



EU-CSI CYBERSECURITY POLICY

TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
1. LEGAL ISSUES & AUTHORIZATION	5
1.1 CONFIDENTIALITY AGREEMENT	5
1.2 POLICY COMPLIANCE	5
1.3 EU CSI PLATFORM EXTENTIONS	5
1.4 NON-COMPLIANCE	5
2. ZERO TRUST POLICY	5
3. THE EU-CSI SPECS	6
3.1 EU-CSI ARCHITECTURE	6
3.2 INFRASTRUCTURE SPECS	7
4. KUBERNETES CLUSTER	7
4.1 CLUSTER CONFIGURATION	7
4.1.1 Services & Containers	7
4.1.2 Network Topology	7
4.1.3 Cluster Hardening	8
4.2 IMAGE SECURITY	8
4.2.1 Image scanning and Vulnerability management	8
4.2.2 Container Runtime Security	9
4.3 NETWORK SECURITY	10
4.3.1 Network policies	10
4.3.2 Network status	10
4.3.3 Network security	11
4.4 ACCESS MANAGEMENT	11
4.5 BACKUPS	11
4.6 MONITORING	11
5. APPLICATION	12
5.1 APPLICATION SECURITY	12
5.1.1 Secure Design and Architecture	12
5.1.2 Security Testing – Vulnerability Management	12



5.2 AUTHENTICATION AND AUTHORIZATION	13
5.3 AUDITING	13
5.4 BACKUPS	14
5.5 LOGGING AND MONITORING	14
5.5.1 Logging	14
5.5.2 Monitoring	14
6. DATABASE	14
6.1 AUTHENTICATION	14
6.2 ENCRYPTION	14
6.2.1 Transport Layer Security (Encryption-in-transit)	14
6.2.2 Transparent Data Encryption (Encryption-at-rest)	14
6.3 BACKUPS	15
6.4 MONITORING	15
7. EMAIL SERVER	15
7.1 NETWORK POLICIES	15
8. RECOMMENDATIONS	15
8.1 FULL DECOUPLING OF PROD & STAG ENVIRONMENTS	15
8.2 EU LOGIN – 2FACTOR AUTHENTICATION	ERROR! BOOKMARK NOT DEFINED.
8.3 AZURE WEB APPLICATION FIREWALL	15



EXECUTIVE SUMMARY

The purpose of this documentation is to provide a comprehensive overview of the cybersecurity policies applied to the EU Cybersecurity Index (EU-CSI) platform, both in terms of software solutions and infrastructure in general. The Executive Summary section provides a high-level summary of the applied policies, while the technical details are explained in the remainder of this document. More specifically, the technical details outline the measures taken to safeguard against unauthorized access, data breaches, and other potential cyber threats. Through the implementation of these policies, the main focus is to maintain the confidentiality, integrity, and availability of application's data and services while minimizing the risk of cyber-attacks.

The main aspects of the cybersecurity strategy for the operation of the EU-CSI platform are summarized below.

Zero trust policy: According to this well-established security model, we assume that no user, device, or application is trusted by default and all requests should be verified and authenticated before served. We implement this policy by enforcing strong (multi-factor) authentication, role-based access control, network segmentation and continuous monitoring of all activities. More technical details can be found in *Section 2: Zero trust policy*.

EU-CSI application and infrastructure secure architecture: We follow a modular, componentised architecture that facilitates addressing security concerns and monitoring of various components in isolation. Additionally, the various components are shipped and deployed as independent containers. In addition to the application components, there are several infrastructure elements that enforce or enhance security, including the Identity and Access Management module and the API Gateway. More technical details can be found in *Section 3: The EU-CSI Specs*.

Kubernetes cluster: Kubernetes is a container orchestration technology that is mainly used for high-availability, scalability, performance, and security. In terms of security configuration, we enforce cluster nodes and namespaces segmentation, as well as Virtual Private Networks (VPNs) for restricted access for external networks. For image security, we enforce image scanning and vulnerability management for every release of the EU-CSI platform. The runtime containers are also subject of security policies, including use of trusted source, SSH-only access and resource usage limitations. For network policies, we employ encryption, network segmentation and IP whitelisting. More technical details can be found in *Section 4: Kubernetes cluster*.

