Horizon 2020 Programme
Instrument: Innovation Action

Proactive Risk Management through Improved Cyber Situational Awareness

Start Date of Project: 2016-09-01

Duration: 36 months

D4.6 PROTECTIVE UI Framework - CSA Visualisation v2

Deliverable Details	
Deliverable Number	**D4.6**
Revision Number	**E**
Author(s)	**AIT**
Due Date	**31/10/2018**
Delivered Date	**24/10/2018**
Reviewed by	**AIT**
Dissemination Level	**CO**
Contact Person EC	**Alina-Maria Bercea**

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement no 700071.

Contributing Partners

Contributing Partners	
1.	AIT (contributor)
2.	SYNYO, PSNC (contributor, reviewer)
3.	PSNC (contributor, reviewer)
4.	GMV (reviewer)

Revision History

Revision	By	Date	Changes
E	AIT	24/10/2018	Version Submitted to Agency
B1	AIT	22/10/2018	Reviewed by SYNYO, TIDA and GMV
A4	SYNYO	22/10/2018	Added additional valuations
A3	AIT	21/10/2018	SC4 and SC5 input
A2	SYNYO	19/10/2018	Ready for input from consortium
A1	AIT	17/10/2018	First outline and table of content

Abbreviations

AS	Autonomous System
ASM	Asset State Management
CA	Context Awareness
CSIRT	Computer Security Incident Response Team
CSV	Comma Separated Variable
DDoS	Distributed Denial of Service
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
IP	Internet Protocol
JSON	JavaScript Object Notation
MSP	Managed Service Provider
NREN	National Research and Education Network
SC	Scenario
SME	Small Medium Enterprise
TI	Threat Intelligence

Executive Summary

The software in this delivery implements the PROTECTIVE UI. This report describes the implementation and its main features. It develops visualisation support to give the analyst an overall view of their situational awareness. It provides views on constituency, mission, threat and risk awareness -- both individually and showing the relationship between different perspectives. It allows the analyst both to drill down to identify particular issues and to have a high-level overview at the organisation or asset class level. It provides support to view trend information in order to discover threat and alert patterns over time. It combines inputs from both threat and risk awareness functions as well as context awareness.

The document is outlined as follows: Section 1 introduces the document and sets out the purpose. Section 2 describes the requirements and scenarios that the document implements while section 3 describes the architecture and design decisions that drove the development of the UI. Section 4 describes the implementation including the principal Scenarios UI components. Chapter 5 outlines the Pre-requisites for deployment of the software. Finally, Chapter 6 and 7 address IPR and ethical issues relevant to this deliverable (there are none).

Horizon 2020 Programme
Instrument: Innovation Action

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement no 700071.

List of Figures

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement no 700071.

List of Tables

1 Introduction

The software in this delivery implements the PROTECTIVE UI. This report describes the implementation and its main features.

It develops visualisation support to give the analyst an overall view of their situational awareness. It provides views on constituency, mission, threat and risk awareness -- both individually and showing the relationship between different perspectives. It allows the analyst both to drill down to identify particular issues and to have a high-level overview at the organisation or asset class level. It provides support to view trend information in order to discover threat and alert patterns over time. It combines inputs from both threat and risk awareness functions as well as context awareness..

2 Scenarios and Requirements

Scenarios, requirements and architecture are documented in chapter three of the deliverable "D2.1 Requirements Capture, Specification, Architectural Design and Model". In this chapter, those scenarios and requirements which have been fulfilled or partially fulfilled by this delivery are documented. It also includes fulfilment from earlier deliveries.

2.1 Scenarios

The scenarios for PROTECTIVE are shown in Figure 1. This deliverable D4.6 contributes to each of the scenarios shaded in yellow in Figure 1. For scenario 1 a dashboard is provided for the visualisation of recent relevant activities (4.3). For scenario 2, an alert search form is provided in Prot-Dash for searching of malicious activities (4.4). Time series plots are provided for SC3's monitoring of data trends over time (4.4). For SC4, query forms for meta-alert prioritization and a sortable meta-alert table has been provided (4.6: note this is not yet integrated with Prot-Dash). Force directed graph visualisations are provided for SC7, together with an intuitive mechanism for interactive with CA_MAIR, involving uploading and downloading data and interacting with the directed graph (4.9).

