SMARTBEAR

CALL H2020-SC1-FA-DTS-2018-2020
Trusted digital solutions and Cybersecurity in Health and Care
TOPIC DT-TDS-01-2019
Smart and healthy living at home

SMART BEAR

"Smart Big Data Platform to Offer Evidence-based Personalised Support for Healthy and Independent Living at Home"

D24 (D5.6) – Report on Continuous Security Assurance & Privacy by design - enabling mechanisms v3

Due date of deliverable: 30/09/2023
Actual submission date: 28/02/2024

Grant agreement number: 857172
Start date of project: 01/09/2019

Lead contractor: CNR
Duration: 60 months

Project funded by the European Commission within the EU Framework Programme for Research and Innovation HORIZON 2020	
Dissemination Level	
PU = Public, fully open, e.g., web	✓
CO = Confidential, restricted under conditions set out in Model Grant Agreement	
CI = Classified, information as referred to in Commission Decision 2001/844/EC.	
Int = Internal Working Document	

D24 (D5.6) – Report on Continuous Security Assurance & Privacy by design - enabling mechanisms v3

Editors

L. Koumakis, G. Kalogiannis, O. Dountouraki (STS)

Contributors

Athanatos Manos, Barmpaki Anthi, Diamantaris Michalis, Hatzivailis George, Ioannidis Sotiris, Kopanaki Depoina, Lakka Eftychia, Shetsov Alexander (FORTH)

Reviewers

Emmanouil Michalodimitrakis (FORTH)

Lisa Gius (2B)

The work described in this document has been conducted within the project SMART BEAR, started in September 2019. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857172

Executive Summary

The objective of the SMART BEAR platform is to seamlessly integrate diverse sensors, assistive medical devices, and mobile technologies to facilitate continuous data collection from the daily activities of elderly individuals (participants in the SMART BEAR study). This data is then analysed to gather insights necessary for delivering personalized interventions aimed at promoting their well-being and autonomy. Following the collection of health-related data from clinical pilot patients through various channels such as device usage logs, the SMART BEAR mobile app, and collaborative efforts between SMART BEAR, Smart4Health, and HoloBalance projects, our technical solution employs advanced big data analytics and machine learning capabilities. This enables large-scale data analysis to derive the evidence essential for tailoring interventions to individual needs.

To ensure the utmost privacy and security, the SMART BEAR platform implements data handling mechanisms designed with privacy preservation and security at their core. These mechanisms encompass all facets of data management, including data at rest, in processing, and in transit, covering comprehensively all components and connections utilized by the SMART BEAR platform.

This document represents the culmination of the activities undertaken in "WP5: Secure & Privacy-aware Data Handling." It introduces additional functionalities compared to those outlined in D5.4, reflecting insights gained from extensive evaluation during the Pilot of Pilots in Madeira, as well as feedback gathered from end users such as clinicians and help desk assistants who will utilize the services during the main pilots. D5.6 is associated with KPI-14, 15, 16, 17, and 18, and in conjunction with D5.5 (demonstrator), it presents the third operational iteration of mechanisms addressing the security and privacy requirements of the entire solution.

Contents

List of acronyms

(m)IoT	Mobile Internet of Things
API	Application Programming Interface
CCM	Clinical Case Manager
CI/CD	Continuous Integration and Continuous Deployment
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation
GUI	Graphical user interface
HDO	Help Desk Operator
IT	Information Technology
KPI	Key Performance Indicator
ML	Machine Learning
PET	Privacy measures and Enhancing Technologies
PROD	Production Environment
REST API	Representational State Transfer API
RBAC	Role-Based Access Control
SPAP	Security and Privacy Assurance Platform
SB	SMART BEAR
SB@Cloud	SMART BEAR Cloud
SB@Dashboard	SMART BEAR dashboard
SB@SecurityComponent	SMART BEAR Security Component
STS	Sphynx Technology Solutions
TRL	Technology Readiness Level
UUID	Universally unique identifier
UEBA	User and Entity Behavior Analytics
UI	User interface

List of tables

List of figures

1 Introduction

1.1 Purpose of the document

One technical objective of the SMART BEAR (SB) project is to collect medical data/measurements and integrate diverse (m)IoT devices and sensors from various vendors. This integration aims to facilitate continuous data collection from the daily lives of SB study participants. The gathered data will be analyzed using Machine Learning (ML) predictive models developed within WP4 to provide evidence for personalized interventions promoting healthy living.

Within the framework of WP5, mechanisms are being developed (and refined to meet evolving requirements, such as synergies with HoloBalance and Smart4Health) to ensure the security and privacy of data stored in the SB platform (SB@Cloud), as well as the integrity of the platform itself.

The security features implemented within the Security Component (SB@SecurityComponent) focus on maintaining the integrity, confidentiality, and availability of data at rest, in transit, and during processing for data flows (KPI-15) when interacting with external devices and systems (such as smart house sensors, IoT device usage data via smartphone applications, and others). These mechanisms adhere to the "Privacy and Security by Design" principle, embedding security and privacy measures and enhancing technologies (PETs) directly into the system design.

1.2 Delta – Updates from the second iteration (D5.4)

This deliverable primarily focuses on integrating new functionalities to address feedback from clinicians and other end users of the SMART BEAR platform, as well as introducing the Security and Privacy Assurance Platform within SMART BEAR. These new features were developed based on existing foundations (as outlined in the final architecture presented in D2.2 and the core components supporting authentication and authorization outlined in D5.2) to ensure the anonymity of newly introduced data structures (such as device inactivity).

Furthermore, the report covers the migration of the SMART BEAR platform from Docker to Kubernetes and the transition from WSO2 to KeyCloak for identity and access management solutions. Emphasizing the critical importance of data protection for the SB platform, the SB@SecurityComponent includes mechanisms to fully comply with GDPR provisions, even for data generated and transmitted by HoloBalance and Smart4Health devices.

Lastly, the deployment and operation of the Security and Privacy Assurance Platform within SMART BEAR are thoroughly documented.

1.3 Structure of the document

This deliverable provides detailed insight into the new requirements currently supported by the SB@SecurityComponent and highlights the differences from the second iteration (as outlined in D5.4). The structure of the document is as follows:

- An executive summary provides a concise overview of the deliverable.
- Section 1 offers a general overview, outlining the goal and structure of the document in the introduction.
- Section 2 presents the newly requested requirements supported in alignment with the Privacy by Design principle.
- Lastly, section 3 discusses the continuous security assurance solution utilizing the Security and Privacy Assurance Platform within SMART BEAR.

The deliverable concludes by noting that the improvements presented were prompted by the current utilization of the SB platform.

2 SB Security Component Updates

2.1 Migration to Kubernetes

The migration from Docker to Kubernetes represents a strategic shift in the orchestration and deployment of containerized applications. Kubernetes, known for its robust container orchestration capabilities, emerges as a natural choice to address the complexities of managing and scaling containerized applications. This paradigm shift from Docker to Kubernetes signifies a pivotal transition in optimizing containerized workloads, enhancing scalability, and streamlining the management of microservices architecture.

The migration process brings forth a set of challenges and considerations that the project had to navigate to ensure a seamless transition. One key challenge is reconciling the fundamental differences between Docker and Kubernetes in terms of orchestration, networking, and resource management. Organizations must evaluate and modify existing Docker-compose configurations to align with Kubernetes' declarative approach. Additionally, considerations around persistent storage, service discovery, and load balancing must be readdressed to harness the full potential of Kubernetes.

For the Security Component, deployment to the Kubernetes production environment adhered to the integration guidelines outlined in Deliverable D6.5. A continuous integration and continuous deployment (CI/CD) pipeline, defined within the GitLab /SMART BEAR project (Figure 1), streamlined the deployment process for desired versions to the production environment PROD.

