COMPACT

CYBERSECURITY FOR PUBLIC ADMINISTRATIONS

D4.5 COMPACT Integration (v1)

Work Package:	WP4		
Lead partner:	Grupo S21sec Gestión S.A. (S21SEC)		
Author(s):	Ion Larrañaga, Iker Muñiz , Sheila Pérez (S21SEC)		
Due date:	31st of October 2018		
Version number:	1.0	**Status:**	Final

Grant Agreement N°:	740712
Project Acronym:	COMPACT
Project Title:	Competitive Methods to protect local Public Administration from Cyber security Threats
Call identifier:	H2020-DS-2016-2017
Instrument:	IA
Thematic Priority:	Secure societies – Protecting freedom and security of Europe and its citizens
Start date of the project:	1st of May 2017
Duration:	30 months

Dissemination Level	
PU: Public	✓
PP: Restricted to other programme participants (including the Commission)	
RE: Restricted to a group specified by the consortium (including the Commission)	
CO: Confidential, only for members of the consortium (including the Commission)	

Horizon 2020 European Union Funding for Research & Innovation

Project co-funded by the European Commission under the Horizon 2020 Programme.

Revision History

Revision	Date	Who	Description
0.1	10/10/2018	S21SEC	First draft of the document
0.2	17/10/2018	S21SEC	Minor corrections
0.3	21/11/2018	S21SEC	Adaptation to CINI review
0.4	07/12/2018	S21SEC	Adaptation to ENG review
1.0	08/12/2018	S21SEC	Final version

Quality Control

Role	Date	Who	Approved/Comment
Reviewer	05/12/2018	ENG	Approved with minor comments
Reviewer	15/11/2018	CINI	Approved with minor comments

Disclaimer:

Table of Contents

List of Figures

List of Codes

Definitions and acronyms

WAR	Web ARchive
SQL	Structured Query Language
GUI	Graphic User Interface

Executive summary

This document aims to describe the integration of the different tools that implement services offered by the COMPACT platform: Risk Assessment, Awareness and Education, Monitoring and Information Sharing services.

To complement the loosely coupled interaction among the COMPACT components, presented in *D3.2 Overall COMPACT Architecture (v1)*, the consortium decided to also introduce a centralized presentation layer, called the COMPACT Dashboard. In this platform, LPA Managers can access relevant information from the COMPACT tools, in an aggregated way. To present the overall status of the COMPACT components, the COMPACT Dashboard leverages data received through the COMPACT Communication Bus.

The main outcome of deliverable D4.5 is the first release of the COMPACT platform. This document complements the first release with useful information for organizations that will adopt the COMPACT platform. It has been divided into three sections: (1) rationale for the integration approach, (2) deployment instructions and (3) common data exchange format.

1. Introduction

The COMPACT platform includes a set of tools providing the following services: Risk Assessment, Awareness and Education Services, Monitoring Services and Information Sharing services.

As most of the technical partners involved in this project are adapting already existing components to the COMPACT needs, identified in *D2.10 User requirements and use cases (v2)*, a strong integration between different components is arduous. In addition, the end-user can be overwhelmed with the sheer number of tool reports, voluntarily or involuntarily forgetting to check some of the more complicated reports.

A solution to this issue is to develop a unified platform where COMPACT tools are loosely coupled, and end-users can access the data generated by each component at a glance, instead of accessing each tool's dedicated interface. This unified platform provides an easier interaction with the different components than a monolithic and strictly integrated system.

