



DHIS2 EVALUATION REPORT

APRIL 2024

Guardian Project conducted an independent evaluation of DHIS2, Version 40.0 and Android v2.8/ May 2023
This task was commissioned by the Digital Center of Excellence (DCoE) at UNICEF.

PREPARED BY
Guardian Project



Statement

July 2024

This assessment was completed and reviewed with the DHIS2 team in April and May 2024. Since then, the DHIS2 team has been working on implementing the recommendations from this assessment along with community feedback. Key security issues have been addressed and we await a formal update from the DHIS2 team.



Evaluation Report

Performed by Guardian Project

Final Report, April 2024

Solution Name and Version: DHIS2 v40.0 and Android v2.8/ May 2023

The evaluation of the DHIS2 solution by Guardian Project in 2024 is now a publicly available document under [CC-BY copyright](#).

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.

Introduction

This is the Guardian Project's latest iteration of the solution provider evaluation report for DHIS2. This work is being done by the request of the Digital Center of Excellence (DCoE) at UNICEF.

Table of Contents

Glossary of Terms	3
Evaluation Summary	4
Executive Summary	4
Summary of the DHIS2 Solution	6
Elements of the Solution	9
Some current capabilities of the solution	11

Threat and Risk Assessment	12
Initial Assessment Results	14
Initial Thoughts	14
General Impressions	14
Key Team Members and Roles	15
Summary of Initial Interactions	15
Concerns and Blockers	15
Questions to investigate further	15
Review of Assets	16
Source Code Security Audit	18
Processes and Tools	18
Research and document the complete “Software Bill of Materials” (SBOM)	18
Open-Source Software (OSS) vulnerability scanning	19
Static application security testing (SAST) scanning	19
High-level overview	19
Summary of test environment setup, steps taken to complete analysis	20
Summary of communication with vendor related to disclosures and direct feedback	20
Status of any mitigations, patches, updated releases	20
Outcomes	20
Application Architecture Audit	31
High-level overview	31
Summary of test environment setup, steps taken to complete analysis	32
Summary of communication with vendor related to disclosures and direct feedback	32
Outcomes	32
Overview of the structure of the application	32
Evaluation of Configurability	33
Evaluation of Extensibility	35
Evaluation of maintainability and performance at scale	39
Penetration Testing Audit	39
Summary of test environment setup	39
Details of process, setup, tools utilised	39
Outcomes	40
Feedback	49
Summary of communication with vendor related to disclosures and direct feedback	49
Status of any mitigations, patches, updated releases	49
DevSecOps Analysis	50
Summary of test environment setup, steps taken to complete analysis	50
Test Environment Deployment Overview	51
Outcomes	51

Operation best practices	51
Operations Management Review	53
Production Deployment Guidance	55
Secrets Management	56
Final Report and Recommendations	56
Overall Findings	56
Actionable Recommendations	59
Closing	61
Appendix	62

Glossary of Terms

Listing of combined terminology from both general CRVS definition space, along with vendor and audit/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 licenses 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 license 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.
- **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).
- **Health Management Information System (HMIS)**: An information system consisting of computer hardware and software, procedures, and processes that are specifically designed and implemented to store, maintain, collect, process, represent, manage, and

transmit a patient's electronic medical record (EMR) and additional information specific to the health care domain.

- Electronic Immunization Registry (EIR): digital tool that helps collect, store and digital track an individual's immunisation history.
- Application Programming Interface (API): is a way for two or more computer programs to communicate with each other. It simplifies software development by enabling applications to exchange data and functionality easily and more securely by providing a set of definitions and protocols to build and integrate application software.
- Content-Security-Policy (CSP) headers: the name of a HTTP response header that modern browsers use to enhance the security of the document (or web page). The Content-Security-Policy header allows you to restrict which resources (such as JavaScript, CSS, Images, etc.) can be loaded, and the URLs that they can be loaded from.

Evaluation Summary

Executive Summary

The following evaluation of DHIS2 was conducted by Guardian Project over the course of Spring, Fall and Winter 2023. The evaluation was designed to provide a comprehensive review of the DHIS2 solution through the following couses:

- 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 DHIS2 solution ([District Health Information Software](#)), to be stable and ready to implement.

It is a free, open-source, generic software platform for reporting, collecting, analysing and dissemination of individual-level and aggregated data. Although, keep in mind, the DHIS2 is not a purpose-built CRVS solution, but rather a software that can integrate with CRVS systems or potentially be extended to include CRVS-related features. Interoperability tends to be expensive to implement, both in terms of initial integration work and long term operation and maintenance. At this stage in the evaluation we will continue to consider what additional costs are associated with combining DHIS2 with a CRVS system.

Provided as a global public good (DPG) to help support sustainable development goals, DHIS2 comes with a suite of visualisation and analysis tools which makes it capable at linking data for reporting and tracking purposes. It can be hosted locally on a device (mobile, laptop, desktop or local server) or in the cloud. It is most commonly used for health data, but integrates and is interoperable with a number of various softwares and tools using the DHIS2 Application Programming Interface (API). It is currently deployed in 75 low- and middle-income countries and over 100 NGOs have implemented the software. Often deployed by a country's Ministries of Health (government department), DHIS2 fits many contexts and use cases, such as logistics, education, COVID-19 vaccine tracking, agriculture projects, e-governance, food and nutrition and more.

To conduct the DHIS2 evaluation we read publicly available documentation, visited their website, attended virtual conferences and leveraged available resources on YouTube and their community resources page. We also started a local cluster for testing using the DHIS2 command line tool (<https://cli.dhis2.nu/#/getting-started>) and used that tool to create a test application to extend the core functionality. We decided to deploy our instance and tested vulnerabilities and dependencies, created a SBOM for the codebase, audited the architecture deployed software in AWS and ran multiple variations of vulnerability scans and penetration tests. We had some ongoing conversations with the DHIS2 team, but found it to be a busy time of year for them, so scheduling additional calls or walkthroughs proved to be difficult. Nevertheless, the team has been responsive on Slack and directly within this evaluation document to provide feedback and clarity. They are an engaged and community-driven organisation.

Summary of the DHIS2 Solution

Originally developed in 1994, for three health districts in Cape Town, South Africa, as a collaborative research project, DHIS2 has become the world's largest health management information system (HMIS). The project is supported by donors who provide stable and regular funding enabling DHIS2 to explore new technologies, develop networks of experts and expand while providing a sustainable product. DHIS2 is free to download, with no licensing fees, and is adaptable to meet local demands and new challenges.

Every individual instance of DHIS2 software and data contents are locally managed and owned. The system is designed to aid the capture of data linked to any level in an organisational hierarchy at any data collection frequency, including event data (patient satisfaction surveys, disease outbreaks) aggregate data (population estimates, routine health facility data, staffing, infrastructure), and individual-level longitudinal data (patient treatment and follow-up, vaccination records, lab sample testing). DHIS2 supports collaboration, sharing of data, commenting and interpretation. Custom applications, third-party software and external data sources can extend the reach of data collection and functionality. Localization and translation support are available for the web-based platform in French, Spanish, Hindi, Chinese, Portuguese, Vietnamese and Norwegian.

DHIS2 is developed and maintained by the [HISP Centre at the University of Oslo \(UiO\)](#), Norway, and is supported by a coalition of software providers and international investors. DHIS2 is a global solution with a global network of local and regional partners, experts and staff. Their core team is organised by function. The core team includes research, training and communication, implementation, development and project support. They collaborate across teams to ensure their work addresses real-world needs. The DHIS2 open-source project also relies on their community network of contributors to help ensure they are addressing real-world use cases and fitting user needs. Contributors can actively contribute source code, custom web applications and software suggestions. Resources for developers to contribute can be found in their developer portal on the DHIS2 website. Because DHIS2 can fit various models of use, documentation around how to implement, use, manage and make it most effective for you are extensive. With digital and in-person support.

DHIS2 is a generic platform which makes it the right solution for many. The team and its partners have also created customised apps, metadata packages and training support for various use cases, which help ease the burden of implementation and adoption. Most recently DHIS2 was deployed to assist in tracking COVID-19 outbreaks and vaccinations. Since many communities were already utilising DHIS2, integrating the software with COVID-19 specific tools was easy, creating rapid roll out on a global scale. Various metadata packages exist for the DHIS2 system which allow countries to use installable templates of DHIS2 rather than starting from scratch. The metadata packages also allow for implementers to adapt and customise their DHIS2 solution to fit the needs of their community and its contexts. The use of metadata for organising and classifying data makes DHIS2 different from many other traditional health information systems.

For more information on recent COVID-19 Surveillance use cases and other examples of how DHIS2 is utilised visit <https://dhis2.org/in-action/>.

DHIS version 1 was publicly released in 1996. This evaluation reviews the latest stable version of DHIS2 which is v40.0 and Android v2.8, recently released in May 2023. A new backend version of DHIS2 is released every 6 months with new features and fixes requested from their global community of users. Continuous app updates and patch releases keeps DHIS2 relevant and working. All of the source code, releases, and documentation are available via their website and Github. DHIS2 is open-source software, shared under a [BSD 3-Clause licence](#). BSD 3-Clause stands for Berkeley Software Distribution, offering flexibility of use and few restrictions. The licence requires inclusion of the BSD copyright when publishing modified versions of the source code. All DHIS2 source code is hosted on [Github](#) and you can find it at github.com/dhis2. The server-side code is found in the [dhis2-core](#) repository.

The DHIS2 system works as a centralised data warehouse capable of capturing data from a number of different sources and devices such as an Android app, SMS, direct web entry and other data sources like spreadsheets or other electronic systems. Broadly the data enters the system organised into three data models 1) numerical or aggregate data, 2) anonymous individualised data or events, and 3) individual data with identifiers or tracked over time data. There is a specific capture and input method for each model of data collected. The forms and hierarchy are completely customizable based on the user's needs, however, this separated data entry approach has the ability to help with accuracy and reliability of data entry.

In 2018, the DHIS2 software team developed the Android Capture App. The DHIS2 Android App is open source and available for download on the Google Play Store and Github. The code is available on Github, and the development roadmap and list of known bugs can be accessed on Jira. The DHIS2 Android Capture App is not a stand-alone application, but a mobile extension of the DHIS2 core platform. As such, it can be seamlessly integrated into existing DHIS2 systems, and works natively with aggregate, Tracker, and Event programs. The data syncs automatically with the user's central DHIS2 database, allowing for a smooth flow of data from the local level into a district- or national-level system.

- Android App development: <https://dhis2.org/android/development/>
- Android SDK: The [Android Software Development Kit \(SDK\)](#)
- Feedback on app release is encouraged via the [DHIS2 Community of Practice](#) page

DHIS2 has also produced an SDK that makes it easier for developers to create custom and ad-hoc apps with seamless integration to the DHIS2 platform. Additionally, the DHIS2 platform supports SMS-based solutions, which elevates capture capabilities in offline settings. It has been used for HIV reporting, coordination, tracking mother and child health, education and more. The DHIS2 website provides detailed videos, images and documentation on implementation guidelines, use cases and configurations.