Application security: Extending on the EU-CSI architecture, we apply well-established secure design principles, including the principle of the fewest privileges, expectation of attacks, and avoidance of security through obscurity. Rigorous scheduled security testing and vulnerability management is also in place, in various levels of unit testing, integration testing and automated quality analysis, for every release of the EU-CSI platform. Authentication and authorized access to the application is realized by the EU Login module based on the ECAS authentication system. Additionally, a Web Application Firewall (WAF), a full backup service, and a detailed auditing module is enabled to log all user and system actions. Finally, monitoring policy is implemented to send email alerts to stakeholders in cases of application or infrastructure failures or unavailability. More technical details can be found in *Section 5: Application*.

Database: The EU-CSI platform uses a SQL-based Relational Database Management System (RDBMS) for permanent storage of application-related data. All access to the database is guarded by SQL authentication, while the database is executed in an isolated container. For both data-at-rest and data-in-transit, we enforce SSL/TLS, certificate and key-based encryption techniques. More technical details can be found in *Section 6: Database*.

Email server: For the extensive use of email alerts and communication within the context of the EU-CSI platform, we have setup an email service that runs on a virtual machine that communicates with the Kubernetes cluster. We have applied a series of policies for the security of the email server, including blocking of all incoming traffic, disabling of SSH access, enabling of a single port for communication between the email server and the Kubernetes cluster. More technical details can be found in *Section 7: Email server*.

1. LEGAL ISSUES & AUTHORIZATION

This chapter summarizes the legal aspects concerning the use of EU CSI PLATFORM as a tool.

1.1 CONFIDENTIALITY AGREEMENT

For the confidentiality agreement please check the related file “**EU CSI PLATFORM Privacy Statement.pdf**”, which is publicly available here:

<https://eu-csi.enisa.europa.eu/privacy-statement.pdf>

1.2 POLICY COMPLIANCE

The InfoSec team will verify compliance to this policy through various methods, including but not limited to, periodic walk-thrus, video monitoring, business tool reports, internal and external audits, and feedback to the policy owner.

1.3 EU CSI PLATFORM EXTENTIONS

Any extension to the policy must be approved by the InfoSec team in advance.

1.4 NON-COMPLIANCE

An employee or user found to have violated this policy may be subject to disciplinary action, defined by the ENISA management.

2. ZERO TRUST POLICY

Zero Trust security is a security model that is designed to prevent unauthorized access to resources within the Azure platform. The Zero Trust model assumes that no user, device, or application should be trusted by default, and that all requests to access resources should be verified and authenticated before being granted access.

Our Zero Trust security achieves this by enforcing strong authentication, access control, and monitoring policies for all resources within the Azure environment. This includes implementing multi-factor authentication (MFA), role-based access control (RBAC), network segmentation, and continuous monitoring of all activities.

Key components of Zero Trust Security:

1. Identity and Access Management (IAM)
2. Network segmentation
3. Conditional Access
4. Monitoring and Analytics

The EU-CSI platform has been designed and implemented to comply with the Zero Trust security model of the Azure cloud platform. This ensures that all access to application resources is verified and authenticated, and that users and devices are granted the minimum level of access required to perform their tasks.

3. THE EU-CSI SPECS

3.1 EU-CSI ARCHITECTURE

The components (and their corresponding technologies) of the EU CSI are depicted in the following figure.