The configuration of the CI/CD Pipeline includes a block of jobs grouped under the deploy-prod stage (and Prod label environment), prepared to execute necessary commands against the PROD cluster. Various CI/CD variables ($K8S_SERVER_PROD, $K8S_TOKEN_PROD, and $K8S_CA_CERT_PROD) have been configured with a proper (and restricted) sb-gitlab-cicd-sa ServiceAccount to enable deployment operations.

It's important to note that secrets are not generated automatically and should be manually maintained as a security measure.

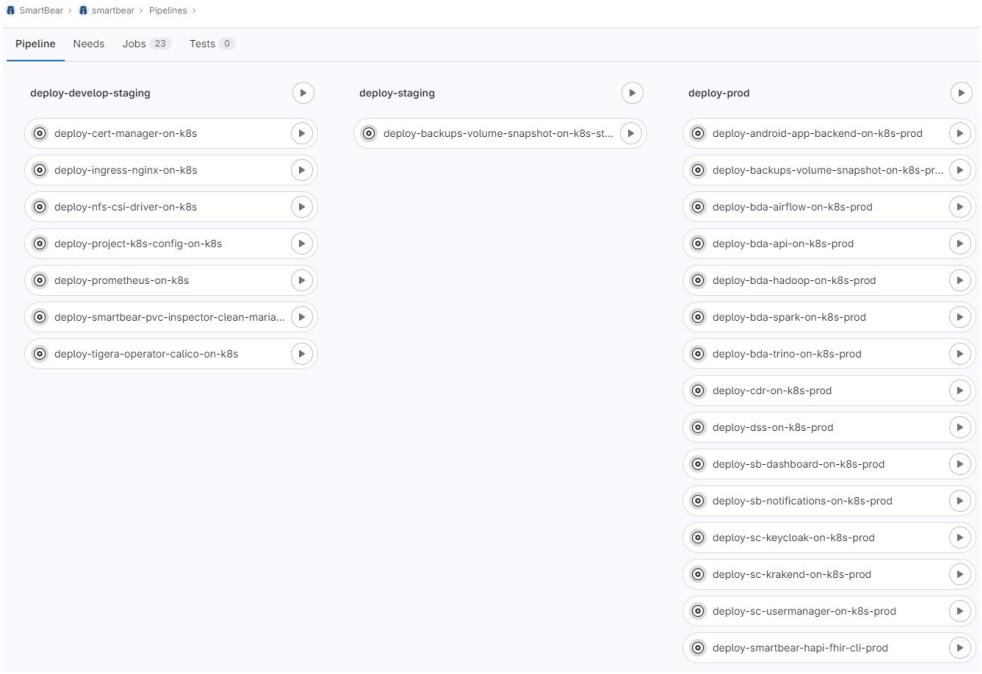

Figure 1: Deployment to PROD environment (deploy-prod pipeline stage).

By default, the version of the components to deploy is the one configured under the deployment configuration under the `master`/`main` branch. If instead, a specific version defined in a TAG should be used, the following variables must be provided to the pipeline *job*. For the Security Component, the following variables have been created:

- Keycloak: `VERSION_SC_KEYCLOAK`
- KrakenD: `VERSION_SC_KRANKEND`
- UserManager: `VERSION_SC_USERMANAGER`

More details about the migration to Kubernetes for the Security Component can be found in D6.5.

2.1.1 Data migration and validation

The Security Component comprises the KeyCloak, KrakenD, and UserManager submodules. While two of these necessitate data migration, the KrakenD API Gateway submodule does not require persistence as its configuration is already integrated into the operational tool. Regarding KeyCloak, the process involves exporting the SMART BEAR Realm configuration from the source instance and importing it into the final production instance. This process entails manually comparing platform users, assigned roles, and regenerating keys to ensure consistency. Basic functionality is validated through client authentication.

As for the UserManager subcomponent, meticulous coordination is crucial for validation to ensure alignment with the list of participants contained in the Clinical Data Repository of SMART BEAR.

The data migration of user management coincided with the transition from WSO2 to Keycloak. This involved migrating all tables and entries into a new database with appropriate mapping, presenting several challenges that needed to be managed such as:

- Correctly transfer the schema
- Correctly transfer the entries
- WSO2 has its own tables, and *UserManager* was using some attributes like the User Id. Keycloak uses other values for those attributes, so a translation/mapping had to be made.

For the migration of the database, the tool *DataGrip* (from *Jetbrains*[1]) was used to perform those operations. For verification, the tool data-diff was used (*Figure 2*).

```
16:35:50 INFO     Diffing using columns: key=('id',) update=None extra=('deleted_at', 'status', 'scope', 'updated_at', 'created_at',
                  'participant_id', 'identity_id').
         INFO     Using algorithm 'hashdiff'.
         WARNING  [PostgreSQL] Column 'identity_id' of type 'Text()' has no compatibility handling. If encoding/formatting differs
                  between databases, it may result in false positives.
         WARNING  [PostgreSQL] Column 'identity_id' of type 'Text()' has no compatibility handling. If encoding/formatting differs
                  between databases, it may result in false positives.
         INFO     Diffing segments at key-range: (1)..(77). size: table1 <= 76, table2 <= 76
16:35:51 INFO     Diff found 0 different rows.
         INFO     Duration: 1.23 seconds.
```

Figure 2: Sample data-diff tool for the validation of the 'identities' records.

2.1.1.1 Users

A user in WSO2 is mapped with an ID and a user in Keycloak is mapped with a (different) ID forcing the update to monitor the mapping of the old IDs to the new ones.

[1] JetBrains Developer tools (*https://www.jetbrains.com*)

Suppose we have a user **User1** with **ID** 12345-12345-12345 in the WSO2 system as shown in Table 1.

First_name	Last_name	Created_by
John	Doe	12345-12345-12345

Table 1: Sample record of participant in WSO2

The same user (User1) will have a different ID in Keycloak. For our example, we will use the **ID** 6789-6789-6789

If the table "participants" is migrated as it is, it will point to the ID of WSO2 (created_by column), which is wrong. Therefore, we have to translate WSO2 ID 12345-12345-12345 to Keycloak ID 6789-6789-6789, so the new table "participants" will look like the data in Table 2.

First_name	Last_name	Created_by
John	Doe	12345-12345-12345

Table 2: Sample record of participant in Keycloak

This means that if we compare the 2 tables, we will find differences (on the created_by column), even though in reality they refer to the same user.

When comparing the tables, we **ignore** the created_by property, to remove false positives. During the migration procedure, the "participants" table had 108 records (102 patients and 6 clinicians). WSO2 database contains Users ID according to WSO2 and the Keycloak database contains the Users ID according to Keycloak. In general, it isn't safe to modify the WSO2/Keycloak database manually.

```
15:43:06 INFO     Diffing using columns: key=('id',) update=None extra=('updated_at', 'organisation', 'participant_concent_data',
                  'sb_email', 'recruitment', 'internal_id', 'dropped_at', 'created_at', 'pilot', 'deleted_at').
         INFO     Using algorithm 'hashdiff'.
         WARNING  [PostgreSQL] Column 'sb_email' of type 'Text()' has no compatibility handling. If encoding/formatting differs
                  between databases, it may result in false positives.
         WARNING  [PostgreSQL] Column 'sb_email' of type 'Text()' has no compatibility handling. If encoding/formatting differs
                  between databases, it may result in false positives.
         INFO     Diffing segments at key-range: (1)..(102). size: table1 <= 101, table2 <= 101
15:43:07 INFO     Diff found 0 different rows.
         INFO     Duration: 1.59 seconds.
```

Figure 3: Migration, comparison of participants table.