Some key features of the app include:

- Real-time data entry by any DHIS2 system user, from frontline workers to system managers
- Full offline functionality with intelligent sync
- Native integration with DHIS2 aggregate, Tracker, and Events programs
- Full Tracker Capture support, including search function for existing records
- Support for the DHIS2 program rules engine, which supports guided data entry and dynamic form updates for complex workflows
- User-friendly navigation, including options for icon-based data entry
- Automatic visual alignment with web-based DHIS2 programs and forms, plus options for custom configuration
- Secure and simple login, QR scan for easy server URL configuration
- Map view and easy GPS location capture for data entry
- Create and scan barcodes and QR codes for easy mobile data entry and record management

The DHIS2 website hosts a plethora of information, documentation on how to implement and integrate DHIS2 with your own workflow, training academies for further learning and connections, stories of impact, and use cases. They have a [YouTube channel](#) with demos and training videos on various topics. Videos and details regarding specific topics or new releases are regularly provided. Overall, the information is organised well, the website is user-friendly and it is clear they've achieved much success with their tool. The team seems engaged in our evaluation process and willing to help when able. Nonetheless, they are clear where their boundary lies as a product developer and sustainer vs. an integrator of the system.

Elements of the Solution

The DHIS2 solution is a generic platform which is compatible with other systems and has been used in many contexts. On the DHIS2 website there's a [number of impact stories](#) sharing about how governments, NGOs and communities have implemented DHIS2. However, there is only one use case (that was found publicly published) which explains the interoperability of DHIS2 with a CRVS system. Since the main objective of this evaluation is to understand CRVS solutions, here are some highlights from the integration of CRVS with DHIS2 in Rwanda.

Use Case: Rwanda

"Rwanda integrates electronic birth and vaccine registry systems to reduce workload, improve data quality and help reach unvaccinated and undervaccinated children"

-Quote from the DHIS2 website, <https://dhis2.org/rwanda-crvs-eir-integration/>

In 2012, Rwanda became an early adopter of the DHIS2 Electronic Immunization Registry (EIR) tracker, which tracks a child vaccination history through routine vaccination schedules. Then in 2016, Rwanda launched a strategic eHealth plan to digitise all CRVS data. A web-based application was developed and debuted in 2020. The application known as the NCI-CRVS system is meant to help facilitate the collection, storage, and production of data civil status events. The NCI-CRVS system currently registers births and deaths, including the cause of death. Birth registration is done at the health facility. The baby's mother (or the person accompanying the mother) provides his or her telephone number to the nurse, who enters the newborn's information into the CRVS system. Once the registration is complete, a message is sent to the parent or guardian with the child's national registration number. They can then request the child's digital birth certificate through an online portal, Irembo, where it can be viewed as many times as needed.

While both the EIR and NCI-CRVS systems were helpful on their own, the lack of integration between the two systems led to some challenges. These included missing unique identifiers for some newborns in the EIR, and double data entry for data managers who were required to capture data for the same child in both the DHIS2 EIR and CRVS systems.

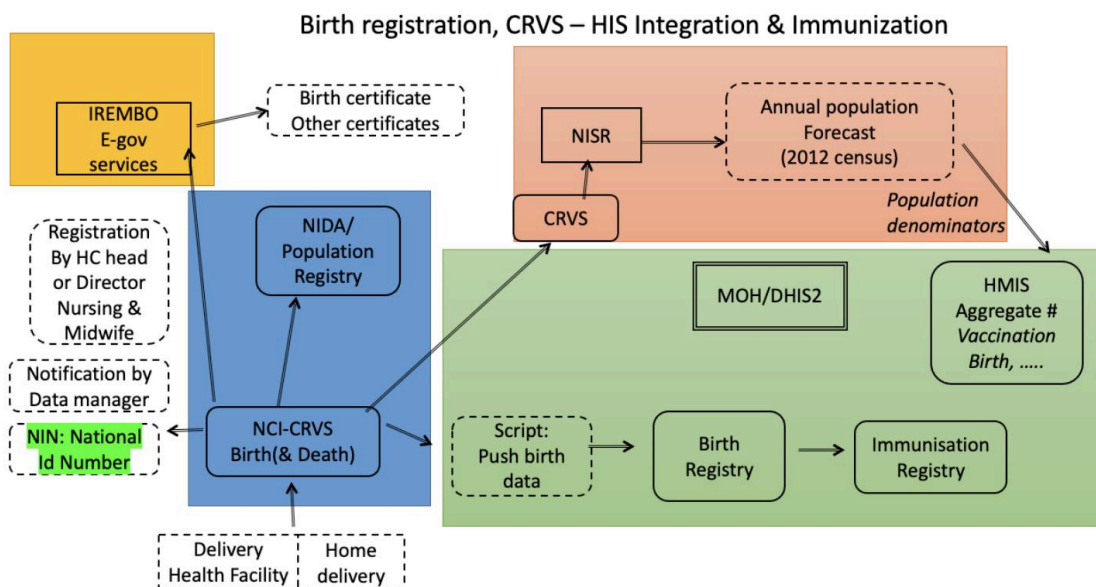


Diagram showing the integration of Rwanda's CRVS and EIR systems

Rwanda integrated the EIR and CRVS systems in 2021 and early 2022. The interoperability between these systems resulted in a simplified process for registering births and vaccinations of newborn children:

- A baby born at a health facility is notified in CRVS by a data manager and registered by a Health Center Head or Nursing & Midwife Director
- A unique number National Id Number (NIN) is issued
- The NIN is used in the Irembo portal to request a birth certificate
- A custom script pushes data on all registered newborns from the CRVS system to the DHIS2 EIR Tracker in the national HMIS system
- During vaccination, a newborn's recording is accessed and updated in DHIS2 using his or her NIN
- Reduced workload
- Eliminated double data entry at health facilities for child registration
- Automatic birth notifications in the EIR tracker can be used for follow-up
- Helps identify missing doses and "zero dose" individuals
- Improved data quality and accuracy
- Real-time records

Read more in-depth about the Rwanda use case of DHIS2 with CRVS at <https://dhis2.org/rwanda-crvs-eir-integration/>

- For a full list of impact stories and use cases visit their website: <https://dhis2.org/category/impact-stories/>
- Explore an interactive map visualising implementations of DHIS2 around the world, on the DHIS2 In Action page on their website: <https://dhis2.org/in-action/>

Some current capabilities of the solution

- Online and offline data at entry via DHIS2 webportal, SMS, mobile Android app or direct import
- Monitoring features and followup with individuals or entities over time
- Various access levels
- Data agnostic
- Language support for web-based platform
 - 7 languages
 - *Determine if the app has language and localization support and at what entry points.*
- User-friendly interface and dashboards
- No licensing or monthly fees
 - *Consider the cost of technical setup, ongoing hosting, staff time to learn and maintain the system*
- Validation checking to ensure no duplicate entries
- Visualisations galore
 - Customizable and shareable graphs, maps, reports, pivot tables, charts
- Custom DHIS2 Android Capture App

- Seamless syncing of data and integration into DHIS2 platform
- Mobile capture
- Optimised for online and offline mode
- Metadata packages
 - “DHIS2 has worked with the World Health Organization (WHO) and other partners to develop [standardized metadata packages](#) to strengthen data use on a national and international level by supporting adoption of global health data standards into national routine health management information systems. Packages are available for analytics dashboards, aggregate data collection, and case-based (Tracker) programs, and can be installed in standalone DHIS2 systems or integrated into existing DHIS2 instances.”
 - Read more and download metadata files on the [Metadata package downloads page](#).
- Suite of DHIS2 web apps
 - The [DHIS2 App Hub](#) provides links to additional web applications for the DHIS2 platform. These apps are fully compatible with the core DHIS2 software platform
 - Visit the [App Hub](#)
- SMS and Mobile Messaging Solutions available
 - Integration with RapidPro
 - Integration of DHIS 2 with RapidPro (<http://rapidpro.io/>) for creating workflows through SMS and messaging systems (Whatsapp, Telegram, facebook, Viber...)
 - <https://github.com/dhis2/integration-dhis-rapidpro>
 - A simple web-based interface for sending SMS to individual or groups of health workers or patients
 - Automatic SMS sent to patients, for example to remind them of an upcoming or missed visit, or as part of a general education program related to a health program
 - Reporting data by sending an SMS to the system
 - Sending messages from SMS to users of the system, for example for support or feedback purposes
 - Registering and enrolling a patient into a health program by sending an SMS
 - Entering individual health data for a patient visit using SMS
 - Checking the status of a patient’s followup using SMS
- Free, ongoing training series and academies
 - DHIS2 community (board): <https://community.dhis2.org/>
 - In-person events like conferences
 - Comprehensive tutorials on youtube: <https://www.youtube.com/@DHIS2org/playlists>
 - Up-to-date documentation and support
 - DHIS2 Academy: an online training & skills program <https://academy.dhis2.org/>

- Developer portal: a community designed to develop DHIS2 applications and connect with the software team

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

The Threat, Likelihood, Impact, and Severity columns listed below are in reference to the world, and specific places where CRVS solutions are implemented, and not specific to the vendor, platform, or solution itself. In addition, this is a general assessment, to give the reader an idea of what Mitigations are in place for this specific solution.

Threat	Likelihood	Impact	Severity	Mitigations
<i>Describe the potential threat, attack vector, bad actor</i>	<i>How likely is it that this could happen? (generally, not specific to vendor/platform)</i>	<i>What will happen if the threat/attack is successful?</i>	<i>How severe will the impact be?</i>	<i>How does the solution reduce the risk, impact, and severity of the attack?</i>
Identity theft or fraud	Likely -	Personal data, including that of children, is increasingly in demand by identity thieves	Moderate -	(1) Encryption and decryption of fields stored in the database (2) Support for OIDC Identity Layer with access controls and flexible deployment of authentication systems
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 -	(1) Encryption of data on the network and at rest (2) Multiple Access Control roles that limit the scope of access to data and capabilities. (3) Data retention policies and features (4) Training, education, and guidance provided by community

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 -	(1) Multiple roles that limit the scope of access to data and capabilities (2) 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 -	(1) Easy access to mobile interface, even with limited connectivity, keeps as much data reporting "in the system" and private as possible (2) 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 -	(1) Availability of deployment docs, training, and support (2) "Platform" approach creates potential for an ecosystem of certified/trusted partners for deployment (3) Transparency of open-source and iterative development means vulnerabilities and updates can be fixed and deployed quickly

* Ultimately, deployment and operational responsibilities are outside of the scope of the DHIS2 software project, though based on the mitigations listed, they do the best they can to guide and support the process

Initial Assessment Results

Initial Thoughts

General Impressions

The DHIS2 solution is a flexible and robust system which can be implemented and configured to fit many use cases. This is both a plus and a minus in the context of this evaluation. Given the vast configurability of the system and the substantial efforts by the developers to create multiple extension points to add new functionality, it is highly likely that the system could be adapted into a powerful CRVS system. On the other hand, this functionality does not exist at this time and would need to be developed prior to adoption.