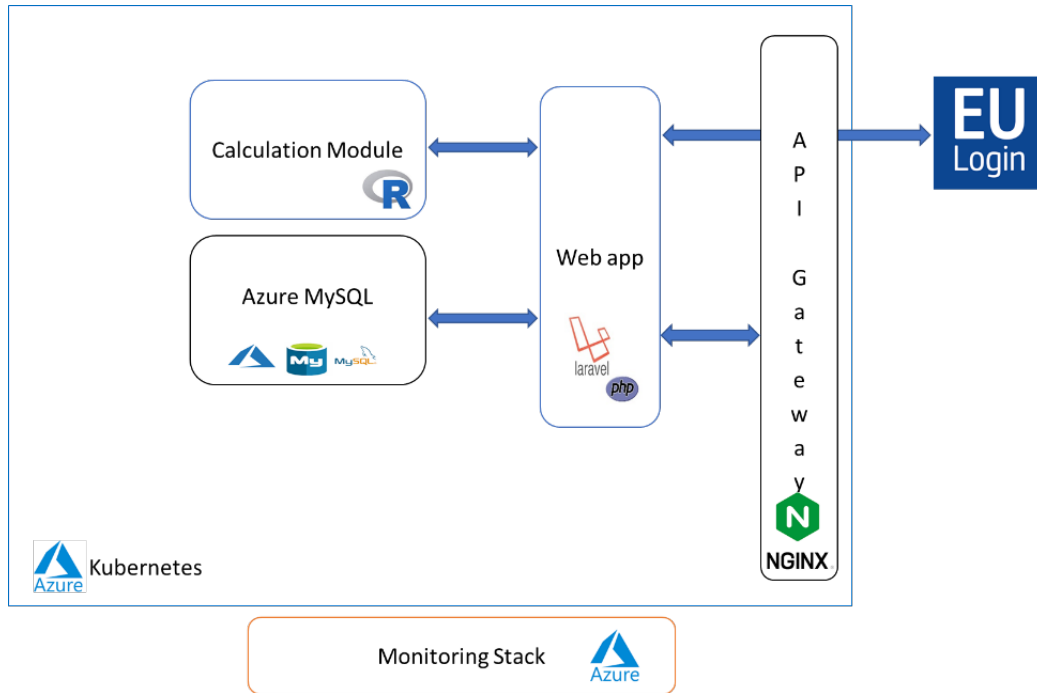


Figure 1 High-level architecture of EU CSI

Component Name	Description	Technology
Web UI	The graphical User Interface provided to the end-user in the form of a web application.	Laravel Blade ¹
Core	The EU CSI Core is a modular component comprising: <ul style="list-style-type: none"> A Rest API The Index Calculation Module The Data Ingestion Module Notification Module 	PHP Laravel ²
Index Calculation Module	The CoinR module is used for calculating the index and for producing reports. The module is exposed internally as a Rest API, used by the EU CSI Core.	COINR ³
Gateway	Assuming the presence of multiple nodes (that could be also created on demand) on the Kubernetes cluster to accommodate increasing service demand levels, this module will also be responsible for the system's dynamic load balancing in response to incoming requests by distributing the load evenly to available nodes to achieve (enabled by Kubernetes) high system availability and performance with optimized use of resources.	Nginx ⁴

¹ <https://laravel.com/docs/5.1/blade>

² <https://laravel.com/>

³ https://knowledge4policy.ec.europa.eu/composite-indicators/coinr_en

⁴ <https://www.nginx.com/>

RDBMS	The data store is based on a Relational Database System. All information that needs to be persistent will be stored in this DB. Relying in an Azure fully managed MySQL database provides scalability and high availability of the data.	Azure Mysql ⁵
Identity & Access Management	All users login through the ECAS EU login system. However, in order to manage the access control (authorization) an Identity and Access Management module is required (implemented in the EU CSI Core).	EU Login PHP client
Monitoring Stack	The monitoring stack collects security, audit and application logs from the various components and provide a UI with search and filter capabilities.	Azure

3.2 INFRASTRUCTURE SPECS

The current infrastructure specifications are the following:

- Azure Kubernetes Service (AKS) with the following specs:
 - Multiple availability Zones
 - Node size: Standard_D2_v2 (7GiB Memory, 2cores)
 - Autoscaling enabled (if autoscaling is not enabled, at least two nodes are required)
- Azure Database for MySQL
 - Resource type: Flexible server
 - MySQL Version: 8
 - Workload type: For small or medium size databases
 - Compute + Storage: General Purpose, D2ds_v4
 - Enable High availability: checked
 - DB Backup enabled with 30 days retention period

4. KUBERNETES CLUSTER

4.1 CLUSTER CONFIGURATION

4.1.1 Services & Containers

Cluster comprises of the following services/containers:

Service	Container name	Image (current)
EU CSI Core	csi-container	eucsistg.azurecr.io/csi_app:v1.1.6
Gateway	nginx-container	eucsistg.azurecr.io/csi_nginx:v1.1.6
Calculation Module	calcmodule-container	eucsistg.azurecr.io/calcmodule:v1.0.0

4.1.2 Network Topology

1. Cluster nodes segmentation: We have separated Kubernetes nodes to different availability zones, to prevent full cluster failure in case of a zone failure.
2. Namespaces segmentation: Application components are divided into namespaces restricting communication between pods and services of different namespaces.