By using the data-diff tool (Figure 3), we can see there are no differences. The next step is to check if the UUIDs have been mapped correctly. A manual comparison of the UUIDs was performed to identify if the mapping was correct. The following list (Figure 4) shows a part of UUIDs and its mapping. IDs of users in WSO2 can be found to the left and to the right we can see the IDs of users in Keycloak. All the IDs have been mapped correctly.

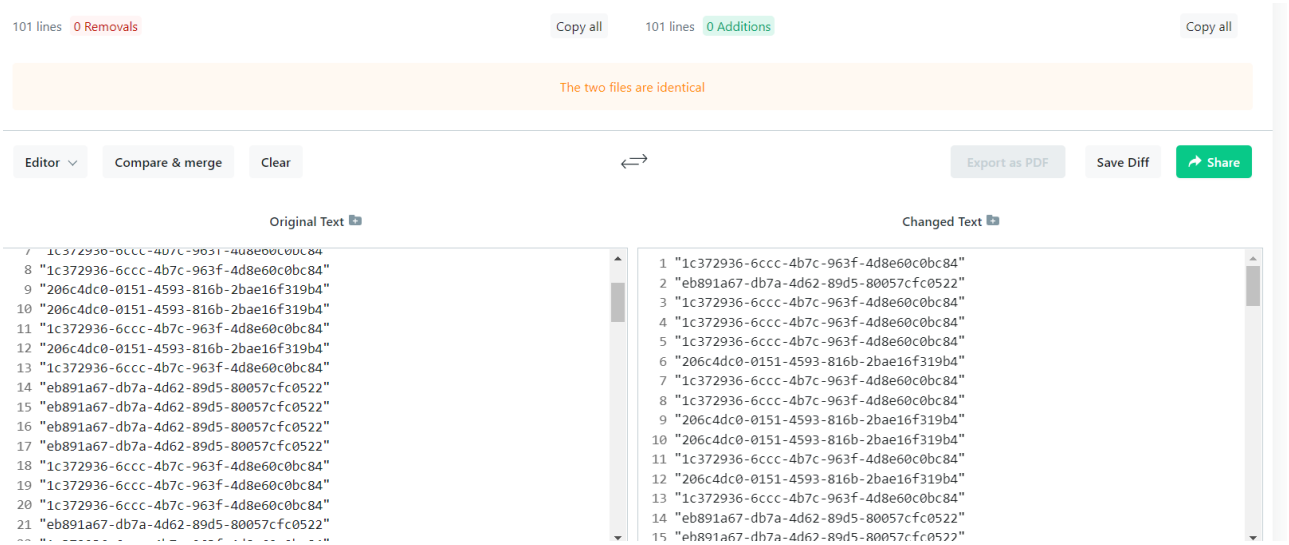

Figure 4: Migration, Mapping participant UUIDs from WSO2 to Keycloak

2.1.1.2 Caregivers

For caregivers, there are only 3 records, so it is easy to compare them manually but we can also use the comparison tool and we can see that the two tables are identical (Figure 5).

```
11:15:14 INFO     Diffing using columns: key=('id',) update=None extra=('participant_id', 'updated_at',
                  'deleted_at', 'email', 'created_at').
         INFO     Using algorithm 'hashdiff'.
11:16:55 WARNING  [PostgreSQL] Column 'email' of type 'Text()' has no compatibility handling. If
                  encoding/formatting differs between databases, it may result in false
                  positives.
         WARNING  [PostgreSQL] Column 'email' of type 'Text()' has no compatibility handling. If
                  encoding/formatting differs between databases, it may result in false
                  positives.
         INFO     Diffing segments at key-range: (1)..(4). size: table1 <= 3, table2 <= 3
11:18:37 INFO     Diff found 0 different rows.
```

Figure 5: Migration, caregivers table.

2.1.1.3 Clinical Case Managers (CCMs)

For CCM, we will use the same method to compare. After comparing all the records of the table (except column ccm_id), we see that there are no differences (Figure 6).

```
14:39:05 INFO     Diffing using columns: key=('id',) update=None extra=('accepted', 'participant_id', 'deleted_at', 'created_at',
                  'updated_at').
         INFO     Using algorithm 'hashdiff'.
         INFO     Diffing segments at key-range: (1)..(304). size: table1 <= 303, table2 <= 303
         INFO     Diff found 0 different rows.
         INFO     Duration: 2.92 seconds.
```

Figure 6: Migration, ccm table.

ccm_id, is a UUID and we had to follow the same procedure as for the participant's table. Therefore, we need to use the same rules and replace WSO2 UUIDs with new ones. As we can see, the mapping is correct (Figure 7).

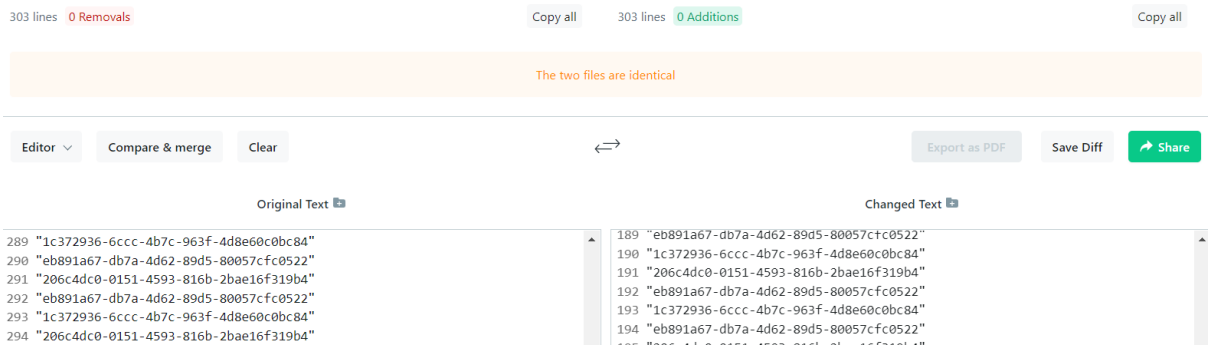

Figure 7: Migration, Mapping ccm UUIDs from WSO2 to KeyCloak

2.1.1.4 Contact phones

For contact phones we have 142 records and all of them are identical (Figure 8).

```
14:47:55 INFO     Diffing using columns: key=('id',) update=None extra=('updated_at', 'participant_id', 'created_at', 'deleted_at', 'iso2',
                  'phone', 'dial_code').
         INFO     Using algorithm 'hashdiff'.
         INFO     Diffing segments at key-range: (1)..(143). size: table1 <= 142, table2 <= 142
14:47:56 INFO     Diff found 0 different rows.
         INFO     Duration: 2.08 seconds.
```

