSM to produce signatures for certain records as part of registration workflows it would be more appropriate, but in that case we would still recommend using a separate HSM for the management of runtime secrets.

Setting up a development environment is required as part of the production deployment. This is where the country configuration will be built. The development environment expects that you will be running Google Chrome on the same machine in order to access it. Given how common it is for vulnerabilities to occur in browsers, it's not advisable to allow the browser to be running with the same user account on the same system as will be building the country configuration. At the very least, a different user account should be used to run Google Chrome, or the necessary ports required may be forwarded over SSH using an otherwise unprivileged user to another machine where the browser is used.

In the section on setting up the country configuration, the documentation recommends 15 extensions to Visual Studio Code. This includes 3rd party extensions that have not necessarily been audited for security vulnerabilities. They are installed from an open marketplace with a low barrier for entry to submit extensions. The extensions are privileged in that they get to modify, and can conceal the modification of, the country configuration code. They are perfectly placed to conduct a supply chain attack. In addition, there is a risk of dependency confusion as the extensions are referenced by name only and not to a particular package or to a package by a particular author.

The documentation also recommends ohmyzsh, a popular framework for customising the zsh shell. While customising the shell can be fun, this kind of framework is not necessary and can be harmful in a production environment due to the possibility of deliberate or accidental bugs that may affect the correct operation of the system.

Recommendation: 3rd party customizations should be limited in the production environment, and in systems that have the ability to pass on code and configuration unaudited to the production environment, to only those that are strictly necessary to either perform the configuration and deployment or to mitigate or remediate other identified risks.

Final Report and Recommendations

Overall Findings

🟢 Positive.

Overall we find the OpenCRVS product to be stable and ready for implementation. It has great documentation for implementers, is interoperable with many e-government solutions, focuses on real-world workflows and a team ready to help. The reliance on third-party dependencies are commonplace and the OpenCRVS choices of microservices makes for a flexible and extensible solution.

We found a few areas for improvement, which are outlined further down, but they do not detract from the overall positive evaluation of OpenCRVS. OpenCRVS is new to the CRVS market (debuted in 2019). It is only a "version 1" generation release, and may not be as flexible or feature rich as other potential solutions. It does benefit from a laser focus on the eCRVS functionality, modern and clean architecture, and an extremely organised and well-run public open-source project. When thinking about how this might be deployed and maintained throughout the world, the idea that a single unified upstream codebase could be improved and updated, and then pushed out to downstream instances, is very attractive and beneficial.

To conduct the OpenCRVS evaluation we read publicly available documentation, visited their website and supporting resources (Github, documentation site, etc), we deployed our own instance and tested for vulnerabilities and dependencies, created a SBOM for the codebase, audited the architecture, deployed the software in AWS and ran multiple variations of vulnerability scans and penetration tests.

Throughout our evaluation of the OpenCRVS solution we communicated any critical vulnerabilities or discovered security risks to their team. Once we completed our evaluation we allowed time for the OpenCRVS team to review our findings and make comments. A number of key issues identified have already been addressed by and even resolved by the OpenCRVS team during this process. While others remain known and discussed.

Since the OpenCRVS Evaluation Report will be public, all publicly disclosed issues remain in the published version of our report and anything we feel are sensitive or detrimental to the security of the solution or its users have moved to a separate Annex which will remain private.

Area of Evaluation	**Readiness**	**Impact**	**Comments**
Evaluation Aspect	General readiness / fitness of solution in specific area	Affect that readiness has on viability of solution as part of this evaluation	Any summary thoughts on each area
Source Code Security	Positive/ Ready	OpenCRVS is a well-built open-source project, built using best practices, that utilises microservices, common backend applications & industry-standard frontend. Security concerns are handled in a timely, serious and transparent manner	Some open issues remain from source code audit, but we have confidence they will be addressed thanks to engaged and open nature of the team

Application Architecture	Positive/ Ready	Overall architecture is sound from both an application developer's and operational engineer's perspective. The chosen open source technologies are reliable and proven projects that allow OpenCRVS to develop and deliver a solid CRVS system with a small development team.	The requirement for complex, coordinated testing, with capable and experienced operational expertise should not be ignored. Be mindful of compliance, regulations, policies regarding protection of sensitive data. * Update Assessment of new v1.3 release changes advised
Penetration Testing	Positive/ Pending resolution of critical and high issues	While critical and high issues were found, these are largely in deployment configuration, and not faults in the core application. The OpenCRVS team is very responsive, and work on resolving and retesting is underway.	Ongoing scanning and vigilance is necessary to ensure a secure solution * Manual Pen Test to be scheduled for final review
DevSecOps	Positive / Needs attention	Documentation about network architecture, Ubuntu LTS deployment, key management (hardware tokens), needs to be updated and/or included. Strong reliance on Github and CI tools which should be more general to support in-country processes and regulations.	Given OpenCRVS is an open project/platform and not a "vendor", the ability for independent deployment without error or misconfiguration is a key area to improve