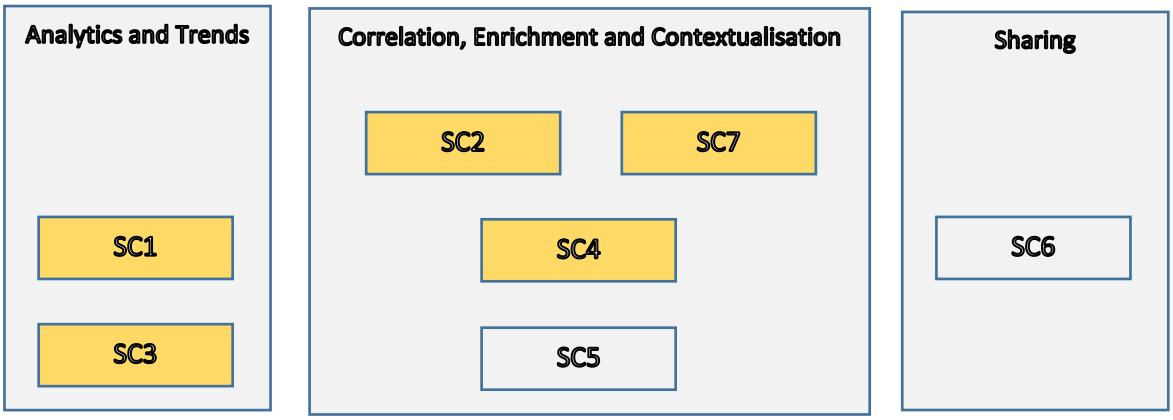


Figure 1: Scenario Contribution of Deliverable

2.2 Requirements

The following requirements have been fulfilled ("Done") or partially fulfilled ("Partially" by this delivery.

2.2.1 Analytics

Table 1: Analytics Requirements

ID	AN-04	Done

Type	FR	
Slogan	PROTECTIVE MUST provide statistics on all ingested information	
Rationale	To improve the awareness of the operator, an accurate overview of the state of the current system must be provided, containing an overviews of asserts, number of alerts/meta-alert etc.	
Related Scenarios	SC-1	
Related Requirements	N/A	
ID	AN-05	Done
Type	FR	
Slogan	PROTECTIVE MUST monitor trends in the ingested information	
Rationale	Trend monitoring of e.g., the number of alerts ingested (per network, sensor, data source etc.) enables the operator with an overview of past events	
Related Scenarios	SC-3	
Related Requirements	AN-04	

2.2.2 User Interfaces

Table 2: UI Requirements

ID	UI-01	Partially
Type	FR	
Slogan	PROTECTIVE MUST allow administrators and operators to manage and interact with the systems	
Rationale	A key part of improving the workflow of the operator is to provide an intuitive and easy to use interface to manage the PROTECTIVE system and to be informed	
Related Scenarios	SC-1, SC-2, SC-3, SC-4, SC-5, SC-6, SC-7	
Related Requirements	N/A	
ID	UI-02	Done
Type	FR	
Slogan	PROTECTIVE MUST allow the operator to view key statistics about the information within the system	
Rationale	To increase situational awareness, the operator must be able to view various statistics about the system; number of sensor sources, alerts in total/per sensor etc.	
Related Scenarios	SC-1	
Related Requirements	N/A	
ID	UI-03	Done
Type	FR	
Slogan	PROTECTIVE MUST be able to notify the operator about important events	
Rationale	Certain events and threats require an immediate response, for this reason the information needs to be pushed to the operator as soon as the event is identified	
Related Scenarios	SC-3	

Related Requirements	N/A	
ID	UI-04	Partially
Type	FR	
Slogan	PROTECTIVE MUST support user (operator) preference persistence	
Rationale	Operators may have different preferences and areas of responsibility, why it's necessary that their system view can be configured to fit their needs. This includes e.g., management of preferences including the ones required for prioritisation by means of MCDA techniques, storing of queries etc.	
Related Scenarios	SC-1, SC-4	
Related Requirements	N/A	
ID	UI-05	Partially
Type	FR	
Slogan	PROTECTIVE MUST support referencing between stored objects	
Rationale	To allow the operator to navigate through the vast amounts of stored information, the relevant objects need to be linked together; e.g., while investigating a meta alert, the operator should be able to get additional information about the system that may be impacted, the attackers IP address, etc.	
Related Scenarios	SC-1, SC-2, SC-3, SC-4, SC-5, SC-6, SC-7	
Related Requirements	N/A	

3 Architecture and Design

3.1 Introduction

In an attempt to give a somewhat detailed overview of the Prot-Dash visualisation web user interface for the Protective project, this chapter will detail the design and implementation choices that were made as well as some of the details of the current implementation.

3.2 Design Choices

3.2.1 Single Page Application (SPA)

During the design stage, it was decided that the responsiveness of the web app was extremely important to the user experience. A SPA is a web application that fits into a single page. Dynamic actions can be carried out on the page without adding long loading times by having to refresh the entire page. For example, if a table on the page needs to get fresh results from the database, it queries the database in the background and triggers the reload of the table content but not the entire page. This makes for a smoother experience for the user when clicking around different parts of the web application.

SPAs are common and used by Gmail (Google, 2018a), Google Maps (Google, 2018b), Facebook (Facebook, 2012) and GitHub (GitHub, 2018) to name a few. Modern Javascript frameworks that harness the power of AJAX make this possible. Although there are several SPA frameworks to choose from, ultimately, we decided on the use of Angular (Angular, 2018) for several reasons. Angular uses change detection, which monitors changes in the source code and can render the changes immediately in the web app on the browser. This greatly speeds up the development process, allowing for the addition of more features quickly. Our existing developers had a good knowledge of Angular.

Angular allows for the simple integration of any JavaScript library that we wish to use. There is easy control of in application routing and module loading.