Figure 8: Migration, contact phones

2.1.1.5 Device metrics

Similarly, we use data-diff for the device metrics table and we can see there are no differences (Figure 9).

```
16:03:11 INFO     Diffing using columns: key=('id',) update=None extra=('transmission_date', 'valid', 'device_id').
         INFO     Using algorithm 'hashdiff'.
         INFO     Diffing segments at key-range: (1)..(67740). size: table1 <= 67739, table2 <= 67739
         INFO     . Diffing segment 1/32, key-range: (1)..(2117), size <= 67739
         INFO     . Diffing segment 2/32, key-range: (2117)..(4233), size <= 67739
         INFO     . Diffing segment 3/32, key-range: (4233)..(6349), size <= 67739
         INFO     . Diffing segment 4/32, key-range: (6349)..(8465), size <= 67739
         INFO     . Diffing segment 5/32, key-range: (8465)..(10581), size <= 67739
         INFO     . Diffing segment 6/32, key-range: (10581)..(12697), size <= 67739
16:03:12 INFO     . Diffing segment 7/32, key-range: (12697)..(14813), size <= 67739
         INFO     . Diffing segment 8/32, key-range: (14813)..(16929), size <= 67739
         INFO     . Diffing segment 9/32, key-range: (16929)..(19045), size <= 67739
         INFO     . Diffing segment 10/32, key-range: (19045)..(21161), size <= 67739
         INFO     . Diffing segment 11/32, key-range: (21161)..(23277), size <= 67739
         INFO     . Diffing segment 12/32, key-range: (23277)..(25393), size <= 67739
         INFO     . Diffing segment 13/32, key-range: (25393)..(27509), size <= 67739
         INFO     . Diffing segment 14/32, key-range: (27509)..(29625), size <= 67739
         INFO     . Diffing segment 15/32, key-range: (29625)..(31741), size <= 67739
16:03:13 INFO     . Diffing segment 16/32, key-range: (31741)..(33857), size <= 67739
         INFO     . Diffing segment 17/32, key-range: (33857)..(35973), size <= 67739
         INFO     . Diffing segment 18/32, key-range: (35973)..(38089), size <= 67739
         INFO     . Diffing segment 19/32, key-range: (38089)..(40205), size <= 67739
         INFO     . Diffing segment 20/32, key-range: (40205)..(42321), size <= 67739
         INFO     . Diffing segment 21/32, key-range: (42321)..(44437), size <= 67739
         INFO     . Diffing segment 22/32, key-range: (44437)..(46553), size <= 67739
         INFO     . Diffing segment 23/32, key-range: (46553)..(48669), size <= 67739
         INFO     . Diffing segment 24/32, key-range: (48669)..(50785), size <= 67739
         INFO     . Diffing segment 25/32, key-range: (50785)..(52901), size <= 67739
         INFO     . Diffing segment 26/32, key-range: (52901)..(55017), size <= 67739
         INFO     . Diffing segment 27/32, key-range: (55017)..(57133), size <= 67739
         INFO     . Diffing segment 28/32, key-range: (57133)..(59249), size <= 67739
16:03:14 INFO     . Diffing segment 29/32, key-range: (59249)..(61365), size <= 67739
         INFO     . Diffing segment 30/32, key-range: (61365)..(63481), size <= 67739
         INFO     . Diffing segment 31/32, key-range: (63481)..(65597), size <= 67739
         INFO     . Diffing segment 32/32, key-range: (65597)..(67740), size <= 67739
         INFO     Duration: 4.50 seconds.
```

Figure 9: Migration, device metrics.

2.1.1.6 Devices

Each device has an assigner ID which is unique and different from Keycloak to WSO2, so we will ignore it.

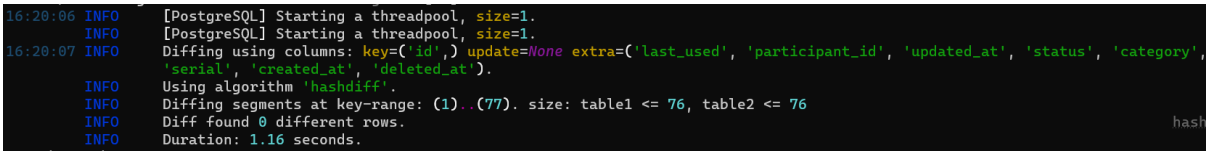

Figure 10: Migration, devices.

All rows apart from assigned IDs are identical (Figure 10). Checking the assigner IDs manually (Figure 11) we can verify that the IDs have been mapped correctly.

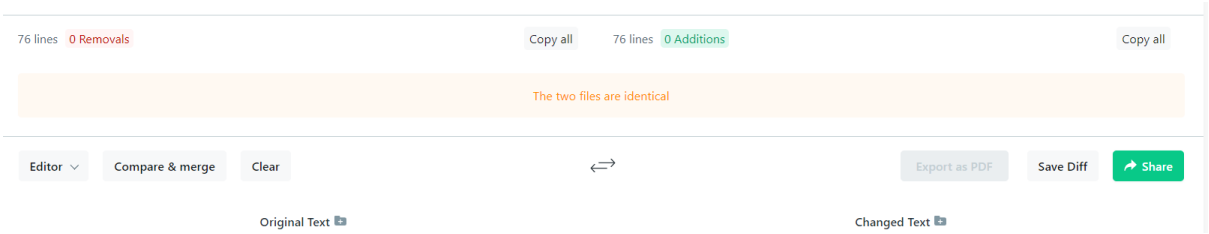

Figure 11: Migration, devices mapping.

2.1.1.7 GDPR Request Files

For GDPR Request Files, we had zero entries, so the migration created an empty table as shown in Figure 12.

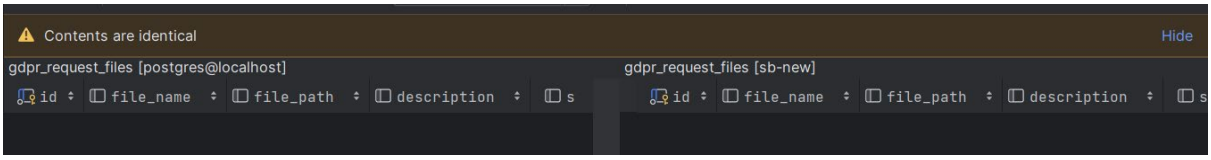

Figure 12: Migration, GDPR request files.

2.1.1.8 GDPR Request

For the GDPR Request, we had zero entries, so the migration created an empty table as shown in Figure 13. It worth mentioning that GDPR Request records store data about the GDPR request per se, while the GDPR Request Files are records about the uploaded files.

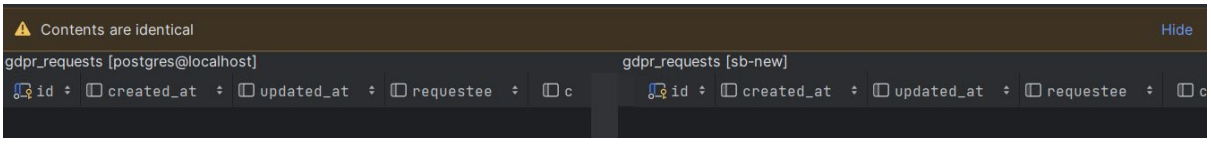

Figure 13: Migration, GDPR request.

2.1.1.9 Notifications

For notifications, we have 316 records. Using the data-diff tool we can verify that the records after the migration are identical (Figure 14).

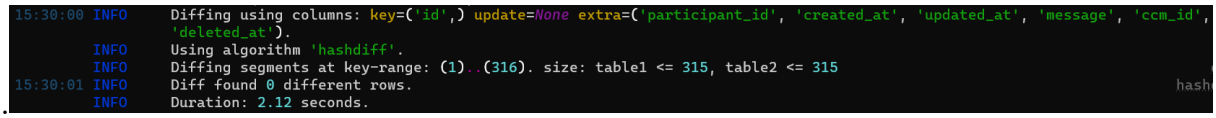

Figure 14: Migration, notifications.

2.1.1.10 Identities

For the identities table, we have UUIDs and we have to handle them as previously. From Figure 15 we can see that the tables are the same if we ignore the UUIDs. Checking the UUIDs manually (Figure 16), we can verify that have been translated correctly.

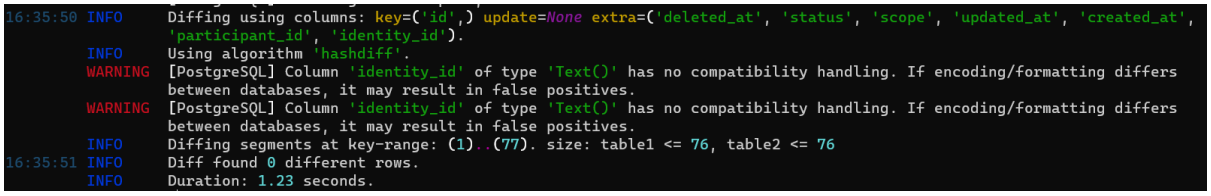

Figure 15: Migration, identities.