⁵ <https://azure.microsoft.com/en-us/services/mysql/#overview>

3. Virtual Private Network: Kubernetes components have been deployed to a private network with its own subnet, isolated from public internet. Access to cluster is permitted from a list of IP addresses, configured at Load Balancer level. See section 4.2.2

4.1.3 Cluster Hardening

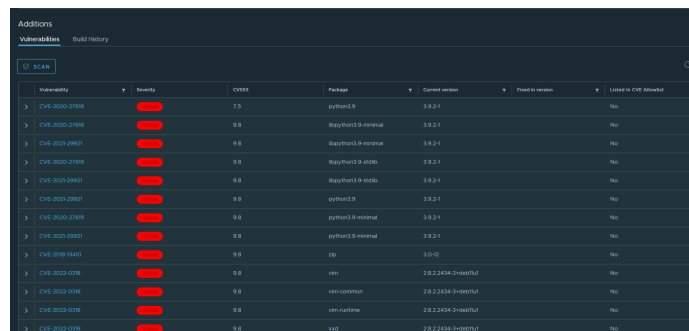
Cluster updates: Kubernetes components are updated regularly to the latest versions, that include security patches and fixes for known vulnerabilities. See section 4.2.1

4.2 IMAGE SECURITY

4.2.1 Image scanning and Vulnerability management

Every image used, before uploaded to Azure container registry, is uploaded to an in-house container registry (Harbor⁶) and scanned for vulnerabilities using integrated image scanning tools. A CI/CD pipeline has been implemented for the build and upload to Harbor process. Another CI/CD pipeline exists to upload from Harbor registry to Azure registry. When a job of the second pipeline finishes, a process is initiated to scan the uploaded image every day for new vulnerabilities and alert if any found using a cronjob.

Vulnerability Scan: Vulnerability scans are enabled in Harbor registry to run every time a new image (or new image tag) is uploaded. The scan is done using an integrated vulnerability scanner (Trivy) which returns vulnerabilities found and the severity of the vulnerability, including possible fix with the regarding package version (see below example).



Vulnerability	Severity	CVEID	Package	Current version	Fixed in version	Linked to CVE database
> CVE-2020-27398	CRITICAL	7.5	python3.8	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-minimal	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8-note	3.8.2-1		No
> CVE-2020-27398	CRITICAL	9.8	ipython3.8	3.8.2-1		No

Figure 2 Image Vulnerability Assessment in Harbor

Cronjob for vulnerability scans: Every time a new image (or new image tag) is pushed from Harbor to Azure registry, a process is initiated to continue monitoring the image for new vulnerabilities or fixes to previously found vulnerabilities. The process is configured to run every day. Main steps:

- Extract previous vulnerability report for an image.
- Perform a new scan on the same image and extract report.
- Compare two reports for new findings.
- In case of new findings (new vulnerabilities or fixes for existing vulnerability) a MS Teams webhook alert is triggered providing information of the report.

General flow from development to image upload at Azure container registry is depicted in the below diagram.

⁶ <https://goharbor.io/>

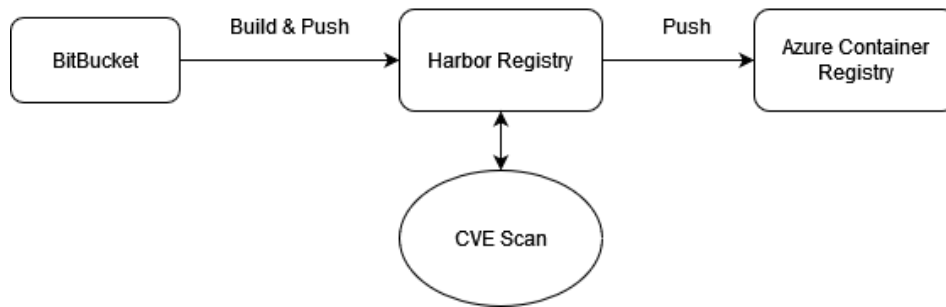


Figure 3 Process for container image deployment

4.2.2 Container Runtime Security

Policies applied to enhance container runtime security:

- **Use of container images from trusted sources:** All container images used in the application come from verified publishers to limit exposure to malicious content.
- **SSH access:** Disabling of SSH access to Kubernetes pods and containers is applied to enhance container runtime security by reducing the attack surface and potential entry points for attackers to gain unauthorized access to the system or network.
- **Resource usage limitation:** Kubernetes resource quotas and limits are applied to restrict the amount of CPU, memory and other resources that containers can use. This will prevent resource exhaustion attacks that can impact the performance of the Kubernetes environment.