Key Team Members and Roles

We connected with key members of the DHIS2 team in April over a video call. As a result of the call, we created a group chat channel on Slack and remain in contact via email as well. We also had a number of follow-up conversations and reviews of our progress. We are satisfied that the sets of skills and knowledge represented by these contacts will be sufficient for us to complete our evaluation. The DHIS2 team was responsive to our questions and spent time reviewing our evaluation findings and made comments and clarifications to improve the accuracy of our work. Below are some of the examples of some of the roles we communicated with:

- Lead, DHIS2 Training & Communication Group, University of Oslo, Norway
- Security Lead, DHIS2, Oslo, Norway
- DHIS2 implementation team members

Summary of Initial Interactions

We began our interactions with the team in April, with a kickoff call. From the call, a Slack group channel was created for further communications. The DHIS2 team is active and willing to provide any support necessary for this evaluation to be successful. Nonetheless, they are a busy team and due to our conflicting schedules we were unable to find a time for a live demo of their system. However, they have a plethora of videos on youtube and recently, from June 12-15, 2023, DHIS2 held their annual conference, #dac2023. This 4-day conference produced numerous recordings of sessions, plenary talks and future plans for the DHIS2 team ([DHIS2 Annual Conference 2023](#)). These recordings have provided useful content in our initial assessment of the DHIS2 system.

Since beginning our evaluation, the DHIS2 version 40.0 and Android app 2.8 were released, May 2023.

Concerns and Blockers

No current blockers.

- One potential unknown is the general understanding of how user roles & flows work with a CRVS implementation of DHIS2. We tried to schedule a follow up with the DHIS2 team regarding a demo, but were unable to do so. We do not feel it is imminent at this time to pursue this understanding however, we do advise that implementing organizations learn more about how these processes work.

Questions to investigate further

An interesting takeaway from our initial review is that the DHIS2 team does not have direct access to any security audits performed on their solution. This is because any audit on DHIS2 is performed by individual implementers on their own instances which are specific to their setups. However, DHIS2 will sometimes be asked to validate findings from implementers' reports and choose to publicly publish any vulnerabilities or security related activities on their website. The DHIS2 website was recently updated to provide information on their security policies, features and workflows.

On the website, DHIS2 does provide excellent, organized public information about any disclosed vulnerabilities, along with other security related updates and activities:

- Trust Center <https://dhis2.org/trust/>
 - “We are continuously improving our software architecture, features and processes to minimize the risk to users and their data. On this page you can learn about our security processes and principles.”
- Security <https://dhis2.org/security/>
 - “DHIS2 includes industry standard security and privacy features. On this page you can learn more about the customizable features that are available in the core DHIS2 software and DHIS2 Android application.”
- Security “Hall of Fame” <https://dhis2.org/security/hall-of-fame/>
 - “On this page, we recognize the people who have identified and responsibly disclosed security vulnerabilities to us. Thank you!”

We appreciate this engaged and transparent approach to security and trust.

We are still working to understand the threat and risks associated with a DHIS2 system. Furthermore, we will continue to consider the additional costs and resources needed to implement DHIS2 with a CRVS system.

Review of Assets

- Product and architecture documents, diagrams, specifications

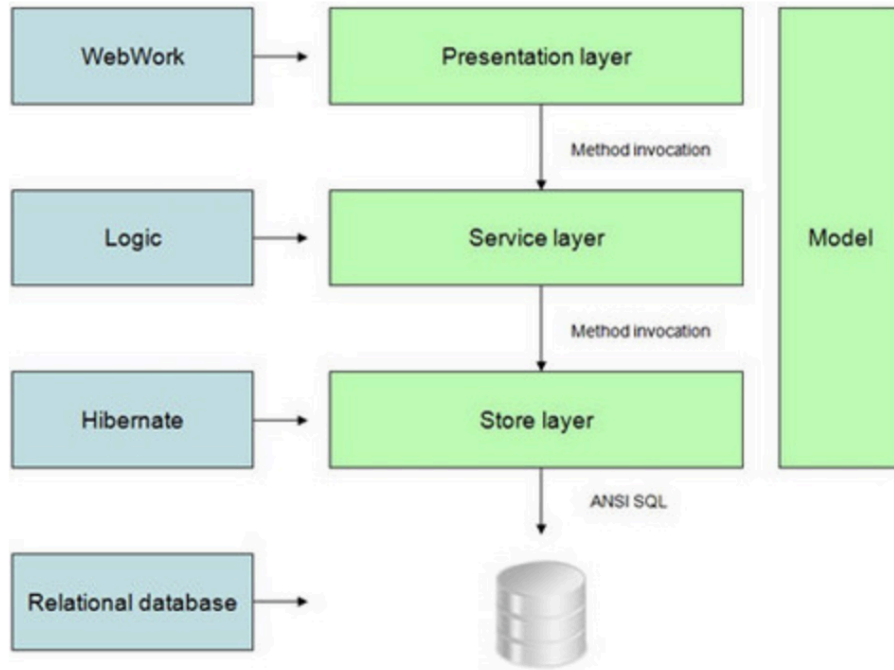


Figure showing the overall DHIS2 architecture

- DHIS2 is written in Java and has a three-layer architecture. The presentation layer is web-based, and the system can be used on-line as well as stand-alone.
<https://docs.dhis2.org/archive/en/2.24/developer/html/ch04.html>
- Overall solution website: <https://dhis2.org/>
- Overall documentation site: <https://docs.dhis2.org/en/home.html>
 - The DHIS2 documentation is divided into 4 broad categories: Use, Implement, Develop, Manage
- Stories of Impact explore how the DHIS2 solution has aided countries and communities to achieve their goals: <https://dhis2.org/category/impact-stories/>
- Use Cases share how the DHIS2 solution can be implemented and what components are provided in the package
 - WHO Health Data Toolkit: <https://dhis2.org/who/>
 - COVID-19 Surveillance: <https://dhis2.org/covid-19/>
 - Immunisation Programs: <https://dhis2.org/covid-19/>
 - Education Management: <https://dhis2.org/education/>
 - Case-based Programs: <https://dhis2.org/tracker-in-action/>
 - Mobile Data Entry: <https://dhis2.org/android-in-action/>
- DHIS2 provides additional training and support via their youtube channel and an online learning platform
 - DHIS2 Youtube Channel :<https://www.youtube.com/@DHIS2org>
 - DHIS2 Online Academy :<https://academy.dhis2.org/>

- DHIS2 is open source software, shared under the BSD 3-Clause license, and the source code is on Github <https://github.com/dhis2>
 - Resources for developers to contribute to the source code or create custom web apps can be found here: <https://dhis2.org/development/>
 - DHIS2 has created a custom android app for importing data into their system. It is integrated as a capture app for field work. The Android App SDK information is found here: <https://dhis2.org/android/sdk/>
- Access to ticketing systems, discussion boards, wikis, and other public development infrastructure
 - DHIS2 Community (board): <https://community.dhis2.org/>
 - DHIS2 Academy: <https://academy.dhis2.org/>
 - DHIS2 Developer portal: <https://developers.dhis2.org/>
- Inputs to and results from any other security audits
 - Unavailable. See comments in the section, Questions to investigate further.*
- Prerequisites for installation, use, access
 - <https://dhis2.org/security/>
- Operational deployment images, tools, and/or scripts for configuration management
 - Server tools: <https://github.com/dhis2/dhis2-server-tools>
 - DHIS2 in Docker: <https://developers.dhis2.org/docs/tutorials/dhis2-docker/>
 - DHIS2 has a server for security testing (<https://specimen.dhis2.org>), but not enough disk for intensive logging so we will setup our own DHIS2 docker installation server found at:
[\(https://developers.dhis2.org/docs/tutorials/dhis2-docker/\)](https://developers.dhis2.org/docs/tutorials/dhis2-docker/):
DHIS2_IMAGE=dhis2/core:2.40.0.1
DHIS2_DB_DUMP_URL=<https://databases.dhis2>

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 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 PDF and Syft export format are available here:

- PDF: https://drive.google.com/file/d/1Tk_W687hjol0GDMbr4mNjl7u5-_pGecx/view?usp=drive_link
- Syft: https://drive.google.com/file/d/1g7fAm81x2dQc7L4s68SWllcyC1lw_Yu1/view?usp=sharing


Open-Source Software (OSS) vulnerability scanning

We used the open-source tool 'grype' to check the source and Docker images for vulnerabilities.

Static application security testing (SAST) scanning

We used the open source SAST scanner 'semgrep' to perform an automated analysis of the Typescript/Javascript codebase.

High-level overview

Our overall impressions of the DHIS2 codebase and its security posture are positive . We have no high-level concerns regarding the source code security of DHIS2. We found a few minor issues, detailed below, but nothing of high or critical severity.

1. The backend codebase is primarily written in Java with the Spring Framework. This is an industry-standard choice for Java-based web applications and has a solid stability and security patching track record.
2. The frontend codebase is written in Javascript and the React library. While the backend is one monolithic server-side application, the frontend is split into different applications that can be dynamically installed and loaded on demand.
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 that the DHIS2 developers respond timely to security concerns and implement best practices when it comes to patching issues. The project maintains a security dashboard <https://github.com/dhis2/dhis2-core/security> and makes it simple for security researchers to disclose issues, while providing implementers of DHIS2 with the information they need to resolve found security issues.

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 deeply probed known problem areas in web applications. Our testing and audit methodology was guided by the OWASP Web Security Testing Guidelines. <https://owasp.org/www-project-web-security-testing-guide/stable/> We also produced a Software Bill of Materials (SBOM) for the DHIS2 Docker image and performed a manual review of third-party dependencies.

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 DHIS2 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

- Details of work on these and other issues discovered later in our vulnerability scanning and penetration testing work can be found later in this report.
- Public issue tracker for DHIS2 is available at: <https://dhis2.atlassian.net/jira/projects>
- Community updates, news, patches, releases and more can be followed here: <https://community.dhis2.org/> and here: <https://community.dhis2.org/tag/patch-releases>

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 DHIS2 team via our communication channel and the public Github project, as needed.

Finding 1: Improper Validation of Client ip Address for ip Allow Listing

DHIS2 allows a user to create a Personal Access Token (PAT). This token can be used essentially like a password. It allows a script, program, or other service to access DHIS2 with the user's credentials. When creating the PAT the usage of the token can be scoped to certain API operations (GET, POST, DELETE, etc) as well as scoped to a certain list of IP addresses.

When DHIS2 receives a request containing a PAT it cross-checks the requester's IP address with those in the allow list. If the requester's IP address is in the allow list, the request is allowed.

To determine the IP address of the requester DHIS2 relies on the X-Forwarded-For header.

Our security audit identified a notable issue in the application's handling of the X-Forwarded-For HTTP header. The application is designed to use this header for IP whitelisting, but it incorrectly treats the entire header value as a single IP address. This approach leads to a functional bug, as the X-Forwarded-For header often contains multiple, comma-separated IP addresses, especially when the client is behind proxies or load balancers.