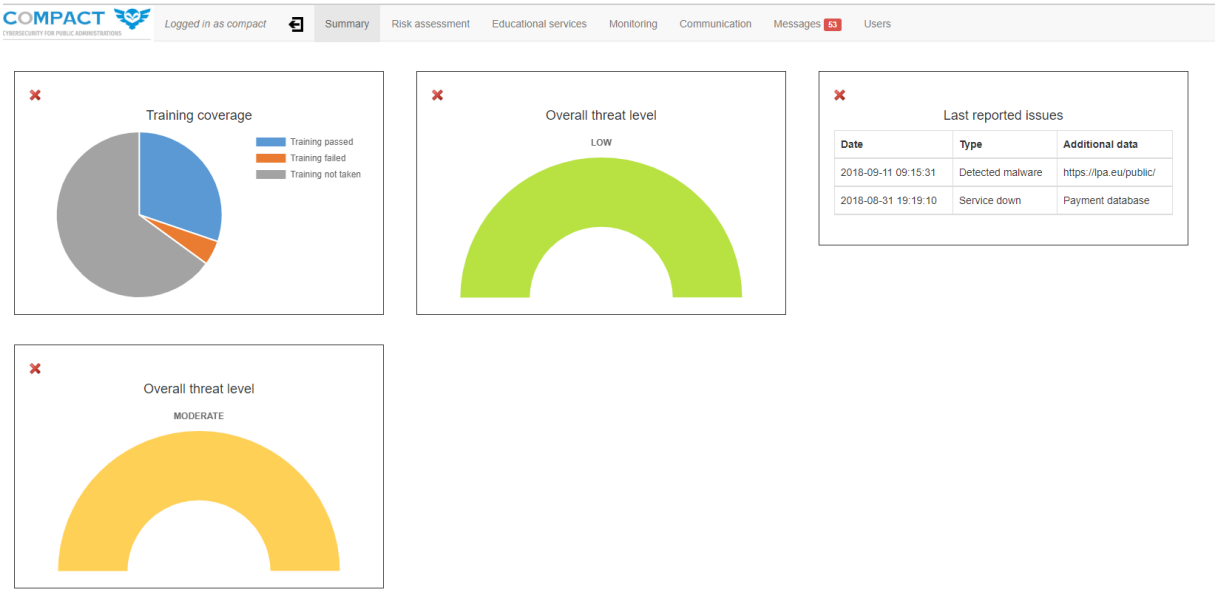

Figure 1 - COMPACT Dashboard Summary

For that reason, COMPACT Dashboard (shown in Figure 1) has been developed for COMPACT project as a unified platform, where data is obtained from the Communication Bus using Apache Kafka, thoroughly explained in deliverable *D3.3 Components Evolution Plan* [1]. In summary, any tool that wants to export data to the Dashboard can asynchronously publish information to a dedicated Apache Kafka topic (Communication Bus), as shown as Figure 2.

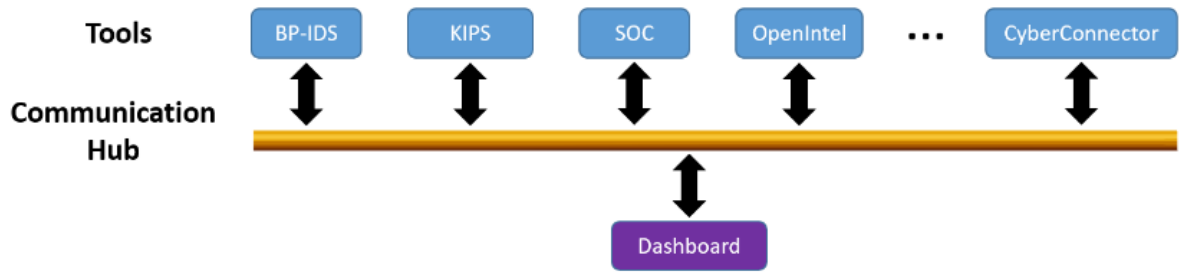

Figure 2 - Dashboard data flow

2. Contents

The contents of this deliverable are, aside from this descriptive document, the first version of the COMPACT Dashboard web application (COMPACT_Dashboard-1.0.0.war) and the COMPACT Dashboard data model that allows creation of the required database structure (COMPACT_Dashboard-1.0.0.sql).

Usage of these files is described more in depth in the installation procedure below.

3. Installation

3.1. Database creation

As a first step, a new database must be created. To this end, the file "COMPACT_Dashboard-1.0.0.sql" contains the necessary SQL sentences to create the database.

The procedure is as follows:

- Connect to database as user "root":
- Create a new user "compact"
- Create a new database or schema named "compact"
- Execute "COMPACT_Dashboard-1.0.0.sql" within the schema "compact"
- Grant select/insert/delete/update on all tables of schema "compact" to user "compact"

After executing these steps, the database should be ready to start working. Note that the script creates a default user "compact" with default password "*mypassword*". It is mandatory to change the password after the first login.

3.2. Web application deployment

The Dashboard application is distributed as a WAR file to be deployed in a servlet engine. Although any servlet engine should be enough, comprehensive tests have been taken under Apache Tomcat/8.5.14 with Java 1.8.0_181, on a Debian 9.5.

It is recommended that the application be deployed in the root ("/") of the web server. Still, as all paths within the web application are relative, the application should work correctly under any path.

The application makes use of HTML forms for authentication, which means that passwords are sent as part of the requests, so it is strongly advised that the application is set to use HTTPS (on default port 443), instead of HTTP (default port 80), as the latter would result in clear text passwords being sent through the network.