4.3 NETWORK SECURITY

4.3.1 Network policies

Network policies applied to the Kubernetes cluster:

1. Encryption: All network traffic within the cluster is encrypted using SSL/TLS protocols to protect sensitive data from being intercepted.
2. Network segmentation: Different components of the application have been separated into network segments and policies defined that allow communication only between authorized segments.
3. IP whitelisting: IP whitelisting has been implemented and applied to load-balancer level to restrict traffic to pods only from a specific set of IP addresses and IP ranges. This helps to prevent attacks such as DDoS, brute-force attacks or unauthorized access to sensitive data.

4.3.2 Network status

4.3.2.1 Ports – Services Definition

STAGING ENV

Service	Port	Port Type	Exposed
calculator-module	80	ClusterIP	Internal
csi-app-service	9000	ClusterIP	Internal
nginx	80	LoadBalancer	External
nginx	443	LoadBalancer	External

PROD ENV

Service	Port	Port Type	Exposed
calculator-module	80	ClusterIP	Internal
csi-app-service	9000	ClusterIP	Internal
nginx	80	LoadBalancer	External
nginx	443	LoadBalancer	External

4.3.2.2 IP Whitelisting

The staging Environment is restricted to a specific set of Ips, listed below.

STAGING ENV

IP Ranges	Owner
79.129.1.187/32	ITML HQ
94.70.167.80/32	ITML HQ

147.67.34.29/32	EU Login
147.67.210.29/32	EU Login
172.65.32.248/32	Let's Encrypt

4.3.3 Network security

NGINX is a high-performance, scalable, secure, and reliable web server and a reverse proxy. NGINX enables all the main web acceleration techniques for managing HTTP connections and traffic. It is configured to act as a secure application gateway to pass traffic from users to the application.

NGINX capabilities applied and used to enhance level of security:

- **Rate limiting:** Rate limiting is configured to limit the amount of HTTP requests a user can make in a given period of time.
 - STAGING: **Shared memory zone**, to store the state of each IP address and how often it has accessed a request-limited URL, is configured to keep information for approximately 160000 IP addresses. **Rate**, for the maximum request rate, is configured to 2 requests per minute.
 - PROD: **Shared memory zone**, is configured to 160000 IP addresses while **Rate** to 2 requests per minute.
- **SSL/TLS encryption:** HTTPS is enabled to enforce SSL/TLS encryption for all incoming requests. To create valid certificates for the HTTPS, a certbot container is running for the NGINX pod. Certbot is also used to renew any expired certificate. In addition, NGINX is configured to accept only TLS versions 1.2 and 1.3.
- **Hide Server Info:** NGINX is configured to hide server information such as server name and version, to prevent misuse of the information for malicious purposes.
- **SSL Ciphers Configuration:** NGINX is configured to support a range of SSL Cipher suites.

4.4 ACCESS MANAGEMENT

Access management for the Kubernetes cluster is done using the Azure Active Directory (AD).

Following table shows the current users and roles specified to access the Kubernetes cluster:

ENVIRONMENT	USER	ROLE
PRODUCTION	dkalampalikis (ITML)	Admin
PRODUCTION	vchatzigiannakis (ITML)	Admin
STAGING	dkalampalikis(ITML)	Admin
STAGING	vchatzigiannakis(ITML)	Admin

4.5 BACKUPS

For backup policies applied to Kubernetes cluster please refer to the Disaster Recovery documentation

4.6 MONITORING

For Monitoring and Alerting policies applied to Kubernetes cluster please refer to the Disaster Recovery documentation, sections 4.1.1 Kubernetes Cluster Performance and 4.2.1 Cluster Alerts respectively.

5. APPLICATION

5.1 APPLICATION SECURITY

5.1.1 Secure Design and Architecture

In the specifications and design phase of this project, we followed the principles of the Security by Design paradigm, which advocates for alternate security strategies, tactics and patterns to be considered at the beginning of a software design; according to the project's needs and context, the most appropriate ones are selected and enforced by the architecture, and then they are used as guiding principles for developers throughout the course of product development. In this approach, security is considered and built into the system at every layer and starts with a robust architecture design. Examples that have been adopted in the EU-CSI Tool architecture are authentication, authorization, confidentiality, data integrity, privacy, accountability, availability, safety and non-repudiation.