Example: If a client with IP 1.1.1.1 connects through a proxy with IP 2.2.2.2, the X-Forwarded-For header received by the application will be 1.1.1.1, 2.2.2.2. The application, expecting a single IP, fails to parse this correctly and thus cannot match either IP against its whitelist, leading to legitimate requests being denied.

The DHIS documentation does document this behavior and says:

“IP address validation relies on the X-Forwarded-For header, which can be spoofed. For security, make sure a load balancer or reverse proxy overwrites this header.”

But elsewhere in the documentation includes a sample nginx config that passes the x-forwarded-for header to the application. This config includes the following line:

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

This line appends the client's IP address (\$remote_addr) to the X-Forwarded-For header. If the header is already present in the request, Nginx appends the client's IP address to the existing header.

In production deployments there can often be several reverse proxies leading to several ip addresses in the X-Forwarded-For header. DHIS2 currently simply does not support this case, and will always fail to allow the requests as described above.

Recommendation: DHIS2 provides or supports an alternative header for user IP address identification, such as X-Real-IP, and/or updates its documentation to use the [nginx http_realip module](#) and iterates the importance of correctly defining trusted proxy addresses to prevent IP spoofing.

References:

- Documentation referencing the trust of X-Forwarded-For
<https://docs.dhis2.org/en/use/user-guides/dhis-core-version-master/working-with-your-account/personal-access-tokens.html?h=x-forwarded+master#serverscript-context>
- Recommended NGINX reverse proxy config
https://docs.dhis2.org/en/manage/performing-system-administration/dhis-core-version-master/installation.html?h=x-forwarded-for+master#install_enabling_ssl_on_nginx

Finding 2: Lack of default Access-Control-Allow-Origin Header Configuration

Our source code review has identified an oversight in the default configuration and documentation of the application regarding the handling of the Access-Control-Allow-Origin header, commonly referred to as CORS. While the application correctly supports the Access-Control-Allow-Origin header validation and configuration, it does not include any allowlisted origins in its out-of-the-box installation. Furthermore, the installation documentation lacks explicit guidance on the necessity of configuring whitelisted origins for security purposes.

The absence of pre-configured allowlisted origins poses a potential security risk. This sets the stage for a Cross-Origin Resource Sharing security vulnerability in a deployed instance of DHIS2.

Incorrect or overly permissive CORS configurations can introduce several significant security risks to a web application. CORS is a mechanism that allows or restricts web pages from making requests to a domain different from the one that served the web page, and is crucial for implementing the same-origin policy.

Recommendation: DHIS2 improve the installation documentation to emphasize the importance of configuring CORS allowlisting

It should provide:

- Clear instructions on how to set up whitelisted origins.
- Best practices for determining which origins to whitelist.
- The security implications of incorrect or overly permissive configurations.

**Finding 3: Use of TripleDES for Encryption of Some Sensitive Data is moved to a private annex and has been addressed directly with the DHIS2 team.*

Finding 4: Assessment of User Input Sanitization and XSS Mitigation

Our security audit has identified a potential vulnerability in the application's approach to handling user input and mitigating Cross-Site Scripting (XSS) attacks. The application currently does not sanitise user inputs before storing them in the database, relying instead on the frontend framework, React, for XSS mitigation. While React does provide a level of protection against XSS by escaping strings in JSX, this approach has limitations, particularly in a multi-developer environment.

Implications of Relying Solely on Frontend Frameworks

1. Third-Party Developer Extensions:
 - a. The application allows third-party developers to extend its frontend. In such scenarios, the reliance on React's XSS protections may not be sufficient if a different framework is used. Third-party developers might not be fully aware of or adhere to best practices for preventing XSS, leading to vulnerabilities.
2. Long-Term Data Integrity:
 - a. Data stored in the database without proper sanitization poses a risk over time. If the application's frontend is modified or extended in the future, or if data is loaded into a different application that does not use React, previously stored unsanitized data could become a vector for XSS attacks.
3. Backend-Rendered Scenarios:
 - a. In cases where the backend directly renders data for the frontend, or in server-side rendering scenarios, React's XSS protections would not apply, potentially exposing the application to XSS vulnerabilities.

Our core recommendations are:

- **Training and Documentation:** Provide comprehensive training and documentation for all developers on the importance of input sanitization and the specific methods to be used in this application. This is particularly important for third-party developers who might not be familiar with the application's security practices.
- **Regular Code Reviews:** Implement regular code reviews and security audits to ensure that input sanitization is consistently applied across the application, especially in parts of the codebase developed by external contributors.

Optional, but recommended:

- **Explicit Sanitization Policy:** The application should adopt a policy of sanitising all user inputs before persisting them in the database. This policy should be explicitly stated and communicated to all frontend developers, including third-party contributors. Utilise robust input sanitization libraries on the server side to cleanse inputs of potentially malicious content. This ensures a consistent level of security, regardless of the frontend framework in use.

While React's XSS mitigation features provide a level of security, they should not be the sole line of defence against XSS attacks in an application that handles user inputs, particularly when third-party developers can contribute code. Improving training, documentation, and code review processes are essential steps in reinforcing the application's defence against XSS attacks. These measures ensure that all developers, including third-party contributors, are adequately equipped to handle XSS risks. Additionally, adopting an explicit sanitization policy for user inputs, while optional, is highly recommended. It serves as an extra layer of security, safeguarding the application against a broader range of XSS attack vectors and ensuring long-term data integrity.

Finding 5: dangerouslySetInnerHTML should disqualify an application from the App Hub

The current guideline for third-party app developers in App Hub advises developers against using dangerouslySetInnerHTML (a mechanism in React that opens the door for XSS).

We recommend a more stringent policy: the use of dangerouslySetInnerHTML in an application should be grounds for disqualification from the App Hub, with exceptions only considered on a case-by-case basis.

The dangerouslySetInnerHTML property in React bypasses the framework's built-in XSS protection by allowing raw HTML to be set directly from JavaScript. This opens a significant vulnerability, as it can easily lead to XSS attacks if the content is not properly sanitised, which as we have laid out in Finding 4, is the case.

Allowing applications that use dangerouslySetInnerHTML contradicts security best practices. It sends a message that the app store tolerates potentially unsafe practices, which could lower the overall security standards of the applications it hosts.

By rejecting applications that use dangerouslySetInnerHTML, the app store encourages developers to seek safer alternatives for rendering HTML content, such as using safer libraries or methods that ensure proper sanitization.

While the use of dangerouslySetInnerHTML should generally disqualify an application, there may be exceptional cases where its use is justified and secured. If all of the following conditions are met, then the use of dangerouslySetInnerHTML may be justified.

1. Specific Functional Requirements: In rare cases, an application might have a legitimate need to render HTML content directly, and no safer alternative meets these requirements.
 - a. *Example*: DHIS2's [own markdown rendering component](#) uses dangerouslySetInnerHTML. This component converts markdown content from users into HTML. However, before this process, the unsanitized user input is passed through the MarkdownIt library, which performs its own comprehensive sanitization.
2. Robust Sanitization Measures: The application must demonstrate that it implements robust, foolproof sanitization measures to cleanse any dynamically rendered HTML content, effectively mitigating the risk of XSS attacks.
3. Thorough Security Review: Applications requesting an exception must undergo a thorough security review. This review should rigorously assess the necessity of using dangerouslySetInnerHTML and the effectiveness of the implemented sanitization measures.

References:

- App Developer Documentation mentioning XSS
<https://developers.dhis2.org/docs/guides/apphub-guidelines/#secure>
- App Developer Documentation mentioning App Hub policies
<https://developers.dhis2.org/docs/guides/apphub-guidelines/#appropriate-for-dhis2>

Finding 6: More Stringent Content-Security-Policy Headers Recommended

Our security audit reveals inconsistencies in the application's implementation of Content-Security-Policy (CSP) headers. Notably, while the login page employs a robust CSP with `script-src 'self' 'nonce-[nonce-value]'`, this level of security is not consistently applied across other pages. Given that the application supports extensions by third-party developers, the lack of uniform CSP implementation presents security risks.

CSP is an important security measure in mitigating the following risks:

1. Cross-Site Scripting (XSS) Attacks:
 - a. CSP is a critical defence against XSS attacks. By specifying which sources the browser should accept scripts from, CSP prevents the execution of unauthorised or malicious scripts. Without CSP on all pages, the application is more vulnerable to XSS, especially in applications created by third-party developers.
2. Data Theft and Site Integrity:
 - a. CSP helps in safeguarding user data and maintaining the integrity of the site. It restricts resources (like scripts, images, and stylesheets) to trusted sources, thereby preventing attackers from injecting malicious content or exfiltrating information.
3. Third-Party Code Risks:
 - a. Third-party applications can introduce unknown or untrusted code. CSP serves as a safeguard, ensuring that only scripts from allowed sources are executed, thereby reducing the risk of malicious code execution.

The App Hub guidelines recommend avoiding externally hosted scripts and stylesheets, citing security and performance issues:

“Avoid externally hosted scripts and stylesheets: External scripts and stylesheets, such as those served by global CDNs, should be avoided unless absolutely necessary - these can cause security and performance issues when accessed in various global contexts.”

This is a sound policy, but we recommend enforcement of this policy via CSP headers from the backend application.

While avoiding external scripts is a good practice for the reasons mentioned, it should be complemented with a robust CSP. CSP adds an additional layer of security. Even if external scripts are used, CSP can restrict them to trusted sources, minimising the risk of malicious content. CSP provides granular control over resource loading and script execution, allowing for a more tailored security approach that aligns with the application's specific needs. As the application evolves and new features or third-party integrations are added, CSP can be dynamically updated to accommodate these changes while maintaining a strong security posture.

DHIS2's existing feature for dynamically updating CORS Access-Control-Allow-Origin headers could be used as a model for a new feature to manage Content-Security-Policy (CSP) headers.

An alternative approach could involve allowing applications to specify their CSP requirements in their manifest. This method ensures that when an application or extension is loaded, its CSP requirements are automatically applied, streamlining the security process and reducing manual configuration errors. By requiring CSP specifications in the manifest, developers are encouraged to consider and address security from the outset of application development. Having CSP requirements declared in the manifest simplifies the process of security audits, as auditors can quickly ascertain whether the specified policies align with security best practices.

We recommend the App Hub should adopt a more stringent policy for submissions:

1. Rejection of External Assets by Default:
 - a. Applications that rely on external scripts or stylesheets should be rejected by default. This policy reduces the risk of third-party vulnerabilities and ensures a higher degree of control over the application's content.
2. Disallowing 'unsafe-inline':
 - a. The policy should explicitly disallow the use of 'unsafe-inline' in CSP directives. This measure is crucial in preventing the execution of inline scripts and styles, which are common vectors for XSS attacks.
3. Case-by-Case Exceptions:
 - a. Exceptions to these rules should be granted only in specific, justified cases. Developers seeking exceptions must provide comprehensive documentation and justification for their requirements. Applications requesting an exception must undergo a thorough security review to ensure that the use of external

assets or 'unsafe-inline' elements does not compromise the application's security.