Finally, it is also recommended that a web server is installed in front of the servlet engine, to manage client connections more cleanly. Although any web server will work, tests have been performed using a local installation of Apache 2.4.25, communicating with Tomcat via AJP.

3.3. Configuration

3.3.1. Database

The COMPACT Dashboard requires a valid database containing application related information, such as users, categories and graph data. Although the use of JDBC throughout the application allows (in theory) any database engine to be used, tests have been performed

with both MySQL 8.0.12 and MariaDB 10.1.26. An SQL file (COMPACT_Dashboard-1.0.0.sql) is also provided to initialize the database.

For the web application to access the database, a JNDI resource with name "jdbc/COMPACT_DB" must be associated to the servlet context. For example, the following resource can be defined:

```
<Resource name="jdbc/COMPACT_DB"
    auth="Container"
    type="javax.sql.DataSource"
    username="compact"
    password="mypassword.changed!"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/compact"
    testWhileIdle="true"
    testOnBorrow="true"
    testOnReturn="false"
    validationQuery="SELECT 1"
    maxTotal="15"
    maxIdle="3"/>
```

Code 1 - JNDI datasource resource example

This configuration can be modified by decompressing the WAR file and adapting the file META-INF/context.xml, or by creating an external context definition XML file (with Tomcat, a file named $CATALINA_BASE/conf/[enginename]/[hostname]/[context_name].xml, e.g. /etc/tomcat8/Catalina/localhost/ROOT.xml).

Note that the use of the default database password is strongly discouraged. A different password should be created for the database user and the resource definition adapted accordingly.

3.3.2. Communication bus

The COMPACT Dashboard receives information from all other components via the Kafka infrastructure acting as communication bus. For the Dashboard to correctly display the graphs and messages generated by other components, it has to be able to connect to the communication bus.

To do so, a context parameter named "communication-bus" has to be defined. This parameter must contain a comma separated list of hostname:port pairs indicating the different bootstrap servers. The Dashboard will connect to these servers and start consuming messages from the "Dashboard" Kafka topic. The configuration of this parameter is done, as with the database parameter, in META-INF/context.xml or in the corresponding external context definition ($CATALINA_BASE/conf/[enginename]/[hostname]/[context_name].xml). The default is trying to connect to a Kafka server located in the same machine, and in the Kafka default port (i.e. 127.0.0.1:9092). An example of this configuration is:

```
<Parameter name="communication-bus" value="127.0.0.1:9092"
         override="false"/>
```

Code 2 - Communication Bus parameter definition

3.3.3. Session management

Due to the way that the COMPACT Dashboard has been designed, HTTP sessions would never time out, as the Dashboard is constantly being refreshed. This is not a secure approach, and so the Dashboard performs its own session timeout management. To this end, a session timeout can be defined using a context parameter named “compact-session-timeout”, which contains the timeout in seconds. By default, its value is 900 (15 minutes).

An additional parameter named “compact-session-timeout-warning” has also to be defined. This parameter indicates, in seconds, how long before session timeout the user receives a visual warning indicating that the session is soon to expire. By default, it has a value of 60 indicating that the user receives a visual warning 1 minute before session expiry.

As they are context parameters, they can be set either in META-INF/context.xml or in the corresponding external XML context definition file. Examples of these parameters are:

```
<Parameter name="compact-session-timeout" value="900"
         override="false"/>

<Parameter name="compact-session-timeout-warning" value="60"
         override="false"/>
```

Code 3 - COMPACT Dashboard session parameters

3.3.4. Configuration example

Summarizing all the above, this is an example of a context definition file containing the default values:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml>
<Context>

<Resource name="jdbc/COMPACT_DB"
   auth="Container"
   type="javax.sql.DataSource"
   username="compact"
   password="mypassword.changed!"
   driverClassName="com.mysql.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/compact"
   testWhileIdle="true"
    testOnBorrow="true"
    testOnReturn="false"
    validationQuery="SELECT 1"
    maxTotal="15"
   maxIdle="3"/>
```

```
<Parameter name="communication-bus" value="127.0.0.1:9092"
          override="false"/>

<Parameter name="compact-session-timeout" value="900"
          override="false"/>

<Parameter name="compact-session-timeout-warning" value="60"
          override="false"/>