Actionable Recommendations

- **General Recommendations**
  - In stage 2 of this project stream, it would be useful to research and create some persona-type modelling of potential government/ organisation deployment capabilities and environments.
    - The persona archetype could be useful across many other digital interventions in which the system is ultimately going to be run by the government. Consider which regions and countries to focus on.

    - Understand if training/upgrade of skills to move towards valuable services like Docker, etc, is something that should be invested in.
    - Discuss and develop an accurate threat model/stance that aligns with the vision of OpenCRVS and expectations of UNICEF.

- **Documentation**
    - Add documentation to the application architecture about the external operational services: LogRocket, Sentry, and PaperTrail.
        - Why are these dependencies used? How do they affect TCO?
        - In the case of PaperTrail, since it is not required by the source code itself, mention possible alternatives.
    - Add documentation to the high-level architecture that an external SMS delivery service provider will be required.
    - Highlight at the high-level the role Contentful plays within the architecture and how it is a defacto dependency.
        - Optionally, include a working example of how to replace Contentful with an alternative, perhaps on-premise, solution.
    - Provide a source code skeleton or starting point with examples, that an implementer could use if they need to customise or extend aspects of the system beyond the basic country configuration. Ideally an example test suite would be included too.

- **Source Code Security**
    - Not a turn-key solution and requires operational expertise to manage and maintain effectively, which is common for many software applications in this functional space.
        - Docker-compose files reduce the burden of setup and maintenance.
    - From our perspective, they could do more with active/ongoing vulnerability scanning, which is why we have introduced the idea of services like Intruder or Astra, along with source code level dependency monitoring as part of their CI process.
    - In the upcoming version 1.3, OpenCRVS will add an additional external dependency of Metabase to facilitate the functional feature of graphs and dashboards for vital statistics. The security, privacy and operational implications of this new dependency should be explored.

        - https://github.com/opencrvs/opencrvs-core/issues/4679
        - https://www.metabase.com/

- **Application Architecture**

  - *We advise* conducting a followup assessment of the Application Architecture Audit against OpenCRVS version 1.3 which is now available, but wasn't ready at the time of our Audit.
    - Specifically focusing on the additional external dependency of Metabase to facilitate the functional feature of graphs and dashboards for vital statistics. The security, privacy and operational implications of this new dependency should be explored.
  - No licensing fees, but consider all of the additional elements necessary for 'Total Cost of Ownership' https://www.opencrvs.org/uploads/images/Understanding-the-costs-of-an-OpenCRVS-Implementation.pdf.
  - The microservices architecture is flexible and scalable, and the use of third-party software packages and services provides a solid foundation for the system. However, the distributed nature of the architecture and the use of many third-party dependencies mean that the system is complex and will require specialised expertise to manage and maintain effectively.

- **Penetration Testing**
  - There are still a number of high and medium issues to resolve, and supplemental rounds of penetration testing to perform to verify full resolution of open issues.

- **DevSecOps**
  - It would be advisable to break down the Ansible playbook into roles, even if these are just maintained within the same repository, to provide individual testable and documentable units. There are also issues with idempotency in the playbooks.
  - **Recommendation:** There is no guidance offered for how to manage this SSH key, although we would recommend that this is also using a hardware token.
  - **Recommendation:** We would recommend that the system be deployed without these specific post-installation modifications for encrypted storage, using encrypted LVM from the start.
  - **Recommendation**: We would recommend that the deployment playbooks be merged together and the true and intentional differences are extracted as variables or controlled with conditional statements.
  - **Recommendation on production deployment:** We recommend that at least a stateful firewall is used to protect the IP subnet on which the cluster is deployed and that that subnet should be isolated at layer 2, either by VLAN or physical layer separation.
  - **Recommendation:** Mozilla's sops may be used to encrypt the relevant secrets and store these alongside the rest of the IaC.

- **Recommendation:** 3rd party customizations should be limited in the production environment, and in systems that have the ability to pass on code and configuration unaudited to the production environment, to only those that are strictly necessary to either perform the configuration and deployment or to mitigate or remediate other identified risks.