References:

- App Developer Documentation mentioning avoiding external assets
<https://developers.dhis2.org/docs/guides/apphub-guidelines/#secure>

Finding 7: Recommendations for Explicit SameSite Cookie Policy

DHIS2 utilises cookie-based authentication but does not explicitly set a SameSite policy for its cookies. While modern browsers default to a SameSite policy of 'Lax', which offers a reasonable level of security, there is an inherent risk in relying on browser defaults, particularly given the variability in user browser choices and versions.

Not all users may be using modern browsers with a default 'Lax' SameSite policy. Older or less common browsers might handle cookies differently, potentially leading to security vulnerabilities. Explicitly setting the SameSite policy ensures consistent behaviour across all browsers.

The SameSite cookie attribute is a key defence against Cross-Site Request Forgery (CSRF) attacks. By setting it to 'Lax' or 'Strict', the application can prevent the browser from sending cookies along with cross-site requests, thereby mitigating the risk of CSRF attacks.

Explicitly setting the SameSite attribute reflects a proactive approach to security, demonstrating a commitment to best practices and attention to detail in security configurations.

As a general recommendation, the SameSite attribute should be explicitly set to 'Lax'.

This setting balances security and functionality, allowing cookies to be sent in top-level navigations, which is typically sufficient for most applications.

For applications where deep linking from external sites is not a requirement, setting the SameSite attribute to 'Strict' offers an even higher level of security. This setting prevents the browser from sending cookies on any cross-site requests, further reducing the risk of CSRF attacks.

The choice between 'Lax' and 'Strict' should be based on the specific needs and behaviours of the application. Factors to consider include the nature of user interactions, the necessity of cross-site navigation, and the overall security requirements of the application.

Application Architecture Audit

High-level overview

Our overall impressions of the DHIS2 architecture and extensibility are positive .

DHIS2 has been actively developed for nearly two decades. The platform is built with widely-used languages and tooling that have stood the test of time. DHIS2 has maintained a predictable biannual release schedule for many years and the project's adoption, longevity and continuous incremental improvement speak to the solid technological underpinnings of the project.

DHIS2 is not a CRVS application, though it has been used for this purpose. It is a highly-user-configurable data collection and analysis tool that was originally created for a healthcare use case, but which has also been used in education, logistics, and human resources contexts.

Given that significant customization would likely be necessary if this project were adopted for UNICEF's CRVS needs, this architectural audit will focus largely on the questions of configuration and extensibility.

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:

- Documentation available at <https://docs.dhis2.org/en/home.html> and <https://developers.dhis2.org>
- Videos targeted toward developers on the DHIS2 YouTube channel (<https://www.youtube.com/c/Dhis2Org>)
- Streamed parts of the annual DHIS2 conference (https://www.youtube.com/watch?v=Huw_rt4x0Gs&list=PLo6Seh-066RzWHF4yzqWwXlM2fnKM-OYF)

- We also started a local cluster for testing using the DHIS2 command line tool (<https://cli.dhis2.nu/#/getting-started>) and used that tool to create a test application to extend the core functionality.

Summary of communication with vendor related to disclosures and direct feedback

There were no architectural issues requiring disclosure or mitigation discovered during this portion of the assessment.

Outcomes

Overview of the structure of the application

- The application is primarily written in Java, Kotlin and Javascript
- The backend core is written in Java using the Spring framework
 - Most of the business logic resides in this part of the application and it provides an API that both the web frontend and Android applications connect to
- The web frontend is a Javascript application written using the React framework
 - DHIS2 has created an extensive library of React UI components that are used in the web frontend and can be imported into custom applications built to extend the core functionality
- The Android application is written mainly in Kotlin, a modern interoperable Java alternative.
 - DHIS2 also created an Android SDK that can be imported into a custom Android application to facilitate data exchange with the backend.
- Data is stored in a PostGIS-enabled Postgres database
 - Postgres is arguably the preeminent open-source database and PostGIS adds standards-compliant geospatial capabilities to the core database.

Whereas other applications we have evaluated use a microservice architecture in which different app components are split into independent services that communicate with one another, DHIS2 is firmly in the opposite camp, commonly referred to as a “monolithic” architecture.

The advantage of a monolithic architecture is that development and deployment are more straightforward. The developer does not need to maintain a mental model of how different services interact and can focus on implementing business logic. A deployed monolith has very few moving parts which can make the application easier to scale and secure.

The disadvantage of this architecture from a developer's perspective is that a monolithic code base can be overwhelming and seemingly minor code changes may have far-reaching effects on other parts of the codebase. The DHIS2 core app has thousands of source code files befitting its massive featureset and long development history and contributing to this codebase would require a very significant period of exploration before a developer could confidently make changes.

A disadvantage of monolithic architecture from a deployment perspective is that if certain elements of the application require extra compute resources to scale, the entire application must be scaled up to address those requirements.

Evaluation of Configurability

DHIS2 is first and foremost a flexible, user-configurable data warehousing and reporting application. By design it makes very few assumptions about what the user is trying to achieve. The few assumptions it does make are:

1. You have data that can be captured in a structured format, likely through a web or mobile form. This data may take the form of either single data points or aggregate data.
2. You wish to report on that data, potentially in a visual manner.
3. You may have multiple projects over time that would need to be configured separately, but there may be a requirement to report across these projects.
4. Users will have predetermined roles in an organisational hierarchy that will determine access to different parts of the application and its datasets.
5. The people configuring the application may not have programming experience.

Since these assumptions are nearly universal in data collection and presentation applications, they impose few limitations.

In written and video documentation, a recurring theme is that almost every aspect of DHIS2 is configurable via the UI. From creating projects, to creating organisational hierarchies and roles, to building forms and visualisations, everything can be achieved by pointing, clicking, and typing. The configuration is all stored in the same backend database that stores the collected data. The user-facing interface for collecting data is dynamically constructed from the configuration.

DHIS2 is organised as a core offering plus a selection of independent sub-applications that can change or extend the core. To customise DHIS2, the user will use the Maintenance app. It allows configuration of:

- organisational unit hierarchy
- data elements (aka fields)

- indicators (aka calculations)
- data sets & data entry forms
- users
- user-based access controls

An understanding of the hierarchy of these entities is essential to the configuration process. If any part of the hierarchy is incomplete, the “capture” user interface will not accept any data. DHIS2 also splits data collection into periods of various lengths, and incorrectly setting the start date of a period can also prevent data entry. DHIS2 provides excellent documentation and a video introduction to the configuration process.

Though everything *can* be configured via point-and-click, many data-collection initiatives require a more formalized approach and may include so many data elements and organizational units that manual entry would be impractical and error-prone, especially if multiple users were to enter configuration data in an inconsistent manner. Thankfully, DHIS2 allows import of complete data hierarchies in the form of JSON documents. The project structure can then be version-controlled and updates made through subsequent data imports. In fact, this is the recommended method to get started with DHIS2’s existing CRVS module

(<https://docs.dhis2.org/en/topics/metadate/chis-community-health-information-system/design/civil-registration-and-vital-statistics.html>) which is available for download here: https://packages.dhis2.org/en/CRVS_VE/1.1.1/DHIS2.39/CRVS_VE_1.1.1_DHIS2.39.zip.

Evaluation of Extensibility

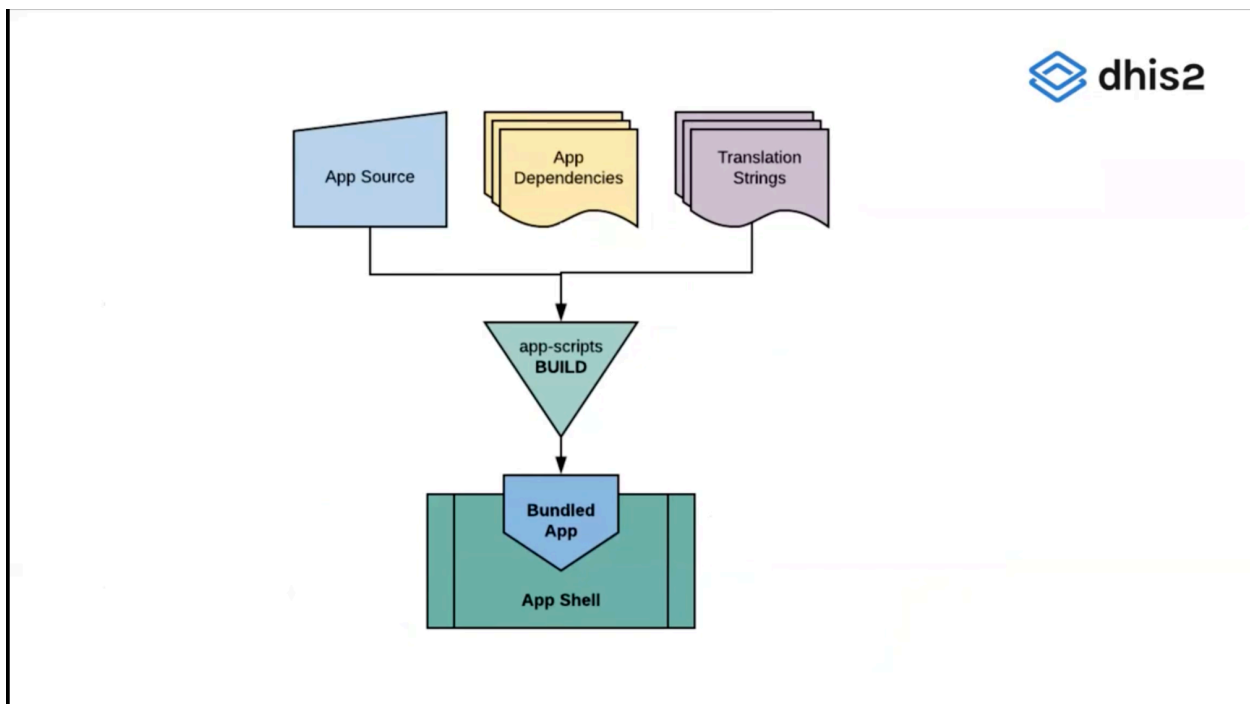
Extensibility is a core feature of DHIS2: the project’s developers have provided many extension points and tools to facilitate custom development.

When creating data-input applications, you may choose one of three types of implementations:

- *Data Forms*: Totally data-driven, based on your custom data model. They are generated automatically and offer no control over layout.
- *Section Forms*: Allow customization of the order and grouping of input elements, while maintaining a no-frills interface. They are easy to translate and update, and are compatible with web and Android.
- *Custom Forms*: Allow full customization of the user interface, but are not available on Android and require more development resources for upkeep.