The fundamental guidelines for Security by Design include a) **the principle of fewest privileges**; when executing a task assign the minimum privileges required to roles that execute that task. In EU-CSI, we applied this principle both at the application level, as well as the infrastructure level, b) **expect attacks**; incorporate technologies (e.g., secure data transfer) and mechanisms (e.g., end-point protection) to defend against expected attacks, and c) **avoid security through obscurity**; we have adopted a modular approach to the design of the tool to address this issue. It is often the case that complex design is used to address security concerns. In reality, when attacked, complex architectures prove to be more vulnerable and more difficult to monitor and trace. With a modular approach, we make sure that each module follows the appropriate security measures in isolation, and also the communication and interactions between them is also secured.

For the EU-CSI, we designed a system consisting of a modular backend and a front-end web app. The back-end modules and front-end web app are delivered in containerized form and will be deployed and managed by a Kubernetes cluster hosted on the Azure cloud framework.

When necessary, modules are exposed as services (Rest APIs), to be consumed by the web app through an API Gateway component that manages and routes all requests accordingly and also acts as a Load Balancer. The core of the EU-CSI back-end is a Data Factory that comprises a group of modules and databases for connecting to and receiving data from external sources and input entered by the users of the EU-CSI web app with the purpose of validating, aggregating, and fusing the collected data for persistent storage. The components of the EU-CSI Tool are depicted in Figure 1, presented in Section 3.1 of this document.

5.1.2 Security Testing – Vulnerability Management

Security testing is integrated into each stage of the application development lifecycle.

This is done with:

- **Unit Testing:** Unit tests are automated tests that can be run during development to check functionality of code and test security of application by checking for vulnerabilities and potential exploits.
- **Integration Testing:** At the end of each iteration and before each release of the product, we required that a set of integration tests would be in place. These tests guarantee that the features, that are guaranteed to work correctly as units by means of unit tests, interact and collaborate correctly to deliver the desired functionality as a whole.
- **Test scenarios:** At each iteration, we maintained and updated an Excel file with detailed steps for end-users to follow and cover all functionality offered at any moment. The users are able to mark their progress and comment on any missing information, unclear or non-intuitive steps and gave feedback for the improvement of the tools. The users can see the title of the scenario, the concrete steps to follow, any relevant data needed and the expected result. The users then can report any deviation from the expected result and mark the test as passed or failed.
- **SonarQube:** SonarQube is a static code analysis tool which, when integrated into the development, can help identify potential security vulnerabilities and other code quality issues. It can provide a comprehensive analysis of the codebase, including identifying potential security issues, code smells, and bugs. The Dashboard view of Sonarqube for the EU-CSI Tool is shown below in Figure 4, summarizing the number of bugs, vulnerabilities, security hotspots, etc., as well as the grade (A to F) for the code quality of the project.