Closing

Our team has many years of experience working with, testing, building on, and deploying open-source projects and platforms. Behind these efforts are a wide variety of people and teams who vary in terms of scale, experience, quality, reliability, vision, and commitment. Finding the needed blend and balance of creativity and experience, innovation and reliability, to achieve a usable, dependable solution, is always difficult. You might have someone who can quickly prove something is possible with a clever bit of code, but then not have the follow through to implement all the needed features. Alternatively, you may have someone who is meticulous and detailed, but not flexible enough to quickly update, iterate, and improve their work as their growing communities demand. In addition, there are the many decisions of how to build a solution - do you write everything from scratch or first principles, because you believe you must or are have some kind of scratch itch, or do you commit yourself to depending upon, and supporting, a rich ecosystem of open-source modules, libraries, and services? There isn't really an answer, only a path forward, and the requirements and expectations that come with it.

With OpenCRVS, we strongly feel they have navigated these many complexities deftly and wisely, to create a solid “1st generation” open-source eCRVS solution, backed by a robust, committed team. It is truly an open-source solution, developed transparently, with open documentation, issue tracking, bug reporting, auditing, and more. This isn't a project built in a proprietary manner, whose code was then dumped as a zip file into a github repo. OpenCRVS understands what it means to be open, and embraces that. This also includes focusing on what they know how to address at the core, both from CRVS requirements perspective and target community usability needs, and utilise existing third-party code as often as they can. We approve of this approach, and it has allowed their solution to come a long way very quickly.

The downside of both being a relatively new solution and depending upon an ecosystem of dependencies, are the possible vulnerabilities that you expose yourself to. Throughout this process we have discovered a number of them, be it within the source code itself, or the version of nodeJS it is built upon, or regarding the security of how an instance is deployed. This was not unexpected or a surprise to find the sort of issues we did, at this stage in the project. What was a surprise was how engaged and positive the OpenCRVS team was at reviewing and addressing the issues. Fixes were implemented, nodeJS was updated, deployment configurations were improved. These are very good signs of a healthy project, who is ready to face the challenges of the real world.

All that being said, we do have strong concerns about improving the documentation that truly will allow this project to be "self-service" deployed. If we do want OpenCRVS to be a key solution for scaling up eCRVS around the world, we all need to ensure it can be deployed in repeatable, reliable, private and secure manner, by anyone who is qualified to do so. The commitment to open, quality documentation is there, but there needs to be some confusing aspects ironed out, some additional pieces provided, and some improved scripts and deployment tools provided.

It has been a pleasure reviewing the OpenCRVS solution.

Appendix

- [ ] Website: https://www.opencrvs.org/
- [ ] Documentation link: https://documentation.opencrvs.org/
- [ ] Case Studies: https://www.opencrvs.org/about-us/case-studies
- [ ] March 14, 2023 - Live Product Demo over Zoom (Raw Notes here: https://docs.google.com/document/d/17CWjDPqyJZ6s4IBHlLxXgCRratHvSgR-sKimZ9UedXE/edit?usp=sharing)
- [ ] Github Issue list for core OpenCRVS codebase https://github.com/opencrvs/opencrvs-core/issues
- [ ] PDF reports of finding from Vulnerability Scans and Automated Pen Testing area available here: https://drive.google.com/drive/folders/1VfeO4JZOqyEUm6CcWnG7qeSY6UPdWVyA?usp=sharing
- [ ] PDF report of the Manual Penetration test is available here: https://drive.google.com/file/d/1HwYzlczp9QAiGP2ySXqfeZTuBsLsVV3i/view?usp=sharing
- [ ] Software Bill of Materials (SBOMs) archive format
    - [ ] SBOMs in JSON format are available here: https://drive.google.com/drive/folders/1gmNYA2DeaKXzV_f7IDmdJBF1bw_n2bTH

Annex– Issue Resolutions & Mitigations

Since our assessment, OpenCRVS has been busy addressing the issues found while integrating feedback and making improvements to their overall code. Numerous technical enhancements have been made to the platform, and they are now about to release OpenCRVS v1.5.0

(https://documentation.opencrvs.org/general/releases/v1.5.0-release-notes). The following issues, recommendations and resolutions are based on communication with the OpenCRVS core team, our report findings and their updates with release candidate OpenCRVS v1.5.0.