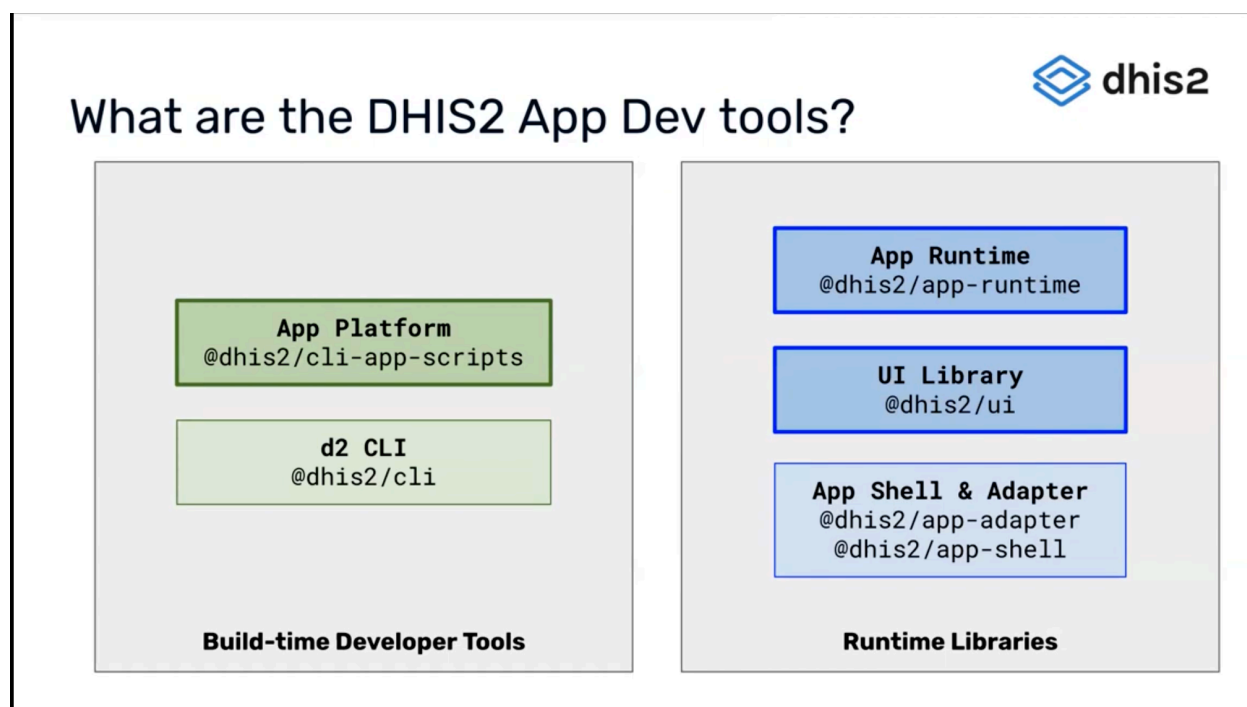
We believe that custom development will likely be required to implement DHIS2 for CRVS uses, so we focused our investigation on the tools provided by DHIS2 to facilitate this type of customization. These tools include a comprehensive REST API for reading and writing data from external applications and a “plugin” architecture that provides a standard way to create and

deploy apps within the DHIS2 platform. The developers have wisely chosen to use these tools themselves to build first-party functionality, thereby ensuring that they are feature-rich and well-maintained when used by third-parties.



The DHIS2 developers have created a number of tools to make the process of extending the core application simpler and more consistent with the platform. These include:

- a command-line tool that can create test DHIS2 clusters and generate the “scaffolding”, or default project-structure, of a custom application
- a web UI library that custom apps may use to better match the look and feel of the platform
- an “app runtime” that features a data-fetching library for web applications that wraps the DHIS2 REST API
- an Android SDK to facilitate data-fetching in custom Android applications



To evaluate the development experience we created a simple React web application that reads and writes data using the data-fetching library and displays it using the DHIS2 UI library. We also imported the provided CRVS module to create the basic data collection hierarchy.

The data-fetching library was easy to use: instead of managing many HTTP requests to various API endpoints, developers can describe the requests they wish to make as a JSON object. The library takes care of constructing the correct HTTP request and the converting parameters and results between text and Javascript objects. The results can be accessed directly inside a React component. A query for a list of programs would look like this:


```
const query = {
  results: {
    resource: "programs",
    params: {
```

```
    pageSize: 20,  
    fields: ["id", "displayName", "created"],  
  },  
},  
};
```

The UI library is extensive conforms to expectations of a React component library. It includes all of the basic elements needed to build a form-driven user interface including buttons, tables and menus. DHIS2 provides the components as both a React Storybook, which allows implementers to experiment with component configuration, and a Figma file that is optimised for a designer's workflow. The Storybook is available here <https://ui.dhis2.nu/demo/> and shows the breadth of the library.

The only minor potential complaint about the tools available is that they are written in plain Javascript at a time when most similarly-sized projects have transitioned to Typescript. As a typed-superset of Javascript, Typescript allows for clearer function interfaces, improved maintainability and better development-environment integration. However, we understand that the choice of language may represent a preference or necessity of the team, based on project history or available developer resources.

There are several planned additions to the dev tools, some of which are outlined in this slide from a video from a developer workshop:



Coming Soon to the App Platform

- **Dynamic Modules** (UI components always up-to-date!)
- **Feature Toggling** (Talk to multiple different server versions!)
- **Data Caching** (Minimize network traffic automatically)
- **Offline & PWA support** (take your DHIS2 app with you to the field)
- **Routing** (next.js-style file-based routing)
- **Web Hooks & Server-Side Extensions** (eventually)

What else? We want your feedback!

We should also note that these development tools are targeted at web-based usage. Since the components provided by the UI library are responsive, they can be used in mobile environments. If a truly native equivalent of a custom app were a requirement, then development resources would also be needed to develop an equivalent capture experience on Android.

Evaluation of maintainability and performance at scale

The DHIS2 has a long history of delivering improvements on a consistent schedule. New versions have been released at roughly 6-month intervals, typically with a major release in Spring and a patch release in the Fall. The current major version (40) was released in May 2023 and the next version will be released next May 2024. With their latest release v40 and the Android Capture App release v2.8, DHIS2 streamed a live webinar on May 10, 2023, to share updates and discuss the newest features in their release. The video can be found on YouTube at

[▶ Webinar on new DHIS2 software releases: v40 and Android v2.8](#). Additionally, DHIS2 provides guidance on future features on their roadmap (<https://dhis2.org/roadmap/>) which includes the next two major releases. The fact that the organisation has delivered updates with such consistency for more than a decade strongly implies a well-managed and maintained project.

When considering performance at scale, one of the main questions is whether the application can be scaled horizontally (adding more instances of the application as usage goes up) or vertically (increasing the compute resources available to a single instance). The technologies used in the DHIS2 application can be scaled in either dimension, but it seems most amenable to vertical scaling. Happily this method of scaling is also usually operationally simpler than the alternative. In particular Postgres is easily scaled by increasing available compute resources and given our current understanding of DHIS2 usage it seems likely that even a large installation could use a single, well-provisioned database.

The DHIS2 team provided feedback that it can also [scale horizontally through a cluster](#).

Penetration Testing Audit

Summary of test environment setup

Version 2.40.0.1
Build revision cddb161
Build date June 30, 2023 at 12:54

Test endpoint: <https://dhis.test.gpcmdl.n.net/>

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

Outcomes

The Manual Penetration Testing of the test DHIS2 instance occurred during the end of November and beginning of December 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 10 High or Medium vulnerabilities and the remaining findings and recommendations at “Low” or “Info”.

NOTE: The scores listed in the port use a custom “Risk Score” value, which is different than the typical industry Common Vulnerability Scoring System (CVSS) Score. In the table below, we will use the CVSS score. (more info at: <https://www.balbix.com/insights/understanding-cvss-scores>)

Full PDF reports of findings from the Manual Penetration Testing is available here:

- Executive Summary:
<https://drive.google.com/file/d/1hEtn1tYHOBs3B6Xe44nUIB7BxDwmKQ3j/view?usp=sharing>
- Full Report:
<https://drive.google.com/file/d/1wj9aER5p1lfQ1IGvqOZ2x0KA54GkG-82/view?usp=sharing>

High-level concerns, issues

Here is a summary of the top vulnerabilities discovered and their proposed fix. We will be sharing these with the DHIS2, and give them full access to information to reproduce these findings.

Title	Score /Severity	Description	Proposed Fix	Solution/Vendor Comments
[CRITICAL] PDF File Upload leads to Stored Cross Site Scripting (XSS)	6.3 Medium	During the Pentest it was observed that a user could trigger Stored XSS vulnerability by uploading a malicious pdf	In order to prevent Stored XSS attacks, the best way is to handle the input securely in both client-side and server-side code in a proper manner before it gets stored permanently on the webserver	Fixed in next release: https://dhis2.atlassian.net/browse/DHIS2-16845 review related to CORS settings, and feedback that attack is difficult to implement, and focus is on infecting client and not compromising the data of the service itself
Dangerous JS Functions	7.4 High	Most common JavaScript attacks vectors include: executing malicious script, stealing a user's established session data or data from the browser's localStorage, tricking users into performing unintended actions, exploiting vulnerabilities in the source code of web applications.	Employ JavaScript security best practices to reduce this risk	
Sensitive information Disclosure (Access to System Configuration)	4.3 Medium	During the Pentest it was observed that a low-level user could access sensitive information such as Server Settings by accessing the affected API as mentioned in the POC	Make it mandatory for developers to declare 'Allowed' access for each resource, and by default, deny it	False positive. These API endpoints provide configuration settings for all authenticated users and don't contain any sensitive information. In some specific setups, one of these endpoints may contain an SMS gateway configuration

				(although in an encrypted form), which is unnecessary to disclose. There is a remediation task to remove this encrypted snippet as well.
Unauthorised Message Send	6.6 Medium	During the Pentest it was observed that an unauthorised user could send a message in an ongoing message thread	Make it mandatory for developers to declare 'Allowed' access for each resource, and by default, deny it	It will be fixed in the next release. Issue ID: https://dhis2.atlassian.net/browse/DHIS2-16846
Anti-CSRF Token Missing	6 Medium	The site appears to be vulnerable to a cross-site request forgery (CSRF) attack. With every POST request, an anti-CSRF token should be sent	Enable CSRF Protection, Use Security Libraries and Frameworks, Generate a Unique Token, Verify the Token	
CORS Misconfiguration	6 Medium	The site appears to be vulnerable to a cross-site request forgery (CSRF) attack. With every POST request, an anti-CSRF token should be sent	Limit the Origins, Configure Headers, Use Pre-Flight Requests, Use Authentication	Could not reproduce this issue and would like to see more evidence.
Server Leaks Version Information via "Server" HTTP Response Header Field	5.7 Medium	This information can be valuable to attackers as it reveals the specific version of the software running on the server, which can be used to identify known vulnerabilities and attack the server	Remove the version information from the Server HTTP response header field	This issue is outside of the DHIS2 scope. The version information can be reported by software (application server, reverse proxy web server) that DHIS2 runs on. We have no control over how implementers configure their systems, although we will add some guidance on this topic to our documentation

CSP: style-src unsafe-inline	5.2 Medium	These attacks are used for everything from data theft to site defacement or distribution of malware	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header	
Unauthorised Google Maps API Key Usage	6.5 Medium	We were able to find that Google Maps API Key is vulnerable to unauthorised access by other applications	Please follow the API best practices from https://developers.google.com/maps/api-key-best-practices	This is a legacy Google Maps API key intended for public use. It was also hard-coded in the Github repository. However, we are moving away from this practice and will remove this key entirely both from the code and public access.
Software Component Version Leaked	4.9 Medium	The web/application server is leaking information	Ensure that your web server, application server, load balancer, etc. is configured to suppress software version leaks	Out of scope. The version information can be reported by software (application server, reverse proxy web server) that DHIS2 runs on. We have no control over how implementers configure their systems, although we will add some guidance on this topic to our documentation.
CSP: script-src unsafe-inline	4.8 Medium	These attacks are used for everything from data theft to site defacement or distribution of malware	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header	
Vulnerable JS Library	4.8 Medium	A JavaScript library that is missing security patches can make your website extremely vulnerable to various attacks	it is recommended to keep all JS libraries used in the application up-to-date and to regularly check for any known vulnerabilities in the libraries	
Username Enumeration via Forgot Password	5.3 Medium	it was observed that the user could enumerate valid registered users by	It is recommended to show a response that does not show variation leading to	Issue tracked here: https://dhis2.atlassian.net/browse/DHIS2-16848

		analysing the response the server sends back on triggering a reset password request	username enumeration when brute forced	
--	--	---	--	--

Feedback

Summary of communication with vendor related to disclosures and direct feedback

We have provided the DHIS2 access to this document, and given them opportunity for direct feedback and communication with us through multiple channels. You can see their comments in the table above.