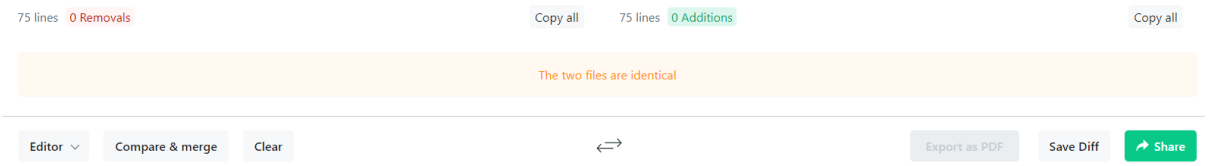

Figure 16: Migration, mapping identities.

In summary, the performed tests verify the correct transfer of the data. Specifically, the transferred records per DB table are shown in Table 3.

Table	Number of Records	Difference
Users	102 patients and 6 clinicians	The IDs transformed from WSO2 UUIDs to Keycloak UUIDs. Comparing the data and the mapped UUIDs there are no differences during the migration (data-diff tool was used).
Caregivers	3	data-diff result 0
CCM	303	data-diff result 0
Contact phones	142	data-diff result 0
Device metrics	67740	data-diff result 0
Devices	76	data-diff result 0

GDPR request	0	data-diff result 0
GRPR request files	0	data-diff result 0
Notification	316	data-diff result 0
Identities	77	data-diff result 0

Table 3: Verification of data transfer per table in the DB

2.2 Keycloak configuration

During the deployment and setup of a new Keycloak instance, the following steps are required for the integration with the SB platform.

2.2.1 Keycloak Realm creation

Create the SMART BEAR realm by clicking the dropdown menu on the sidebar and selecting "Create Realm". In the realm name enter "`smartbear`" and click "Create" (*Figure 17*).

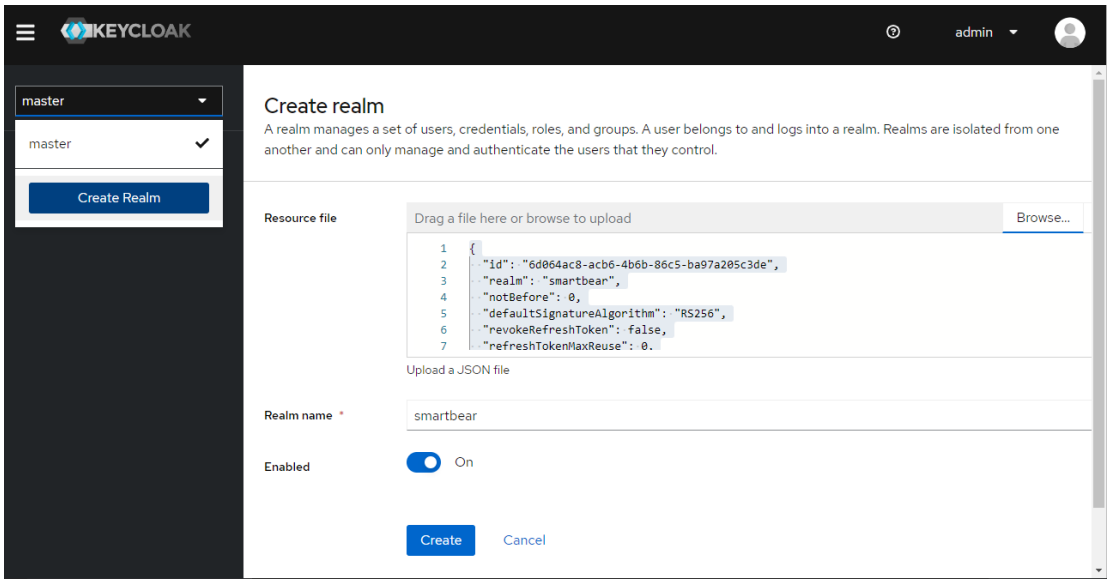

Figure 17: Keycloak realm creation from previously exported one.

2.2.2 Add realm roles

Click on the "Realm Roles" in the sidebar, and add the following roles:

- **Auditor**: accesses the SMART BEAR platform to audit its behaviour, especially for monitoring the data processing procedures in terms of compliance with guidelines and regulations.
- **Caregiver**: provides care for a person who needs extra help. This could mean a family caregiver, a respite caregiver, a home caregiver, or a primary caregiver, to name but a few. In the context of elderly care, this job title typically refers to a private-duty home caregiver or senior caregiver.
- **CCM**: a health care professional that works as a primary caregiver of a patient in a hospital, skilled nursing facility, clinic, or patient's home. A clinician diagnoses and treats patients. The Case Manager uses the SMART BEAR platform to increase the level of evidence-based practices and care plans.

- **DataScientist**: a professional responsible for collecting, analysing and interpreting the overall amount of data collected by the SMART BEAR Cloud. Data science plays an important role in the effective exploitation of the data collected by the SMART BEAR platform.
- **PolicyMaker**: exploits the statistical report and analytics resulting from the SMART BEAR Cloud to create policies and plans, especially for the good of a territory or community.
- **SysAdmin**: is responsible for the setup and reliable operation of the SMART BEAR platform. It should be able to ensure that the uptime, performance, resources, and security of the computers they manage meet the needs of the other users. The system administrator may require access to the system components and to the registers accounting for the behaviour of the SMART BEAR platform.
- **Tech**: a technician from the clinical site who assists the patients with the wearables and IoT devices.

2.2.3 Add clients

Go to "Clients" in the sidebar, click on "Create client" and add the following clients:

- **Dashboard**:
  - Name: SMART-BEAR Dashboard
  - Click on "Next". Disable "Direct Access Grants" and keep only "Standard flow" enabled.
  - Go back to the client list, click on the newly created client and fill in the following:
  - Valid Redirect URIs: http://localhost:8080/, /
  - Valid post logout redirect URIs: +
  - Web origins: +
  - Save.
- **UserManager**
  - Client ID: usermanager
  - Click on "Next". Enable "Client authentication". Disable "Standard flow" and "Direct Access Grants". Enable "Service account roles".
  - Save.
  - Go to "Users" in the sidebar. Select "service-account-usermanager". Go to the Role Mapping tab. Click on "Assign Role" and add "realm-management view-users".
  - Save.
- **HoloBalance**
  - Client ID: holobalance
  - Click on "Next". Disable "Direct Access Grants" and "Direct Access Grants". Enable "Service account roles".
  - Save.
- **Mobile**
  - Client ID: mobile
  - Click on "Next". Disable "Direct Access Grants" and "Direct Access Grants". Enable "Service account roles".
  - Save.
- **Smart4Health**
  - Client ID: smart4health
  - Click on "Next". Disable "Direct Access Grants" and "Direct Access Grants". Enable "Service account roles".
  - Save.
- **Smart4HealthBox**

- Client ID: smart4healthbox
- Click on "Next". Disable "Direct Access Grants" and "Direct Access Grants". Enable "Service account roles".
- Save.

We have to state that the synergies (HoloBalance, Smart4Health and Smart4HealthBox) are not granting access to information on the SB platform or App but merely allowing for transfers of data from the synergies to SB.

2.2.4 SMART BEAR theme

Go to "Realm Settings" in the sidebar, and click on the "Themes" tab. From the dropdown, select "SMART BEAR" for the "Login theme" and "Email theme".