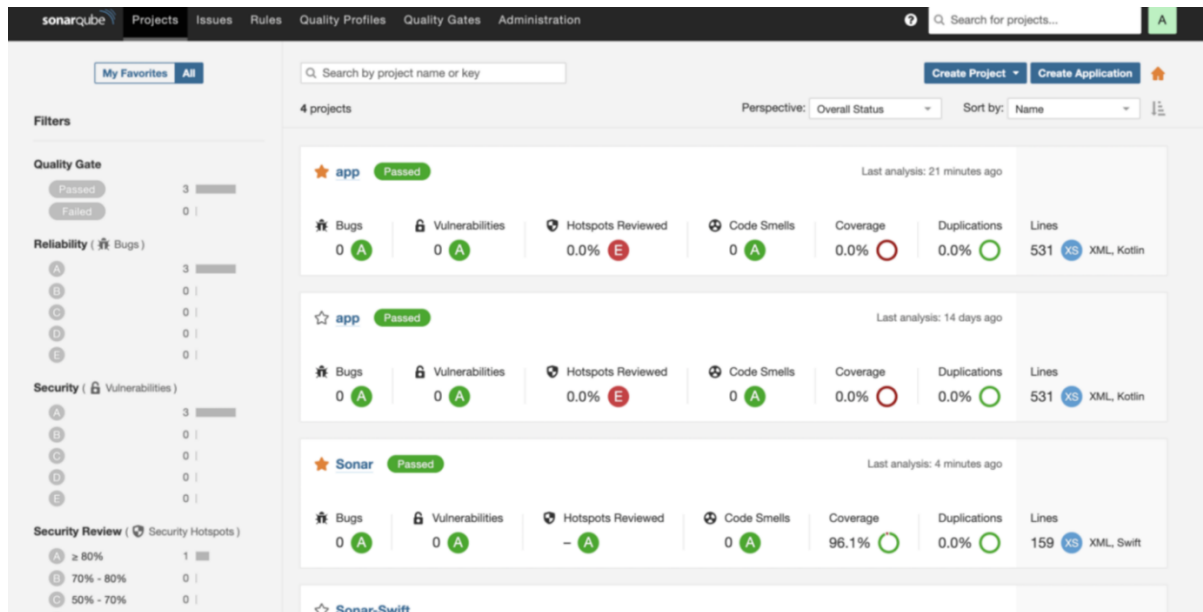


Figure 4 Sonarqube Dashboard view

5.2 AUTHENTICATION AND AUTHORIZATION

Authentication and Authorization is done using EU Login Module. Before entering the application, all users must be authenticated through EU Login Module.

Enabling Two-factor authentication (2FA) is one of the most effective measures to combat credential theft. It requires a second piece of information beyond username and password. We have enabled 2FA, particularly EU Login 2FA, mandatory for its IT systems handling Sensitive Non-Classified (SNC) information. Users are requested to provide additional means of identity verification, other than their email/password information, including the EU-Login mobile app or SMS/QR code verification.

5.3 AUDITING

EU CSI Core has an auditing feature. Every login and write action is logged in a dedicated table. The web app provides a dedicated GUI (only to ENISA admins) with search and filtering capabilities.

The interface provides information about the type of event (e.g update/delete/create or login), in different kind of database tables (users, indexes, configurations, etc).

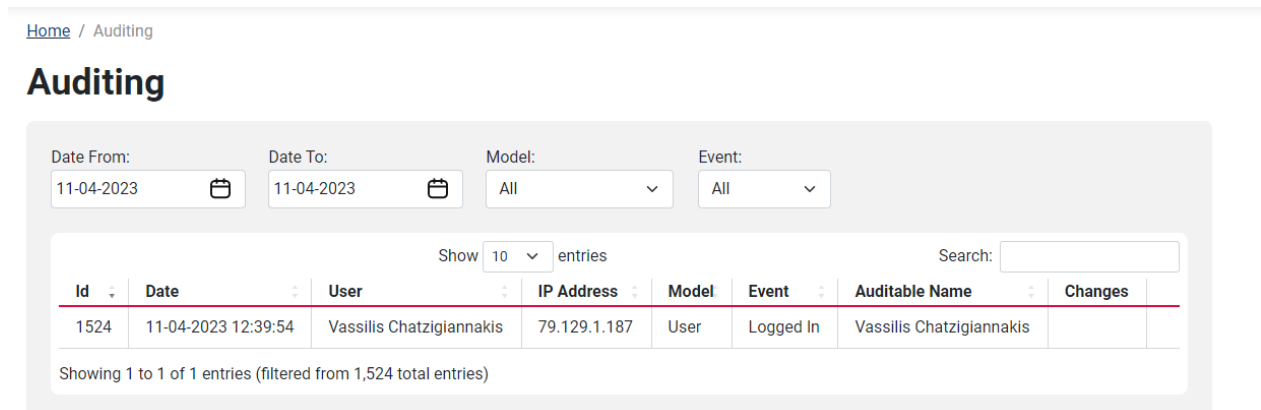


Figure 5 Auditing

5.4 BACKUPS

All application data are stored in the deployed SQL Server at a corresponding database, based on the environment, hence, backup policies are applied at database level.

5.5 LOGGING AND MONITORING

5.5.1 Logging

Logging policies defined on application level:

- Log Aggregation: Logs generated by the application are aggregated to a centralized location, which helps to identify issues and errors.
- Log levels: The application generates logs at different levels (debug, info, warning, error) depending on the severity of the issue, providing assistance on identifying issues that require immediate attention.
- Log retention: Log retention policies are established to ensure that logs are retained for a specific period of time, based on regulatory compliance requirements.

5.5.2 Monitoring

Monitoring policy is implemented to send alert via email to stakeholders in case of an application failure – unavailability (Application generates 500 HTTP response code).

6. DATABASE

6.1 AUTHENTICATION

SQL authentication refers to the authentication of a user when connecting to Azure SQL Database or Azure SQL Managed Instance using username and password. By default, Azure SQL database provides a server admin which is responsible to create additional SQL logins and users, that can connect to database using username and password.