Manual Penetration Test Notes

In the Manual Penetration test report, 1 of the Critical and many of the High and Medium findings were in fact related to:

- https://community.opencrvs.org (A forum website powered by Discourse)
- https://documentation.opencrvs.org (A docs website powered by Gitbook)

None of these issues have anything to do with OpenCRVS Core Civil Registration application.

Found Issues, Recommendations & Resolutions

Web Application Recommendations		
OpenCRVS v1.3.0 assessment	**OpenCRVS v1.5.0 upcoming release**	**Notes**
Critical: JWT None hashing Algortihm	Resolved	Crypto replaced by bcrypt Pull request: https://github.com/opencrvs/opencrvs-core/pull/5024 **Issue:** https://github.com/opencrvs/opencrvs-core/issues/4979
Critical: Mongo injection	Not Applicable	The finding was on community.opencrvs.org (a Discourse community forum) which has nothing to do with the OpenCRVS Core civil registration application.
High: SQL Injection	Not Applicable	The finding was on community.opencrvs.org (a

Web Application Recommendations		
OpenCRVS v1.3.0 assessment	**OpenCRVS v1.5.0 upcoming release**	**Notes**
		Discourse community forum) which has nothing to do with the OpenCRVS Core civil registration application.
High: Backup file exposure	Not Applicable	The test failure refers to their documentation website, powered by Gitbook and has no relation to the OpenCRVS Core civil registration application.
High: Insecure password change mechanism	Not Applicable	Suggestion: A current password should be required to change the email address or password. Response: The current password has always been required to change the password. Email changes do not take effect without a 2FA code. *Therefore a possible mistake in reporting.*
High: CSP	Not Applicable	A Content Security Policy header is in place via Nginx. We posit that this is present in the documented responses from OpenCRVS Core and implemented correctly in the core application. The test failures point out that it is not implemented in documentation, as that is powered by Gitbook and has no relation to the OpenCRVS Core civil registration application.

Web Application Recommendations		
OpenCRVS v1.3.0 assessment	**OpenCRVS v1.5.0 upcoming release**	**Notes**
Medium: Issues with Gitbook	Not Applicable	Many medium findings are related to documentation.opencrvs.org (a Gitbook documentation site) which has nothing to do with the OpenCRVS Core civil registration application.
Medium: Rate limiting	Resolved	Introduced in OpenCRVS v1.5.0: https://github.com/opencrvs/opencrvs-core/issues/5930
Medium: Lockout policy	Resolved	Resolved by rate limiting. A lockout policy would not be an effective solution in low resource settings as it would be an operational management overhead.
Medium: Username enumeration	Not Applicable	The failure describes "If an account associated with the provided email address exists, an email containing instructions on how to reset the password would be sent." This is true, but no other approach for forgotten usernames can be proposed without operational overhead in low resource settings. The email account of the government staff member must be secured with a strong password and 2FA as advised in our Data Security Framework

Infrastructure Recommendations		
OpenCRVS v1.3.0 assessment	**OpenCRVS v1.5.0 upcoming release**	**Notes**
Port 8200, 9000, 9001 & 9200 open	Resolved	These ports were open in v1.3.0 for debugging but protected behind strong passwords. No data was accessible behind them. ***They have since been closed. The Minio console is removed.***
Port 22	Resolved	The SSH_PORT is now configurable in **OpenCRVS v1.5.0** via Github Secrets depending on country requirements to any port the country chooses to use.
SSH Keys	Resolved	Ansible user steps now lock down root access for all SSH users. A "provision" SSH user's id_rsa is now secure in Github Secrets. All SSH users require Google Authenticator when connecting and every SSH access fires an alert to a shared channel / email address.
Ansible Playbooks & Secrets	Resolved	Ansible playbooks are now broken down into easily testable plugins. All playbooks can be repeatedly run together or individually. **Ansible is now no longer run from the command line, but run via provision pipelines in Github Actions.** All secrets are now supplied by Github Secrets using the Ansible vars plugin as advised. So

Infrastructure Recommendations		
OpenCRVS v1.3.0 assessment	**OpenCRVS v1.5.0 upcoming release**	**Notes**
		secrets are no longer potentially stored in the command history of the shell. Using Github Secrets mitigates any need for a Hardware Security Module. Although an HSM or YubiKey would still be a recommendation for LVM encryption keys.
Vulnerability Scanning	Resolved	All containers are now scanned by Trivy.
TLS Version Detect	Resolved	An upgrade to Traefik resolved this: https://github.com/opencrvs/opencrvs-farajaland/pull/607
Node Version	Resolved	Node is upgraded to v18 in OpenCRVS v1.5.0.