3.2.2 Look and Feel

We recognise the importance of the look and feel of a web application from both a user and a development point of view. From a user perspective, the web app must be easy to navigate. Names and links of different sections of the page should be clear. As much as possible, it should be obvious what the functionality of the different links and buttons are. For developers, a clear user interface is also important. When new functionality is developed, it makes the decision on where exactly to deploy the functionality easier. For example, any notification functionality should be linked to the notification dropdown menu.

While this functionality and clarity is important, it is also important to make the user interface elegant and modern. Much thought and time was spent on the choice of an appropriate colour scheme and layout. Ultimately, it was decided that the Bootstrap-based theme “SB Admin” should be used. SB Admin uses the default Bootstrap 4 (Bootstrap, 2018) styles along with a variety of powerful plugins to create a convenient framework for creating admin panels, web apps and back-end dashboards. In addition, SB Admin provides a generic web template as well as an Angular-based template. This made integration with our application much easier. This is illustrated in Figure 2.

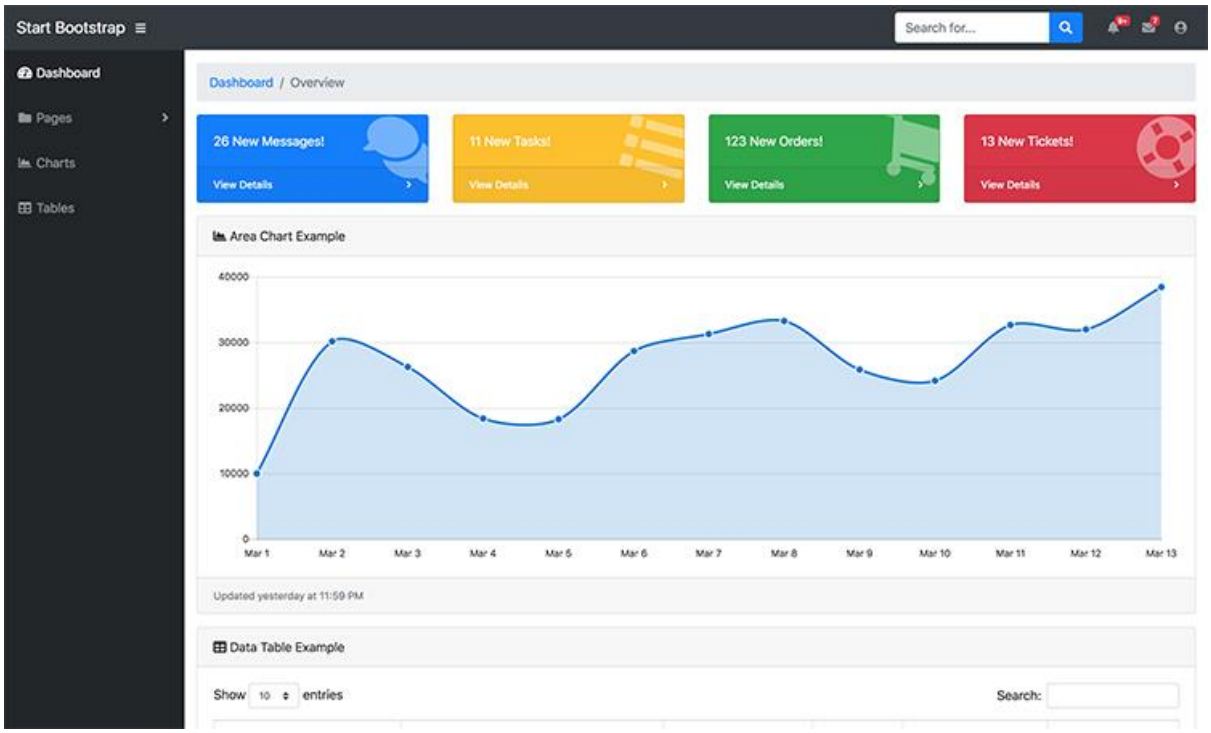


Figure 2: SB Admin Template

3.2.3 Lazy Loading

One of the features that made Angular an attractive choice for the web app was lazy loading. In Angular, using components and modules, we can group related pieces of functionality together. Some pieces of functionality such as access to the user database or the initial dashboard are required at application start-up. However, some modules and components are not strictly required to load at start-up, for example, a very specific graphing module. In these cases, we can use lazy loading of modules. This means that the modules are only loaded when required on-demand when the user

accesses certain parts of the web application. This is advantageous because it can drastically decrease the start-up time of an application, which can improve the user experience.

3.2.4 Routing in Angular

We are all familiar with how to navigate through a web page or the Internet in general, click a link and we are redirected to a different page. In Angular, the "router" imitates this behaviour. It interprets a browser URL as an instruction to navigate to a client-generated view. It supports the passing of parameters along to the specified view component in order for it to decide what content to present. Certain routes in a web site or web application can be "guarded" with programmatically defined criteria. For example, a user cannot access a certain page if they are not logged in. Angular guards will be discussed more in 4.2. In addition, we can decide whether to handle the routing mechanism in angular centrally in one routing file or to distribute the responsibility for routing over several files. In this project, we have opted for the latter. The project consists of so many modules that it made sense to manage routing on a module-by-module basis.

3.3 Implementation Choices

This section details the decisions that were made in relation to the implementation of the Prot-Dash web application. Several concepts will be explained and it will be made clear why we needed to conceive these concepts.