2.2.5 Configure email

Go to "Realm Settings". Click on the "Email" tab. In the "From" field, enter "no-reply@smart-bear.eu". Under Connection & Authentication, enter the following:

- Host: `mail.smart-bear.eu`
- Port: `587`
- Authentication: `Enabled`
- Username: `no-reply@smart-bear.eu`
- Password: `***TBD***`

To test the connection, you need to have an e-mail to the admin account. To do this, go back to the "master" realm and edit the "admin" user.

2.2.6 Configure Token Claims

- Go to "Client scopes" in the sidebar.
- Select "Create client scope". Enter "organisation" in the name and click on "Save".
- Select the new client scope, and go to the "Mappers" tab. Click "Configure new mapper".
- Select "User attribute". Enter "organisation" in the "name", "user attribute", and "token claim name" fields. Click "Save".

2.3 KrakenD API gateway

We use KrakenD's flexible configuration, to configure the gateway. The main configuration file is krakend.tmpl which is a Go template. The config directory contains additional configuration files used by the template, including:

- The templates directory which contains templates for the endpoints, with one template created for each component.
- The partials directory which contains the configuration for Keycloak, directly injected into the endpoint templates.
- The settings directory which contains variables used to configure the component addresses in the form of <host>:<port>.

KrakenD plugins are written in Go and loaded at runtime. For SMART BEAR, plugins are located in the plugins directory and compiled according to the *Dockerfile*. The following plugins are included:

- **bda-pseudonymization**: performs pseudonymization on request and response data of the Big Data Analytics component.
- **sb-data-repo-pseudonymization**: performs pseudonymization on request and response data of the SB Data Repo component. This is the largest plugin, as it handles complex Fast Healthcare Interoperability Resources (FHIR) resources, including recursive structures.
- **mobile-notification**: sends an e-mail notification to a participant's clinical case manager.
- **http-client**: operates just before the request is sent to the backend and just after the response is received. It is used to store the X-Synergy header added to the request by a previous plugin and to add it to the response to be used by a subsequent response plugin. Otherwise, the header would be lost.

2.4 UserManager participants management

User management is handled by Keycloak. APIs for the basic operations of user management (add/edit/delete users for the SMART BEAR realm) have been created and can be used in the future to create a UI within the dashboard for user management.

To manage users from the Keycloak, one must have set up a realm and select the realm from the main menu. Then the following actions can be performed: Add, Edit, Disable.

2.4.1 Add users

- Go to "Users" in the sidebar. Click on "Add user".
- Enter user details (username, email and name)
- Click on "Create".
- Go to the newly created user.
- Click on "Attributes" and enter "organisation" in the key, and the abbreviation of the user's organisation in the value. Click "Save".
- Go to "Role Mapping". Click "Assign Role" and enter the user's role (make sure that “filter by realm roles is selected).
- Go to "Credentials". Select "Credential Reset" and select "Reset password". An email will be sent to the user with instructions on how to set their password.

2.4.2 Edit users

- Go to "Users" in the sidebar. Click on the user.
- Click on "Attributes" and update "organisation" in the key, and the abbreviation of the user's organisation in the value. Click "Save".
- Go to "Role Mapping". Click "Assign Role" and update the user's role (make sure that “filter by realm roles is selected).
- Go to "Credentials". Select "Credential Reset" and select "Reset password". Optionally, if you click the ‘send email’ an email will be sent to the user with instructions on how to set their password.

2.4.3 Disable account

- Go to "Users" in the sidebar. Click on the user.
- Go to "Credentials". Select "Credential Reset" and select "Delete account".

A step-by-step user manual for the creation of end users in Keycloak can be found in the deliverable D5.5.

2.5 New requirements

2.5.1 Patient management: Device inactivity

To support the device inactivity use case, where the dashboard users will be notified for the days of non-transmission of data from the devices of a patient, new entries had to be supported by the patient management system and the dashboard. The patient management system serves to the dashboard the latest date that a device transmitted data to the SB backend and the dashboard visualizes that information accordingly as shown in Figure 18.

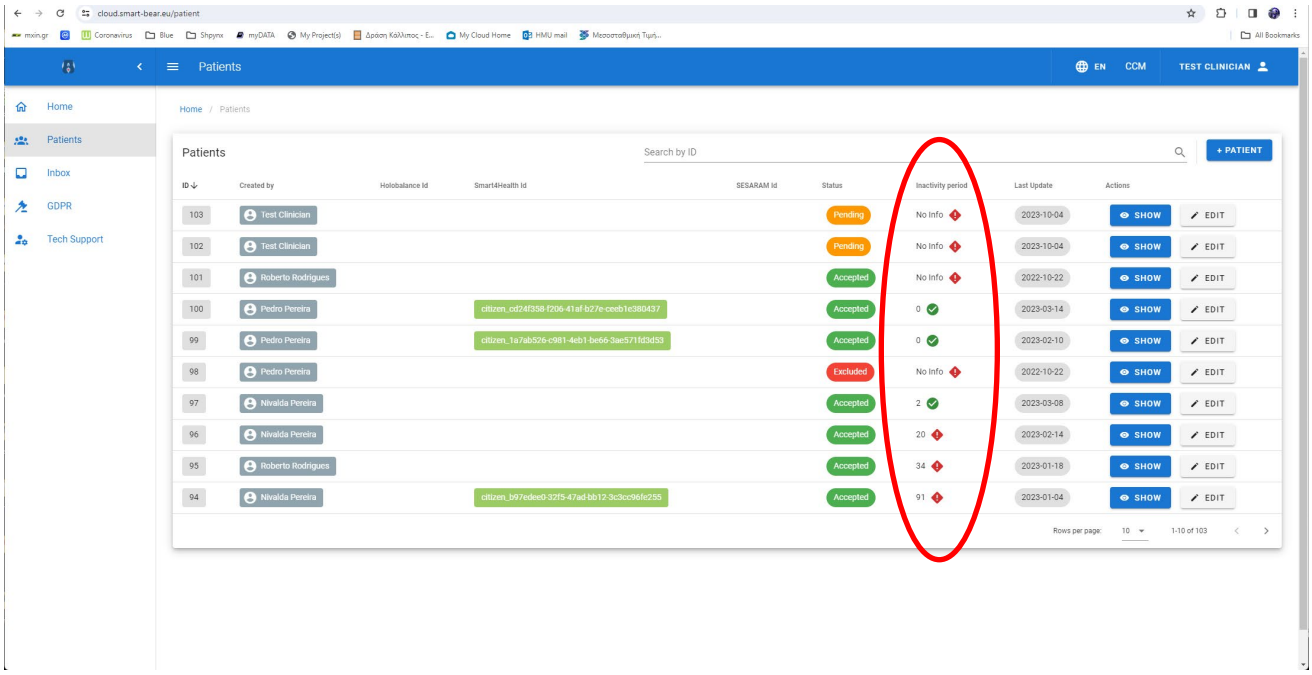

Figure 18: Device inactivity at dashboard.

3 SB Continuous Security Assurance

3.1 Introduction

In parallel to the integration of appropriate mechanisms for securing the SB platform's operations *a priori*, adherence to the "Privacy by Design" entails the operation of continuous monitoring mechanisms and processes. This ensures the security (confidentiality, integrity, and availability) of all assets (including hardware, software, data, personnel, and networking) and safeguards the security and privacy of all data handled by the SB platform in various states (at rest, in transit, and during processing). Despite the effectiveness of implemented techniques or the introduction of new methods to protect these assets, akin to a "defensive perimeter," it is crucial to concurrently implement a post facto methodology for conducting continuous security and privacy assessments. This is essential to counter ongoing security threats, human errors, exploitable technical weaknesses, and other lesser-known vulnerabilities arising from the escalating complexity and interconnectivity of information and communication technologies (ICTs).