Users and roles defined per database:

Environment	Database	User	Role
STAGING	eucsistg	eucsistg	db_datawriter, db_datareader
PRODUCTION	eucsi	eucsi	db_datawriter, db_datareader

6.2 ENCRYPTION

6.2.1 Transport Layer Security (Encryption-in-transit)

Azure SQL Database enforce encryption (SSL/TLS) at all times for all connections. This ensures all data is encrypted "in transit" between the client and server irrespective of the setting of Encrypt or TrustServerCertificate in the connection string. An encrypted connection and not trust the server certificate configurations are specified in application level, forcing the application to verify the server certificate and thus prevents the application from being vulnerable to man in the middle type attacks.

6.2.2 Transparent Data Encryption (Encryption-at-rest)

Transparent data encryption adds a layer of security to help protect data at rest from unauthorized or offline access to raw files or backups. SQL databases are encrypted by default and the database encryption key is protected by a built-in server certificate. Certificate maintenance and rotation are managed by the service and require no input from the user. TDE performs real-time I/O encryption and decryption of the data at the page level. Each page is decrypted when it's read into memory and then encrypted before being written to disk. TDE encrypts the storage of an entire database by using a symmetric key called the Database Encryption Key (DEK). On database startup, the encrypted DEK is decrypted and then used for decryption and re-encryption of the database files in the SQL Server database engine process.

6.3 BACKUPS

For the applied backup policies please refer to the Disaster Recovery document section 2.2.1 Azure Managed Backup.

6.4 MONITORING

For the applied monitoring policies please refer to the Disaster Recovery document section 4.1.2 Database Health Dashboard.

7. EMAIL SERVER

7.1 NETWORK POLICIES

For the communication between Kubernetes services and the virtual machine that hosts the email server, a virtual network has been created.

- All inbound traffic to the virtual machine from public internet has been blocked.
- By default, SSH to the virtual machine has been disabled, but is accessible using Azure Cloud Shell.
- VM's port 25 is exposed internally to the virtual network to allow communication between Kubernetes services and email server.

8. RECOMMENDATIONS

8.1 FULL DECOUPLING OF PRODUCTION & STAGING ENVIRONMENTS

Currently the staging and production environment use the same infrastructure. In Kubernetes, the staging and production are in different namespaces. This ensures that they utilize different internal networks, but both are deployed in the same cluster/nodes.

Moreover, they utilize a single instance of the Azure MySQL Server (with different databases and user credentials). Although the staging containers cannot communicate with the production containers, it is advisable to completely separate the two instances. This action would increase the Azure resources and consequently the costs of the EU CSI infrastructure (to lower the extra cost we can disable redundancy in the staging cluster).

8.2 AZURE WEB APPLICATION FIREWALL

Azure Web Application Firewall (WAF) provides centralized protection of web applications from common exploits and vulnerabilities.

AZURE Web Application Firewall Rules

Azure Web Application Firewall thwarts known exploits by applying rules to an app's incoming HTTP/HTTPS requests. A rule is a firewall code designed to recognize and prevent a particular threat. The rules that Azure Web Application Firewall uses to detect and block common vulnerabilities are mostly managed rules that belong to various rule groups. Each rule group is a collection of rules and a managed rule set is collection of rule groups. AZURE WAF also provides room to create custom rules by creating conditions that include the following components:

- Match type such as geo location, IP address, size, string
- Match variables such as RequestHeader, QueryString, RequestUri, RequestBody, Cookies, or PostArgs

- HTTP/HTTPS request methods such as POST or PUT
- Operators such as Equal Contains, Regex, Begins with, Any, Ends with
- An action such as Allow, Block, Log or Redirect

Geo-filtering

AZURE WAF gives the ability to create geo-filtering custom rules, by defining a specific path on an endpoint to either allow or block access from specified countries/regions.

Modes

Azure Web Application Firewall can be configured to run in the following two modes:

- **Detection mode:** Monitors and logs all threat alerts. You turn on logging diagnostics for Application Gateway in the Diagnostics section. You must also make sure that the WAF log is selected and turned on. Web application firewall doesn't block incoming requests when it's operating in Detection mode.
- **Prevention mode:** Blocks intrusions and attacks that the rules detect. The attacker receives a "403 unauthorized access" exception, and the connection is closed. Prevention mode records such attacks in the WAF logs.