3.3.1 User Interface

To allow the user to view many different visualisations at the one time, widgets are used. The user can define as many widgets as they wish. All of them will be displayed on the specific dashboard where the user defined them. It should be noted that the convenient user interface allows the selection of implemented Data Providers, Mapping Providers and View Providers (described below). As an additional feature, where possible, selection of non-compatible Providers is disabled to avoid user confusion. Widgets can quickly be moved, removed or resized. For convenience, it is possible to view a full screen version of the widget to allow more close examination of a visualisation. A widget can also be edited to alter the data being displayed or make some custom changes to the visualisation. A widget is illustrated in Figure 3.

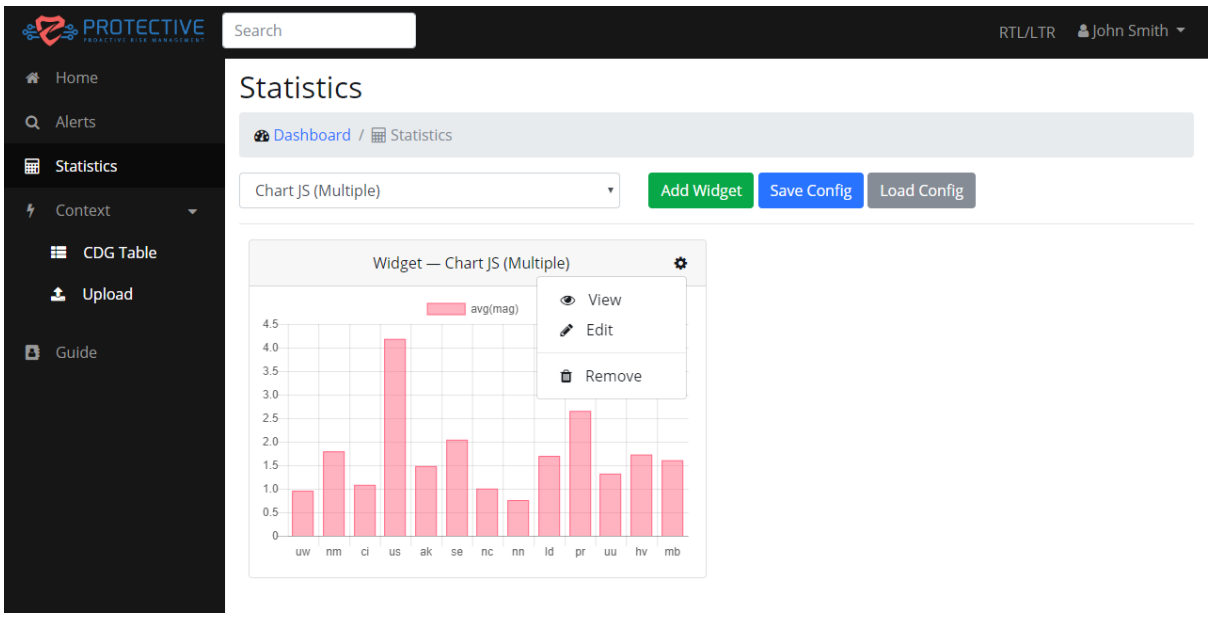


Figure 3: Widget

3.3.2 Prot-Dash Flexibility

Upon careful consideration, it was decided that one of the most important concerns when implementing Prot-Dash was that it was flexible. Flexible in terms of what functionality could be added by integrating already existing systems and libraries. If we implement the web app to only be able to get data from Rest APIs that return a specific format, then we limit the amount of data we have access to. Other data sources could be adapted by creating custom Rest APIs but, if we had to do this on a case-by-case basis, this would greatly increase development time.

Similarly, in terms of visualisation, if we limit the data to only one type of graph, then we are hiding insights from the user. Javascript has a great many visualisation libraries and we considered this in our implementation of Prot-Dash. We realised that not all of these libraries accept data in the same format and therefore had to take this into account in our implementation. The following sections 3.3.3, 3.3.4 and 3.3.5 explain the three core concepts that allow Prot-Dash this flexibility. How these concepts fit into the web application is illustrated in Figure 4.