Within the SB project framework, the realization of Objective 3: Security, Privacy, and Trustworthiness of the platform involves adopting a continuous security and privacy assurance approach. This approach incorporates the utilization of usage metadata (e.g., log files created by existing operating systems/RDBMs/proprietary or open-source software, activity data generated from data sniffers) probes for ongoing monitoring and dynamic testing of the platform. The goal is to generate evidence supporting the continuous assessment of the platform's security and privacy provisions, rendering it transparent, auditable, and trustworthy (KPI-14 to KPI-18).

3.2 SB Security and Privacy Assurance Platform

While the state-of-the-art techniques and processes implemented in SB effectively address all security, privacy, and regulatory compliance requirements identified during the comprehensive risk and compliance assessment conducted in the project's early stages, it is crucial not to overlook the understanding that GDPR is an ongoing process rather than a specific endpoint to achieve. Factors such as human errors, technical bugs, and vulnerabilities in programming languages could potentially lead to data breaches, even in the most secure information systems. To address this aspect, achieving real-time operational compliance of the SB platform is achieved with the integration of the SB Security and Privacy Assurance Platform (SPAP). This integration facilitates security and privacy assessments based on vulnerability analysis, penetration testing, and continuous monitoring of the SB@Cloud assets, providing evidence-based assurance.

SPAP suite is responsible for monitoring, testing, and assessing the security (& privacy, if needed) posture of the protected organisation(s) and their assets, in a real-time, continuous manner. Several built-in security assessments addressing the Confidentiality – Integrity – Availability (CIA) principles (via custom metrics that can be tailored concerning the platform's components) can be utilized, leveraging an evidence-based approach, to provide security assurance assessments with certifiable results. The SPAP is scheduled to perform security and privacy assessments that involve threat and vulnerability analysis, static analysis, penetration testing and continuous runtime monitoring, to provide a comprehensive analysis of the security and privacy posture of a system. It will audit critical components and processes of the SB@Cloud (e.g., SB@Dashboard, SB@SecurityComponent, SB@Repository), through an Event Captor Module developed for the SB platform, while leveraging monitoring mechanisms developed in the context of the project. More information about the SB final architecture showing the interdependencies among the main building blocks and SPAP can be found in D5.4.

A high-level view of SPAP's internal architecture is provided in Figure 19, and it consists of the following components: The core platform, the asset-based vulnerability assessment, the dynamic tester, the event reasoning toolkit (EVEREST), the event captors and the security component (Keycloak and KrakenD).

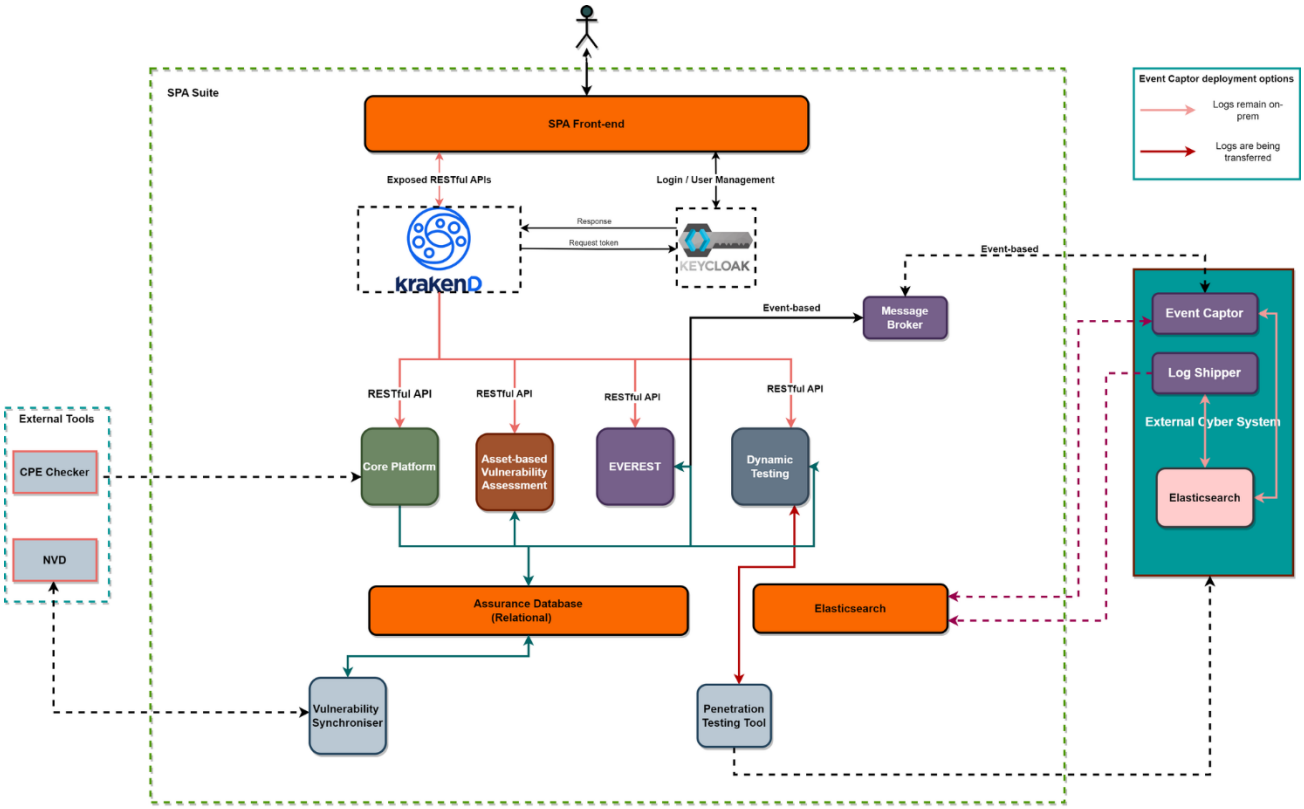

Figure 19: The SB Security and Privacy Assurance Platform's Architecture

3.2.1 Core Platform

The component is responsible for managing the asset and assessment model of an organisation. This component includes (a) the Asset Loader, and (b) the Assessment Loader. The former is responsible for maintaining the cyber system's asset model for the target organization. This model includes the organisation's assets, security properties for these assets, relations between assets in the model, and the security controls that protect the assets. This component provides the following ways to insert one or more assets. First, the user can utilise the SPA Suites' front end to insert one or more assets, through wizards. Secondly, the user can fill out a pre-structured Excel, and either utilise the front end to upload it or the exposed API. Lastly, the user can provide an SBOM (either in CycloneDX or SPDX format) and, again, make use of the platform's front end or the corresponding REST API to insert the assets defined within it, in the system.

As for the Assessment Loader, it is the component responsible for handling the available assessments that the SPA Suite offers. More specifically, it provides means to manage the assessment (a) criteria, (b) profiles, (c) results, and (d) model executions.

3.2.2 Asset-based vulnerability assessment

The component is responsible for performing a passive vulnerability assessment to identify known vulnerabilities of assets defined within an organisation's asset model. This component communicates with the core platform, receives the Common Platform Enumeration (CPE) of the assets within the

assessment execution and then retrieves the relevant Common Vulnerabilities and Exposures entries, by searching in a local copy of the National Vulnerability Database (a U.S. government repository of standards-based vulnerability management data, maintained by the National Institute of Standards and Technology). This copy is continuously updated using an in-house component fetching the latest known CVEs. Following the execution of a vulnerability assessment, the tool will report known vulnerabilities for those assets that have a valid CPE.

3.2.3 Dynamic Tester

The component is responsible for initiating dynamic testing assessments (i.e., penetration testing or active vulnerability assessment) for a target organisation. Such assessments can only be executed if the Target of Evaluation (ToE) document is agreed upon and signed by the organisation. The dynamic tester can also identify assets that are not included in the used asset model. This module is responsible for parsing the report, as exported from the corresponding tool, creating a new assessment model execution, with the corresponding results, and adding the identified assets to the organisation's asset model.