Status of any mitigations, patches, updated releases

There has been significant progress to review, reproduce, address and resolve the open issues that have been found. We look forward to continuing our communication with the DHIS2 team to support any future response to this audit process.

Evaluation of the holistic approach in terms of cyber security

/ Analysis of history of public vulnerabilities

DHIS2 provides a public forum named “The DHIS2-Security Community of Practice” at <https://community.dhis2.org/tag/dhis2-security>. Their transparent and engaged approach to this is excellent and laudable. Through this site you can see any history of announcements, patches, and public disclosures for the last few years.

Overall, DHIS2 works well as an active, engaged, transparent open-source project that has to respond to the discovery of bugs, flaws, and vulnerabilities in a public way.

DevSecOps Analysis

Summary of test environment setup, steps taken to complete analysis

An article¹ in the dhis2-server-tools documentation describes the different approaches to the installation of DHIS:

- Manual installation

¹<https://github.com/dhis2/dhis2-server-tools/blob/b90659fcf1dfefaf24983cd5c1e493d5b5274594/docs/Different-install-approaches.md> (Archive: <https://archive.ph/hxM0c>)

- dhis2-server-tools (Ansible) <https://github.com/dhis2/dhis2-server-tools>
 - Recommended solution that is actively developed and promoted within DHIS2 ecosystem.
- dhis2-tools-ng (bash) <https://github.com/bobjolliffe/dhis2-tools-ng>
 - Bash script implementation (hosted outside of DHIS2 organisation for historical reasons). It is still supported and used where Ansible is not an option.
- Docker
- Kubernetes

The application is formed of two primary components: a Java web application intended to be hosted by Apache Tomcat 9 and a PostgreSQL database. As previously mentioned in the application architecture audit this is a monolithic application rather than an architecture using microservices. This monolithic architecture does not lend itself to taking advantage of the features of Kubernetes and a 3-tier architecture (<https://www.ibm.com/topics/three-tier-architecture>) would be better suited.

The DHIS2 team provided feedback that through use of the DHIS2 Android SDK, to enable organisations and users to create their own mobile applications, their solution can fully reflect a 3-tier architecture .

The ability to quickly deploy using Docker for development and testing purposes has value and Docker is commonly used for this purpose but the learning curve and infrastructure investment required between Docker and a Kubernetes cluster are vastly different.

Manual Installation Method

The review of the manual installation method is based on the instructions published on the DHIS2 documentation website². DHIS2 appears to be a flexible application, however this review will focus on the CRVS use case.

Test Environment Deployment Overview

These instructions recommend the Ubuntu 18.04 LTS operating system, which has been out of support since May 2023 unless ESM support is paid for (available until April 2028). Ubuntu 18.04 shipped with Tomcat 9, and an attempt to use Tomcat 10 failed (it is EOL as of September 2022), and so to ensure that an in-support system was deployed for testing the deployment used Debian 11 which has the end of security support scheduled for July 2024. Tomcat 9 and PostgreSQL 13 were both installed from the official Debian packages.

2

<https://docs.dhis2.org/en/manage/performing-system-administration/dhis-core-version-master/installation.html> (Archive: <https://archive.ph/fYClE>)

DHIS2 team provided feedback on November 28th that, *“We support all current Ubuntu LTS releases, including 20 and 22. The documentation page which refers only to 18.04 LTS needs to be updated.”*

The installation instructions begin with creating a new unprivileged user for running DHIS2 and configuring the server timezone and locale. These two steps show attention to detail and consideration of how the system should be deployed in production, rather than just being able to make it run for development and testing purposes. This is further seen in the steps relating to tuning PostgreSQL performance and configuration of Tomcat’s memory usage. The instructions do not just present a list of tunables and the suggested values but also an explanation of each so that system administrators can decide if the suggested value is right for them, and so that developers can validate that those values are still useful to suggest generally.

Outcomes

Operation best practices

Host Operating System

As discussed above, the recommended operating system version is out of mainline support, however the application does not have strong dependencies on any operating system services and other operating systems are available with security support carrying suitable versions of the Java runtime and Tomcat server. The Debian operating system, based on the Linux kernel, has good all-round hardware support and it is unlikely that this would be a restricting factor in the choice of physical hardware on which to run the application.

We recommend updating the public documentation of supported operating systems to support the latest, most accurate options.

Database Configuration

The application uses the industry standard PostgreSQL server and does not place any specific requirements on that server that would preclude the use of a centrally managed PostgreSQL cluster should an institution have already invested in that infrastructure. Competent administrators should be readily available for recruitment to manage the PostgreSQL cluster given the widespread use of the technology.

File Store Configuration

DHIS2 is capable of using AWS S3 to share files between nodes of a cluster, however the instructions do not indicate a method of specifying an alternative endpoint URL. The AWS S3

API is implemented by other tools that can be either self-hosted (e.g. Minio) or may be hosted as part of a private cloud arrangement. NFS can be difficult to configure in a resilient way and the use of object storage would remove that issue. Local legislation and regulations may require that data is stored on systems that are compliant with standards that AWS S3 may not be compliant with.

OpenID Connect Configuration

DHIS2 supports the OIDC identity layer for single-sign on. While this feature was not configured or evaluated here, it is good to see that this feature is offered by the solution. This will allow for seamless single-sign on for users, and allow for conditional access policies such as device posture or network location to be controlled according to the policies of the organisation or agency operating the software. Support for “generic” providers is implemented so this feature is not tied to only specific cloud vendors as is often the case with software claiming to support OIDC single-sign on.

Database Encryption

Encryption and decryption of fields stored in the database is performed by the Java Cryptography Extension (JCE), a commonly used cryptography extension to the Java language that will have received a good degree of scrutiny to identify flaws in the implementation of the various algorithms. From a review of the codebase, it appears that most strings are encrypted using AES-128 which should be considered sufficiently strong to protect the kind of data that is typically entered into a CRVS system. However, some system settings are encrypted using Triple-DES with a comment that it is due to a bug in Jasypt³. This bug was not investigated here, but it should be checked if there are any sensitive fields that are being encrypted with Triple-DES as the algorithm has been considered unsafe for some time. See *the Source Code Audit section for a more extensive discussion on this topic*.

Secrets Management

As the test environment has been deployed manually, no prescriptions were made by the DHIS2 documentation about how to manage secrets within the deployment infrastructure-as-code. No issues were discovered during the deployment that would have precluded the adoption of best practices for secret management.

Operations Management Review

System Updates

In the manual installation, the language runtime and associated dependencies were installed from the operating system’s package manager. The Debian package manager, also used by Ubuntu, provides simple commands to update the system packages and can be configured to automatically install security patches as they become available.

3

<https://github.com/dhis2/dhis2-core/blob/376fc9b874593a9d6a69584bce803a0783e4fe4a/dhis-2/dhis-support/dhis-support-hibernate/src/main/java/org/hisp/dhis/config/HibernateEncryptionConfig.java#L49>

Security Announcements

Security issues are announced via the DHIS2 community forums and are tagged for easier discovery⁴. An RSS feed is available⁵ to allow for easy subscription and does not require directly interacting with the community forums web interface.

Application Updates

Application updates are performed by replacing the WAR file used by the Tomcat server with the latest release available on the website. This process may be automatable for patch releases, however for minor releases there can be manual steps required for upgrades. Each minor release is published with upgrade notes⁶ which cover both the impact on API users as well as the necessary manual steps required to complete the upgrade, for example manual SQL queries that must be executed.

Documentation

System administrator documentation⁷ is contained within the wider DHIS2 documentation. It covers installation, upgrading, monitoring, audit, SMS gateway configuration, and user impersonation for troubleshooting. This documentation does not feel comprehensive or polished, and doesn't make reference to the frameworks for deployment, e.g. dhis2-server-tools.

We recommend updating the System Administration documentation to be more comprehensive.

Translation Support

Internationalisation is a first class feature in DHIS 2⁸. Internationalisation of the user interface is supported through the use of Java property strings and PO files. Java property files are used when messages originate from the back-end Java server, while PO files are used for front-end apps written in JavaScript. Transifex, a robust industry standard translation system, is used to manage translated strings.

Monitoring and Observability

⁴ <https://community.dhis2.org/t/new-dhis2-security-tag-for-all-important-security-alerts/45342> (Archive: <https://archive.is/nrQLd>)

⁵ <https://community.dhis2.org/tag/dhis2-security.rss>

⁶ <https://github.com/dhis2/dhis2-releases/blob/master/releases/2.37/README.md> (Archive: <https://archive.is/QMQSg>)

⁷

<https://docs.dhis2.org/en/manage/performing-system-administration/dhis-core-version-240/installation.html>

⁸ <https://dhis2.org/localization/>

DHIS 2 integrates a Prometheus exporter⁹ that exports a number of metrics from nodes running the DHIS 2 application:

- DHIS2 API (response time, number of calls, etc.)
- JVM (Heap size, Garbage collection, etc.)
- Hibernate (Queries, cache, etc)
- C3P0 Database pool
- Application uptime
- CPU

Other software components, e.g. PostgreSQL and Redis, will use their own Prometheus exporters as is best practice. The metrics available for API usage should be sufficient to detect when service is degraded or failing but may not be helpful in identifying root causes. Combined with lacking system administrator documentation, this may result in longer outages when they do occur as it may be necessary to resort to the community forums for answers.

Backups and Disaster Recovery

Backups for DHIS2 consist of a dump or snapshot of the PostgreSQL database and a clone or snapshot of the data directory or object storage backend. The system administrator documentation does not include information about disaster recovery procedures but does give information on which data must be backed up in order to be able to recover. This would be an area for improvement in the future. The lack of this documentation has led other users to ask for help on this topic in the community forum¹⁰.