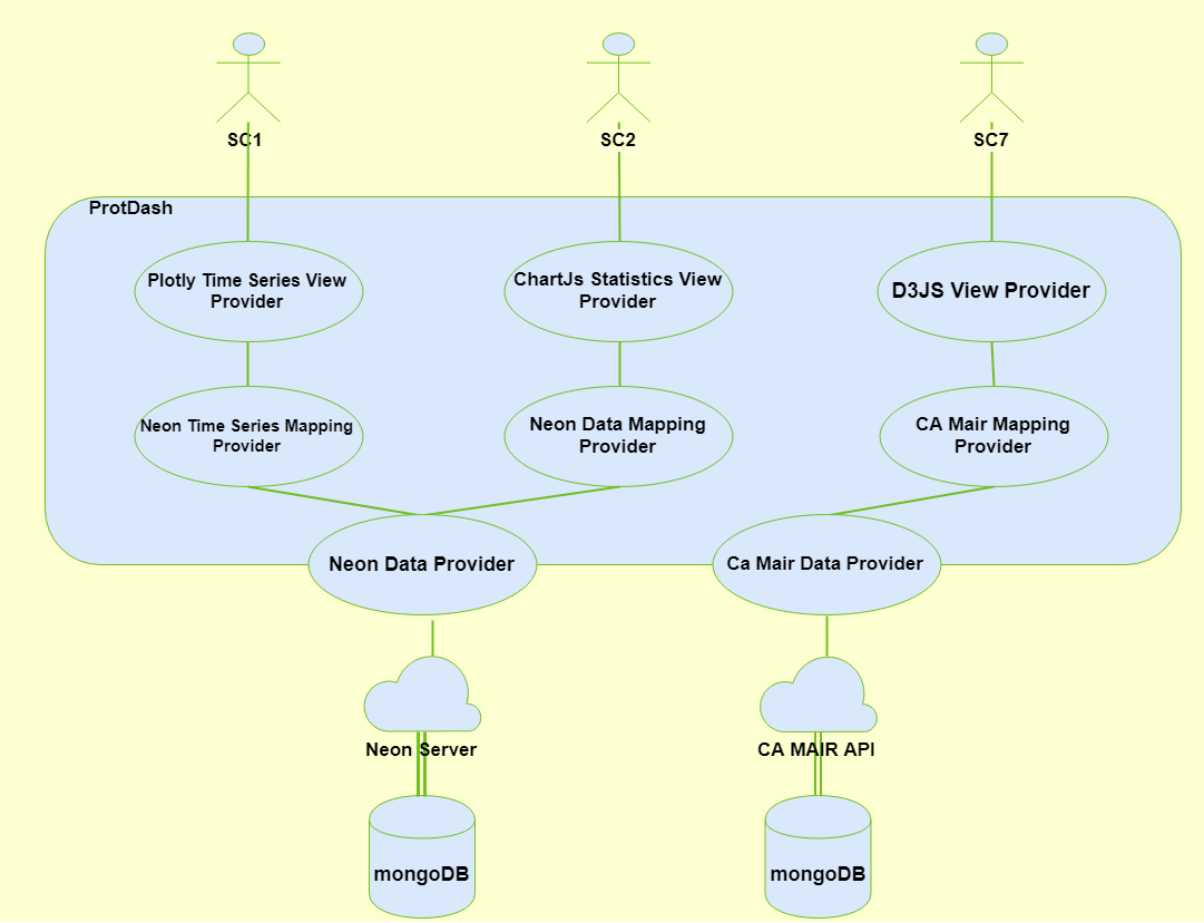


Figure 4: Prot-Dash Concept Mapping

3.3.3 Data Provider

The concept of the Data Provider is a generic object that can be adapted to provide data to the application from any data source, such as any kind of database, using Rest API calls, using specific query languages or other sources. The Data Provider in Prot-Dash is an interface that is used by any service or component in our Angular project that accesses some arbitrary data source.

All the interface requires is that any service or component that uses it has three different methods. The first method getDataType() retrieves the data type of the data that will be incoming from a

specified data source. The second method, testDataProvider() serves to test both the availability and the functionality of a data source. For example, to see if data is returned in the correct format. The final method is the query() method, which makes the call to the data source and returns the data that can later be used in visualisations.

Any service or component that uses the Data Provider interface can add whatever other methods it needs to query or handle the incoming data, as long as it implements the methods describe above. Any new Data Provider that is created will be added as an option for the user to select on the web user interface. So, the user can easily choose between different data sources.

3.3.4 Mapping Provider

The basic idea of a Mapping Provider is to convert any data that is entering Prot-Dash from the data source data format to a format that can be used by whatever visualisation library the user wants to display data with. For example, the data source may provide data that is in raw JSON format but the graphing library may require an object with; date, time, value and weight properties. Any manipulation of the data in this regard takes place in the mapping provider.

In terms of programmatic implementation, the Mapping Provider is supplied as an interface that can be implemented by a service or component that needs to do conversion of data format. An input() method is used to pass data into the Mapping Provider. The supportsDataType() method helps to determine whether the Mapping Provider being used actually supports the data type coming from the data source. If the data type is supported, data is loaded into the Mapping Provider using the input() method. The supportsViewType() method is generally called after the format conversion has taken place and is used to determine whether the “mapped” data is supported by the type of visualisation the user wants to use. If the data type is supported, the output() function is called to pass the data to the visualisation object or View Provider.

3.3.5 View Provider

The final concept that will be described is the View Provider, which uses graphing, charting and illustration Javascript libraries to provide visualisations for the data that has been formatted by the Mapping Provider. This illustrates the flexibility and power of the Prot-Dash concepts. Potentially any graphing library can be used and there are many. Some examples; Flot, chart.js, D3.js, NVD3, Sigma.js or Cytoscape.js to name a few. Although not all of these libraries have been put into View Provider implementations yet, it can be easily done in the future because of the simple flexibility of the web application.