3.2.4 EVent REaSoning Toolkit (EVEREST)

The component responsible for monitoring the target organisation, for potential issues within its cyber system. In its monitoring capacity, EVEREST possesses a multifaceted approach. It surveils the cyber system across a spectrum of crucial aspects, such as network traffic, potential threats from both internal and external sources, misconfiguration, and not properly installed security controls. By continuously evaluating events against defined rules expressed in Event Calculus and Drools, EVEREST can detect anomalies, deviations, and potential risks within the system. EVEREST works with the Event Captors to fetch the raw events from the cyber system.

3.2.5 Event Captors

An Event Captor is a tool that, based on a specification set by EVEREST, aggregates log and event information from the targeted infrastructure, and encapsulates it in a specific format that can be consumed by the EVEREST model. Logs and events can be collected in two modes. The former mode is based on the ELK solution. More specifically, Elasticsearch[2] and some lightweight shippers (namely Beat[3]) are utilised to forward and centralize log data. The latter makes use of SPHYNX's Native Event Captors, i.e., captors that cannot utilise the logging capabilities of the ELK stack. The needed Event Captors are initiated through EVEREST.

3.2.6 SPAP Security Component

The component that provides Identity and Access Management capabilities, as well as, an API Gateway for exposing RESTful APIs (when needed). This component is comprised of two main components:

- Keycloak, which is used for identity and access management, and
- KrakenD, the API Gateway that handles authorization in conjunction with Keycloak.

[2] https://www.elastic.co/elasticsearch

[3] https://www.elastic.co/beats/

More specifically, Keycloak is an open-source identity and access management platform that provides a single point of access for modern applications, APIs, and microservices. It offers features such as single sign-on, identity brokering, and social login, as well as robust user management and authentication capabilities. Following, KrakenD is a lightweight, high-performance API Gateway that helps expose internal and external microservices to the world, while keeping the complexity and routing logic out of core services. It allows one to easily build and deploy API Gateway configurations using a simple, declarative configuration file, and provides features such as rate limiting, circuit breaking, and caching to help manage the performance and reliability of your APIs.

3.3 Deployment of SB Security and Privacy Assurance Platform

The SB Security and Privacy Assurance Platform has been effectively implemented in both the staging and production environments of SB. Recognizing its independence and resource-intensive nature, it was determined that the platform would be deployed within a virtual machine with access to the logs from both the staging and production environments for assessment purposes.

Presently, the Privacy Assurance Platform facilitates availability assessments for the fundamental components of SB, including the dashboard, Keycloak, and databases. Additionally, it conducts confidentiality assessments for users accessing nodes within the Kubernetes cluster.

Deliverable D5.5 provides the steps that the SPAP user has to follow to start an assessment. For demonstration purposes, the availability of SB@Dashboard was used. Figure 20 shows the results page of the SPAP platform for the availability assessment of SB@Dashboard.

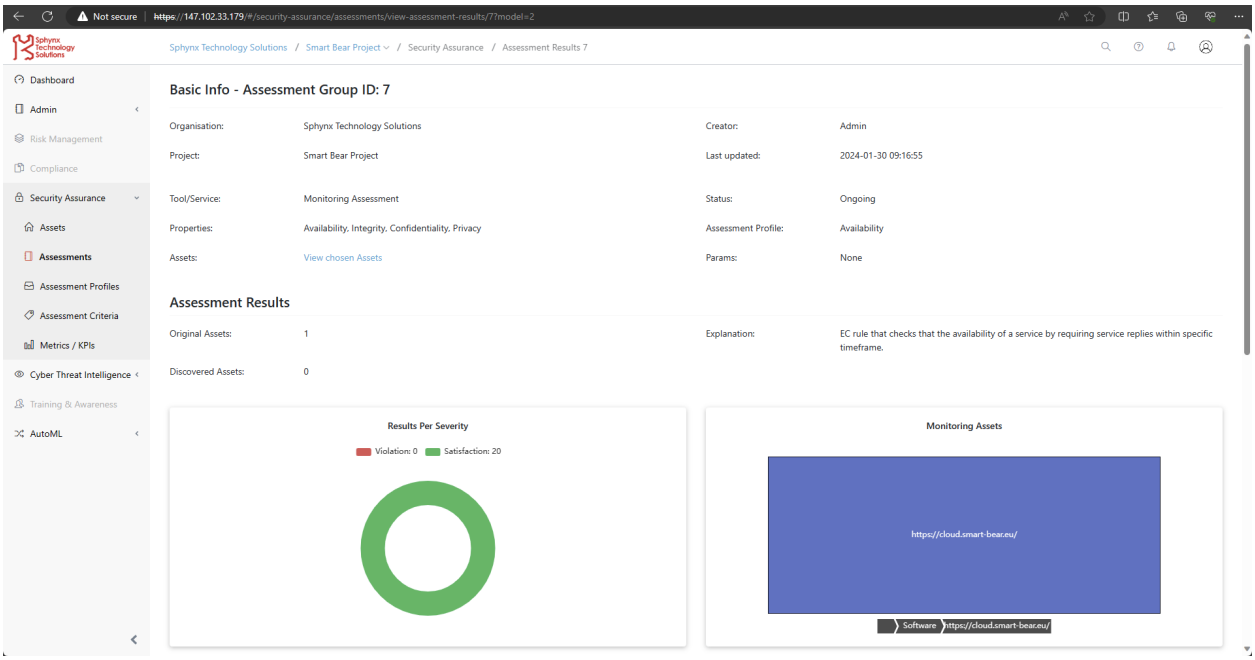

Figure 20: SPAP Assessment results dashboard.

Pending action: t present, the SPAP exclusively supports availability and confidentiality assessments. However, in the forthcoming period, it will broaden its capabilities to encompass GDPR assessment rules and User and Entity Behavior Analytics (UEBA), specifically designed for end users accessing the SB@Dashboard. Conducting the UEBA assessment will necessitate a significant amount of login logs data to effectively train a machine learning UEBA model. These updates and additions will be detailed in the next deliverable (D5.7).

The SPAP serves as a comprehensive security assurance platform, offering continuous monitoring for SB@Cloud assets. With its capabilities, SPAP can accommodate various types of assessments and allows security experts to develop new and customized assessments dynamically. The management of the platform is overseen by STS. It's crucial to emphasize that SPAP doesn't function as an incident response tool for SB, it serves primarily as a monitoring tool to ensure the smooth operation of the SB platform. In the event of anomaly detection, STS will promptly notify technical partners within 24 hours to conduct further investigation into the incident.

4 Conclusion

This document presents an enhanced version of the privacy and security mechanisms implemented within the SB@Cloud platform. It offers comprehensive descriptions of the developed assets, including their rationale and interoperability considerations. In response to updated requirements derived from thorough evaluations conducted at pilot sites and during the Pilot of Pilots in Madeira, as well as feedback gathered from end users, the new functionality introduces mechanisms and services tailored to support SMART BEAR pilots.

The primary objective of the Security and Privacy Assessments Platform is to provide evidence for evaluating the project's adherence to research ethics principles and corresponding legal obligations, as outlined by the European Commission. Any identified weaknesses stemming from the conducted security assessments will be addressed promptly through appropriate response actions.

Apart from the operation of the Security and Privacy Assessments Platform and the additional functionality and improvements of the security components, one key update was the migration from Docker to Kubernetes. Such a migration is not merely a technical shift but a strategic move that unlocks a host of benefits for the project. Kubernetes' advanced orchestration capabilities empower organizations to efficiently scale their applications, automate deployment workflows, and enhance resource utilization. This transition ultimately positions SMART BEAR to embrace a more agile, scalable, and resilient container orchestration paradigm in the ever-evolving landscape of modern application deployment.