Production Deployment Guidance

Hardware Requirements

An in-country production deployment of DHIS 2 is likely 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 DHIS 2 and will be applicable to all the vendors assessed in this project.

Server hardware will be required to run the 3 components of the application: reverse proxies, application servers and database servers. The Redis component may be co-located with the

⁹

<https://github.com/dhis2/wow-backend/blob/master/guides/monitoring.md#dhis2-monitoring-configuration> (Archive: <https://archive.is/dVKJd>)

¹⁰ <https://community.dhis2.org/t/options-for-backing-up-dhis2/52799/2> (Archive: <https://archive.is/8BJld>)

application servers or with the database servers. The servers should have redundant power supplies with at least one supply being connected through a UPS battery backup system.

Further specifics will depend on the scale and nature of the exact deployment, but it will be important to consider:

- Target uptime
- Likelihood of power interruption
- Likelihood of network connectivity interruption
- Environmental factors that may affect MTBF for hardware
- Storage requirements including backups and off-site backup storage

Network Architecture

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

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. 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.

If possible, controlling access between the reverse proxy servers, the application servers and the database cluster with ACLs will provide further security benefits by preventing lateral movement. Using TLS to secure connections between the three clusters will also reduce the risk of data leakage through physical attacks on the infrastructure or compromise of the network hardware.

Secrets Management

We recommend that Secrets should be maintained in a password manager and passed to the deployment tools at runtime

The environment that has access to the secrets should be tightly controlled. Mozilla's sops may be used to encrypt the relevant secrets although other systems are available. 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.

Final Report and Recommendations

Overall Findings

● Positive.

The DHIS2 solution is highly adaptable, capable of being customised to suit a variety of use cases. Its extensive configurability and the developers' efforts in adding multiple extension points for new functionalities suggest it could be effectively transformed into a robust Civil Registration and Vital Statistics (CRVS) system. However, this specific functionality is not currently available and would need to be developed before the system could be adopted for CRVS purposes. The interaction with the DHIS2 team has been productive, including a video call in April with key team members, the establishment of a group chat on Slack, and ongoing email communication. They have been active reviewers and commentators throughout this evaluation process. The skills and knowledge of their various team's are deemed adequate for the evaluation process.

The initial interactions with the DHIS2 team were promising, beginning with a kickoff call and followed by active communication through a Slack channel. While a live demo could not be arranged due to scheduling conflicts, valuable information was gleaned from the team's YouTube videos and recordings of their annual conference (#dac2023), which provided insights into the system's capabilities and future plans. The recent release of new versions of DHIS2 and the Android app further highlights the system's ongoing development. While no major blockers have been identified, there is a need to understand better how user roles and flows would integrate with a CRVS implementation of DHIS2. The DHIS2 architecture is three-layered, web-based, and Java-written, with extensive documentation and resources available, including impact stories, use cases, training videos, and a developer portal. The solution's open-source nature and the community's active involvement in its development are also notable. Security remains a critical aspect, with the DHIS2 team transparently handling vulnerabilities and security updates.

The DHIS2 codebase and its security posture have been assessed positively. The backend, written in Java using the Spring Framework, aligns with industry standards for Java-based web applications and demonstrates a strong track record in stability and security patching. The frontend, developed in Javascript with the React library, contrasts with the backend's monolithic structure by being divided into multiple applications that are installable and loadable on demand.

The project exhibits robust automated test coverage, and the development team adheres to best practices in feature implementation and testing. A notable aspect of DHIS2 is its proactive approach to security. The developers respond promptly to security concerns and follow best practices for patching issues. This is evidenced by the maintenance of a security dashboard on GitHub, which facilitates issue disclosure by security researchers and provides implementers with necessary information to address security issues.

In terms of security concerns, the analysis revealed no major issues with the source code of DHIS2. Only minor issues were identified, none of which were of high or critical severity. To conduct this assessment, the team reviewed the application source code, set up a development instance following the project's documentation, and conducted in-depth probes into known web application problem areas, guided by the OWASP Web Security Testing Guidelines. Additionally, a Software Bill of Materials (SBOM) was created for the DHIS2 Docker image, and a manual review of third-party dependencies was performed, further solidifying the security evaluation of the DHIS2 system.

The overall impression of the DHIS2 architecture and its extensibility is notably positive. With nearly two decades of active development, DHIS2 is built using well-established languages and tools that have proven reliable over time. The platform's consistency in maintaining a biannual release schedule and its sustained adoption and incremental improvements underscore its strong technological foundation. Originally designed for healthcare data collection and analysis, DHIS2's high configurability has enabled its application in diverse fields such as education, logistics, and human resources. However, for use in UNICEF's CRVS (Civil Registration and Vital Statistics) projects, significant customization would be required. Therefore, the architectural audit is primarily focused on assessing DHIS2's configurability and extensibility to meet these specific needs.

Since the DHIS2 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	The DHIS2 codebase and its security posture have been assessed positively.	See 7 findings and recommendations.
Application Architecture	Positive/ Ready	The DHIS2 architecture and its extensibility is notably positive.	Must consider additional customization and integration required to fully support CRVS requirements.
Penetration Testing	Positive/ Pending resolution of high issues	While high issues were found, these are largely in deployment configuration, and not faults in the core application.	Ongoing testing, scanning and vigilance is necessary to ensure a secure solution.
DevSecOps	Positive/ Ready	The DHIS2 deployment tools and guides were usable and functional.	See recommendations related to updated documentation and deployment information.

Actionable Recommendations

- General Recommendations
- Documentation
 - Training and Documentation: Provide comprehensive training and documentation for all developers on the importance of input sanitization and the specific methods to be used in this application. This is particularly important for third-party developers who might not be familiar with the application's security practices.
 - The system administrator documentation does not include information about disaster recovery procedures but does give information on which data must be backed up in order to be able to recover. This would be an area for improvement

in the future. The lack of this documentation has led other users to ask for help on this topic in the community forum.¹¹

- We recommend updating the public documentation of supported operating systems to support the latest, most accurate options.
- Source Code
 - Findings 1: DHIS2 provides or supports an alternative header for user IP address identification, such as X-Real-IP, and/or updates its documentation to use the [nginx http realip module](#) and iterates the importance of correctly defining trusted proxy addresses to prevent IP spoofing.
 - Findings 2: DHIS2 improve the installation documentation to emphasise the importance of configuring CORS allowlisting
 - Findings 3: It is strongly recommended to migrate to a more secure encryption algorithm like AES and to implement a comprehensive encryption strategy that includes secure key management practices.
 - Findings 4: Regular Code Reviews: Implement regular code reviews and security audits to ensure that input sanitization is consistently applied across the application, especially in parts of the codebase developed by external contributors.
 - Finding 5: We recommend a more stringent policy: the use of `dangerouslySetInnerHTML` in an application should be grounds for disqualification from the App Hub, with exceptions only considered on a case-by-case basis.
 - Finding 6: We recommend the App Hub should adopt a more stringent policy for submissions:
 - i. Rejection of External Assets by Default: Applications that rely on external scripts or stylesheets should be rejected by default. This policy reduces the risk of third-party vulnerabilities and ensures a higher degree of control over the application's content.
 - ii. Disallowing 'unsafe-inline': The policy should explicitly disallow the use of 'unsafe-inline' in CSP directives. This measure is crucial in preventing the execution of inline scripts and styles, which are common vectors for XSS attacks.
 - iii. Case-by-Case Exceptions: Exceptions to these rules should be granted only in specific, justified cases. Developers seeking exceptions must provide comprehensive documentation and justification for their requirements. Applications requesting an exception must undergo a thorough security review to ensure that the use of external assets or 'unsafe-inline' elements does not compromise the application's security.
 - Finding 7: As a general recommendation, the SameSite attribute should be explicitly set to 'Lax'.
- Application Architecture

¹¹ <https://community.dhis2.org/t/options-for-backing-up-dhis2/52799/2> (Archive: <https://archive.is/8BJld>)

- Significant customization would likely be necessary if this project were adopted for UNICEF's CRVS needs.
- Penetration Testing
 - There are a number of critical, high and medium issues to review, verify and fix.
- DevSecOps
 - 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
 - Recommendation: Secrets should be maintained in a password manager and passed to the deployment tools at runtime

Closing

DHIS2 is a well-established solution with a strong network of developers and users and a history of delivering production-quality code. During our evaluation we found DHIS2 to be extensible and adaptable at the code-level. It is also a highly-user-configurable data collection and analysis tool, with almost every aspect of DHIS2 configurable via the UI. The application will need configuration to be used as a CRVS solution, and the implementing party may require significant resources to set up the application, understand the workflows and fields necessary to “capture” data and ensure proper data flows and validation. It may also be desirable to build a custom Android app to capture data, depending on the requirements for mobile usage. Additionally, updated CRVS-specific documentation would be needed to successfully customise and secure this solution as a CRVS option for UNICEF.

Our evaluation does not dive deeply into the breadth of community networks, training/tutorial academies, government relations and policy guidance required for a successful CRVS rollout. We highly recommend UNICEF pursue an understanding how user roles and flows work with a CRVS implementation of DHIS2 before adopting. Some of these questions we will address during our Stream 2 research, however, at a glance DHIS2 does provide support in many of these areas.

Overall, we find the DHIS2 solution to have a long history of successful implementations with a robust, engaged group of contributors and team members. It is an open-source solution, developed transparently, with open documentation, issue tracking, bug reporting and auditing. DHIS2 has many of the right “nuts and bolts” necessary to build a well-rounded, integrated CRVS solution. If UNICEF has a team of knowledgeable people to explore the codebase and configure the remaining custom pieces necessary to make a CRVS system integrated with DHIS2, then we believe it has great potential. Our team approves DHIS2 as a viable solution option when integrated with a CRVS system. We have enjoyed reviewing the DHIS2 solution.

Appendix

- Website: <https://dhis2.org/>
- Documentation link: <https://dhis2.org/resources/>
- Developer portal: <https://developers.dhis2.org/>
- Resources page (including developer support): <https://dhis2.org/resources/>
- Academy link: <https://dhis2.org/academy/>
- Case Studies: <https://dhis2.org/in-action/>
 - Rwanda use case of a deployed DHIS2 eCRVS solution:
<https://dhis2.org/rwanda-crvs-eir-integration/>
 - For a full list of impact stories and use cases visit their website:
<https://dhis2.org/category/impact-stories/>
 - Explore an interactive map visualising implementations of DHIS2 around the world, on the DHIS2 In Action page on their website:
<https://dhis2.org/in-action/>
- Github issue list for core DHIS2 codebase: github.com/dhis2
- PDF reports of findings from Manual Penetration and Vulnerability Scans:
 - Guardian Project DHIS2 - Pen Test and Security-Scan-Report - 121323.pdf
- Software Bill of Materials (SBOMs):
https://drive.google.com/file/d/1Tk_W687hjol0GDMbr4mNjl7u5-_pGecx/view?usp=sharing