The code design of the View Provider is slightly different than the other Providers in that it doesn’t just use a View Provider interface, it also inherits from a Base View Provider. There was no need to define a blanket interface that required the interface user to define a lot of methods required by the interface because the majority of methods are common across all View Providers. Inheriting from the Base View Provider component meant much less repetition of code and easier implementation overall. The inherited component provides many method implementations to do with discovering the correct Mapping and Data Providers, configuring the View Provider and displaying contextual error messages.

4 Implementation

In this chapter we describe the scope of the implementation and describe the user interfaces for the different scenarios.

4.1 Current scope of Application

In this section, we give detail of what is currently implemented in the application in terms of Data Providers and View Providers. We will also discuss briefly how authorization and access control have been enforced in Prot-Dash in a convenient and configurable way.

4.1.1 Configured Data Sources

Neon Server: Neon server is part of a larger project (Neon, 2018) that has the aim of providing a visualisation/data access framework for accessing and visualising many kinds of data easily. Neon server's Data Access API allows users to send a query to No-SQL databases using a SQL-like language. Neon does the "heavy lifting" by converting the query to a format that is understood by the target database. So, there is no need to create database-specific constructs. Neon allows this API to be accessed by a Javascript library or a restful endpoint. We use the former in our project. Neon server currently supports MongoDB, ElasticSearch and Spark SQL database back-ends.

Using the neon Data Access API Javascript library in our project, we created a Data Provider that accesses a MongoDB database. This database contains alert and security vulnerability information that is produced by other elements of the Protective project. The Neon Data Provider is configurable through the web application interface. Different collections and fields can be chosen within the UI, date ranges can be specified and data aggregations such as counts and averages can be chosen by the user.

CA MAIR API: The CA MAIR API is part of the Protective project that provides context-awareness data from a backend MongoDB database. The data specifically has to do with assets (hardware, software, network…_) that have been uploaded to the database through CA MAIR. Several restful endpoints have been exposed by CA MAIR that allow easy querying the database to acquire information on specified assets.

Specifically, we create a data provider for CA MAIR to retrieve data based on the names of specific assets within the database. The ultimate aim of this Data Provider is to construct a directed graph with granular information on every single asset present and the relationships between those assets.

4.1.2 Viewing Libraries Used

Chart.js: Simple, clean and engaging HTML5 based JavaScript charts (chart.js, 2018). Chart.js is an easy way to include animated, interactive graphs on any website or web app for free. A View Provider for Chart.js has been created that uses all 7 available charts; line, bar, radar, doughnut, pie, polar area, bubble and scatter. The bar chart is shown in Figure 3. Using the edit option on the widget dropdown menu allows the user to switch between the different types of charts.js charts to experiment with which view is the best.

D3.js: D3.js is a highly configurable low level library that can be used to create a multitude of different visualisations (D3JS, 2018). In particular, we use d3.js to create a force directed graph to visualize the data received from CA MAIR (0). The force directed graph shows the nodes of the directed graph in a colour coded way with a legend. The links or lines between the nodes represent some relationship between those nodes. The graph is interactive. The nodes can be moved around to make examination of the graph detail easier. As there may be hundreds of nodes to a graph, pan and zoom functionality has been added to make it more convenient for the user to visualise. Hovering over a node or a link will allow cause a label to appear, which gives some details about the data. Hovering over a node will also trigger highlighting of every link and node that is connected, making viewing relationships easier. It is also possible to filter out/in categories of nodes by clicking on the relevant label/type in the graph legend. More detailed node data can be viewed by clicking on the edit option in the upper right corner

of the widget and then clicking the View Provider tab. Detailed JSON data is available there on whatever node or link is clicked.

More sample D3JS visualisations have been provided here to show what D3JS is capable of but these implementations are not full.

Plotly: Built on top of d3.js and stack.gl, plotly.js is a high-level, declarative charting library. plotly.js ships with 20 chart types, including 3D charts, statistical graphs, and SVG maps (plotly, 2018). We use plotly for the visualisation of alert time series data from neon. A stacked bar chart is produced that can be used to show alert or vulnerability data grouped by type of attack or any other available statistic the user wants to group by. The time period, date and granularity of the data returned can be specified in the query. An interactive graph is produced, where a number of options are available. The time granularity can be changed, we can zoom or pan and hovering over the stacked bars details all the groups present in the bar.

4.2 Security Implementation

This section will give a brief introduction to the access control and authorisation methods we use for the web application. Further detail is provided in WP5 D5.4. To implement Single Sign-on authentication and role-based access control for the resources made available through Prot-Dash, we use Keycloak (Keycloak, 2018). Keycloak is meant to provide a “no fuss” solution for the securing of a wide range of applications. Users defined in Keycloak are given access to the application using Open ID Connect (Sakimura, Bradley, Jones, de Medeiros, & Mortimore, 2014)(OIDC). The location and configuration of Keycloak is supplied to the Angular application through the use of both a Javascript plugin and an Angular specific Keycloak library. We also configure our Prot-Dash application as a resource in the Keycloak server.