</Context>
```

Code 4 - COMPACT Dashboard configuration example

As stated above, the default database password should never be used in a production environment.

4. Content update

4.1. Communication bus and basic JSON format

All communication with other COMPACT components to update the information displayed in the Dashboard is performed by using the communication bus, with the topic "Dashboard". The Dashboard subscribes to this topic and any module can send messages to this topic in order to generate additional information or update existing one. The message sent to Kafka is a JSON object, containing two attributes:

- **messages**: An array of JSON objects describing the different COMPACT messages that the application wants the Dashboard to display. All received messages will be displayed regardless if the message was received previously. The format of these JSON objects is described in depth later.

- **graphs**: An array of JSON objects describing the graphics that the application wants the Dashboard to display. In this case, if a new object defines a category and name that already exist, the existing graph is updated. If the category/name combination does not yet exist, a new graphic is created. The format of these JSON objects is described in depth later.

An example of a valid (but useless) empty communication bus message is as follows:

```
{
  "messages": [],
  "graphs": []
}
```

Code 5 - Example of a valid message for COMPACT Dashboard

4.2. COMPACT messages

COMPACT modules can send alerts to be displayed in the Dashboard, by sending JSON objects in the “messages” attribute of a communication bus message. These JSON objects contain the following attributes:

- **level**: An integer between 0 and 100 indicating the severity of the message (0-25 means low severity, 26-75 medium and 76-100 high).
- **date**: A string containing the date when the alert was generated (not sent), in “YYYY-MM-DD hh:mm:ss.ccc” format. For instance, "2018-09-23 13:45:17.232"
- **category**: A string containing the category this message belongs to (“Risk management”, “Educational services”, “Monitoring” and so on). This value is not enforced, so an invalid value still results in the message being displayed.
- **tool**: A string containing the tool that generated this message. As with “category”, this value is not enforced so an invalid tool name still results in the message being displayed.
- **content**: A string containing the text to be displayed

An example of a communication bus message containing an alert is as follows:

```
{
   "messages": [
                    {
                       "level": 90,
                       "date": "2018-09-23 13:45:17.232",
                       "category": "Risk management",
                       "tool": "TO4SEE",
                       "content": "Risk level progressed to Very High"
                     }
                   ],
   "graphs": []
}
```

Code 6 - Example of a Communication Bus message containing a TO4SEE alert

4.3. Graph data

4.3.1. Basic structure

Graphics can be created and updated by sending a special JSON object. The attributes of this object are:

- **category**: A string describing the category this graphic belongs to. It must be one of the categories already defined in the database (corresponding to the menu options, except “Messages” and “Users”).

- **name**: A string containing the name of this graph. If a graph with this name already exists in the specified category, this communication bus message is interpreted as an update. Otherwise, a new graph is created.
- **chartspan**: An integer indicating the width of this graph. By default, a row can contain up to 3 graphs. A value of 3 indicates that this graph fills the entire row, while a value of 2 indicates that this graph is double the normal width. The default value is 1.
- **location**: An integer between 0 and 999 indicating the location of this graph. Graphs are shown in ascending order of this field. The exact value of the field is not relevant, apart from being relative to other graphs (i.e. three graphs with locations 1, 2, 3 are shown in the same positions as graphs with locations 7, 59, 153). The default value is 500.
- **content**: A JSON object with the content to be displayed within the graph. This object must contain the following attributes:
    - type: A string indicating the type of the attribute. Valid values are indicated later.
    - data: An attribute that defines the content of the graph. Its format depends on the value of the "type" attribute.
    - options: An attribute that defines additional options for the graph. The available options depend on the value of the "type" attribute.

4.3.2. Table graphs

Table graphs are those that contain "table" as the value of the "type" attribute. In this case, the content has to be shown in a tabular way. The JSON object contains the following attributes:

- **type**: A string indicating the graph type. In this case it must be "table"
- **data**: A JSON object containing the table data. It must contain the following attributes:
    - labels: An array of strings containing the values to be shown in the table header. It must have the same number of elements as the table has columns.
    - datasets: An array of arrays of strings containing the different values to be shown. Each element of the outer array represents a row, whereas elements of the inner array represent cell values.
    - backgroundColor: An array of strings containing the background colour for each table row. It must have the same number of elements as the table has rows.

An example of a message defining a table of 4 rows (plus header) and 2 columns is as follows:

```
{  "messages": [],
   "graphs": [
        { "category": "Risk assessment",
          "name": "Assets",
          "content": {
                "type": "table",
                "data": {
                      "labels":["Asset", "Threat level"],
```

```
                    "datasets":[["Data on personnel", "VERY HIGH"],
                                ["On-going R&D innovation projects", "MEDIUM"],
                                ["On-going R&D innovation projects", "MEDIUM"],
                                ["Copyright", "LOW"]
                               ],
                    "backgroundColor": [ "#FF6384", "#FFD056", "#FFD056", "#4BC0C0" ]
                   }
                }
            }
        ]
}
```

Code 7 - Example of a message defining 4 rows and 2 columns.

4.3.3. HTML graphs

HTML graphs provide greater freedom when defining the graphic. They are identified because the value of the type attribute is “html”. They contain the following attributes:

- **type**: A string indicating the graph type. In this case it must be “html”
- **data**: A string containing the HTML code to be shown within the graphic. The text is passed “as is”.

An example of an HTML graph is as follows:

```
{
 "messages": [],
 "graphs":
  [
  {
   "category": "Communication",
   "name": "Last publications",
   "content":
    {
     "type": "html",
     "data": "<table class='table table-bordered'>
          <tr><th>Title</th><th>Category</th><th>Date</th></tr>
          <tr><td><a href='https://www.cisco.com/c/en/us/about/security-
       center/firewall-best-practices.html'>Cisco Firewall Best Practices
       Guide</a></td><td>Best practices</td><td>2018-09-14</td></tr></table>
          <img src='./img/Compact_logo_public_670.jpg' style='width: 100%;'></img>"
    }
   }
  ]
 }
```

Code 8 - Example of an HTML graph

4.3.4. Chart.js graphs

Any other type not mentioned yet (i.e. anything but “table” and “html”) is considered to be a Chart.js [2] graph. In this case, the attributes are:

- **type**: A string containing the specific *chart.js* type (e.g. “bar”, “pie”…)
- **data**: A JSON object containing the data to be shown. It is completely type-dependant.
- **options**: A JSON object containing options on how the data is to be visualized (e.g. axis values, labels, legend…)

A simple example of a bar graph is as follows:

```
{
 "messages": [],
 "graphs": [
    { "category": "Monitoring",
      "name": "Non-closed issues by priority",
      "content": {
          "type": "bar",
          "data": {
              "labels":  ["Critical","High","Medium","Low","Very low"],
              "datasets":[{
                 "label":"Number of issues",
                 "data":[3, 5, 7, 4, 2],
                "backgroundColor":
["#FF6384","#36A2EB","#FFD056","#4BC0C0","#9964FF"]
                }
              ]
            },
            "options": {
                "responsive": "true",
                "scales": {"yAxes": [{"ticks": {"beginAtZero": "true"}}
                    ]
              }
            }
       }
      }
   ]
}
```

Code 9 - Example of a bar graph

5. References

[1] L. Eccher *et al.*, “D3.3. Components Evolution Plan (EIB),” 2018.

[2] Chart.js, “Chart.js project,” 2018. [Online]. Available: https://www.chartjs.org/. [Accessed: 07-Dec-2018].

6. Checklist for S.E.L.P. by design

Risk	Requirement
Potentially severe impact of research results on human rights of individuals or groups (e.g. privacy issues, discrimination, stigmatisation)	• Risk assessment (fill in the DPIA) • Indicate the methods used for correct interpretation of research results, to avoid/reduce the negative impact on human rights • Indicate the methods used regarding the dissemination and publication of results, to reduce the negative impact on human rights • State that no data other than the results of the project (software and documentation) will be exported to non-EU Member States
Please justify your measure(s): COMPACT is a research project and could potentially have severe impact on human rights of individuals due to the collection and elaboration of their data, it could be a danger for the privacy of the participants. Only project results will be disseminated in an aggregated way to ensure anonymity and the lowest possible negative impact on human rights of participants.	

Risk	Requirement
Potential misuse or abuse of research	• Indicate the measures used to reduce/avoid the potential misuse or abuse of the research • Indicate details on the storage and destination of research data • If applicable, store copies of personnel security clearances
Please justify your measure(s): Research data will be stored on S21sec servers. The server is protected by S21sec SOC-CERT service (H24 operation), data is encrypted and only available to a limited number of people to prevent misuse and potential abuse.	

Risk	Requirement
Non-compliance with data protection by design and by default principles	• Risk assessment (fill in the DPIA)
Please justify your measure(s): Refer to section 7.	

Risk	Requirement
Disclosure of confidential information	• Indicate the methods used regarding the dissemination and publication of results, to avoid the disclosure of confidential information of partners • State that partners complied with non-disclosure agreements and internal contracts in relation to research data
Please justify your measure(s): The project results will be disseminated using the project's dissemination channels but only aggregated data will be shared. The project will comply with non-disclosure agreements with third parties and all study and trial deliverables will be internally checked.	

7. Data Protection Impact Assessment (DPIA)

This deliverable does not provide a filled up DPIA, the reason is that the COMPACT's consortium as agreed to create a separated document, the Common DPIA, which will include the DPIA for each component/partner and will be updated whenever it will be necessary.