Any user trying to access the Prot-Dash UI without first providing their credentials is redirected to a Keycloak-specific login page where they must enter their username and password. Non-authenticated users are not allowed any access to the UI. This is enforced on the Angular side using a “guard”. All routes are guarded for authentication. Once the user is authenticated, Keycloak sends a JWT bearer token to the application. This gives details of the users’ roles that have been defined in Keycloak. For example, a user might have a Data Analyst role or a Dashboard role. Now, when the user tries to access a specific section of the application such as the “statistics” link, another “guard” kicks in. This “guard” tries to verify that the user in question has the permissions to access the specified resource. The bearer token is then used to access the Keycloak Entitlement API. A Requesting Party Token (RPT) with permissions detailing whether or not a user should be given access is returned from the entitlement API. A user may need several roles to access a specific resource or just one or perhaps none at all. It depends on how configuration is carried out. If a user is not granted access, they are re-directed to an access-denied link.

One of the key components used in the elaboration of the PROTECTIVE system are the seven scenarios defined during the requirements capturing process as part of the creation of PROTECTIVE D2.1 - Requirements Capture, Specification, Architectural Design and Model. In the context of visualisation, the following section discuss each of the seven scenarios, if applicable, from a visualisation point of view, focusing on what information shall be conveyed to the user, such that they can easily comprehend and consume the relevant information as well as getting an overview of the overall performance of the PROTECTIVE system in front of them.

Each of the following sections discusses a specific scenario, identifies the key visualisation necessary in order to convey the needed and relevant information as well as providing a mock-up of how these

visualisations may be implemented such that a further discussion and elaboration with the end-users, the operators, may be undertaken to further refine the functionality.

4.3 SC-1: System and Sensor Data Statistics UI

In order to aid the operator in their day-to-day tasks and activities, Scenario-1 (SC-1) assures the implementation of a general overview of the current status of the PROTECTIVE system instance they are operating as well as the most recent events. This view, shown in Figure 5, is based on a multi-layered approach. The first layer is represented by a high-level dashboard that includes the most important parameters, and that allows the operator to drill down into relevant activities. The second layer provides additional details about whatever component the operator chooses to investigate. The visualisations shown on the dashboard in Figure 5 are completely configurable but default to displaying the most recent alert information.

	Description
What	- **Overview of what is currently being ingested into the local PROTECTIVE instance.**
How	- **Statistics, time series, <see existing platform>**

- **Alerts per source:** This time series plot shows, using a plotly View Provider (4.1.2) stacked bar chart. The y-axis indicates the number of alerts that have been recorded. The x-axis shows the time at which the alerts were recorded. This time granularity is adjustable. Each bin represents a time period for which alert data was recorded. We can see that each bar is multi-coloured and each colour represents a source from which the alert is coming. The graph is interactive, it is possible to; pan, zoom and perform other functions by clicking the buttons on the graph.
- **Alerts per partner:** This is similar to the Alerts per source graph but in this case the different colours in the graph represent the partner from the which alert originated. A brief glance at the graph shows that most of the bars are blue and therefore most of the alerts are being generated by CESNET. Information like this would prove to be useful. This may prompt CESNET to have a closer look at why so many alerts are originating there and perhaps take some steps to counteract the attacks.
- **Alerts per category:** This graph, as the name suggests, is illustrating the category of attack using the colours in each of the bars. The categories include; Malware, Exploit Attempts, DDos alerts and others. This information is useful on its own but if we look at it in conjunction with the previous two illustrations, we gain more insights. The trend of the data in each of the graphs is the same. They are all recording from the same time period and therefore record the same number of alerts in each bar. If we look at the last bar, we see that the majority of alerts are from net.roedu and cz.cesnet, the most prevalent categories of alerts in this time period are severity.low and Attempt.exploit and it seems that it is mostly “cowrie” and “dionea” that are reporting these attacks. This is a good illustration of how these time series graphs give us insight.
- **Source status:** Finally, the table gives more specific information about the number of alerts being generated at the time recording. The dashboard overall gives a good overview of the threat status of the network and which partners are more at risk.

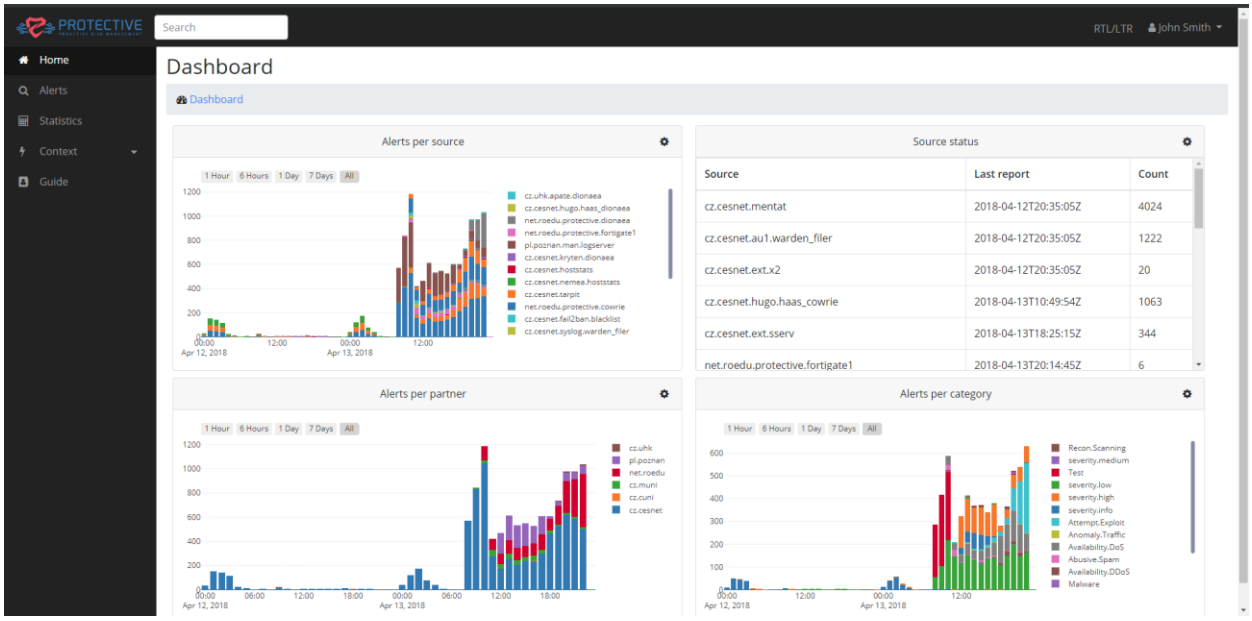


Figure 5: SC1 Dashboard Recent Events

Although it is strategically and analytically important to use visualisations to give a high level overview of the threats and vulnerabilities surrounding the Protective system, it is equally important to be able to drill down into the data to see the specifics of the alerts being generated by various attacks. A time series plot may give an indication of the category of threat or the source domain of the attack over a period of time but it does not tell you exactly where a specific threat came from or what tool reported it. For this reason, in SC1, a search form was implemented as part of Prot-Dash.

The search allows us to access all parts of the alerts database through Neon Server (4.1.1). This is illustrated in Figure 6. Through the form, we can choose any available database and any table within that database. However, the form comes pre-configured to access the “alerts” database so that we can immediately search through the alerts. We can query all alerts within a user-specified time period, we can filter data using fields. It is possible to specify a source IP address, target IP address or both. We can select which tool we want to see the alerts from and which category of alerts we wish to see

	Description
What	- **Configuration of prioritisation attributes** - **Most relevant events/incidents**
How	- **Database Search Form**

the results from. We also have control over the number of results that are returned. All of these filters are carried out on the server side to minimize any impact the search might have on the application speed.

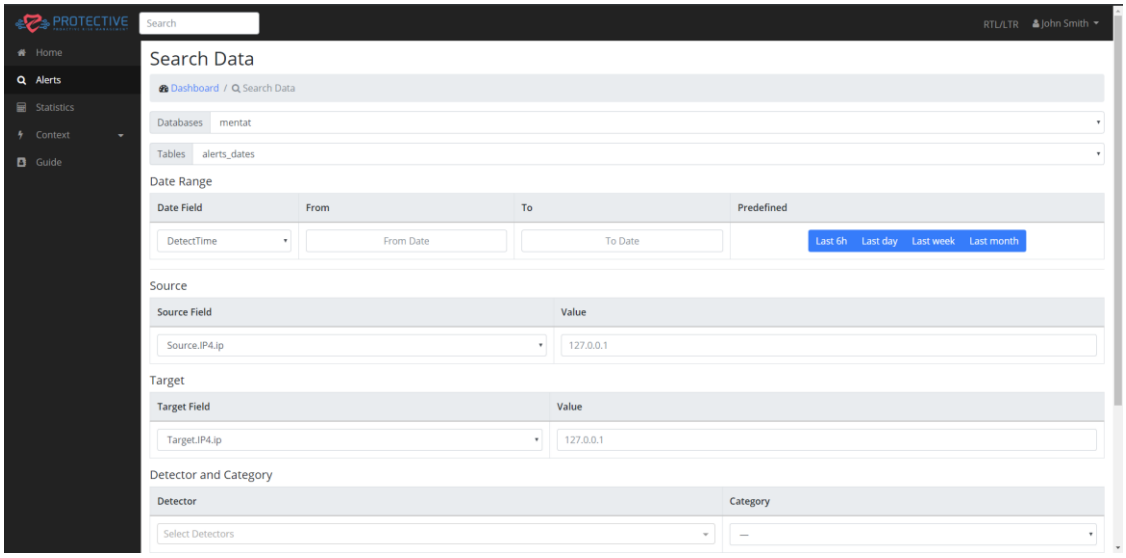


Figure 6: Alert Search Form

Once the query is sent to Neon Server with the relevant filters and time period specifications, a formatted table of results is returned. The results table is paginated in such a way as to minimize the performance impact on the page. In the design stage, we considered that there may be thousands of results returned and decided that there should be convenient ways of searching through the results from the initial search. This was achieved through the use of a client-side search. The user can easily search for ip addresses or country codes or categories or other things through the table search bar provided. Functionality is also provided to export the results to a CSV file. In this way, the user can take the returned data out of Prot-Dash and possibly use some third party software to analyse the data. Finally, a tab is provided that will show the original query to the database in JSON format. This may be useful if the user wishes to examine exactly what the filters were. A sample result table is illustrated in Figure 7.

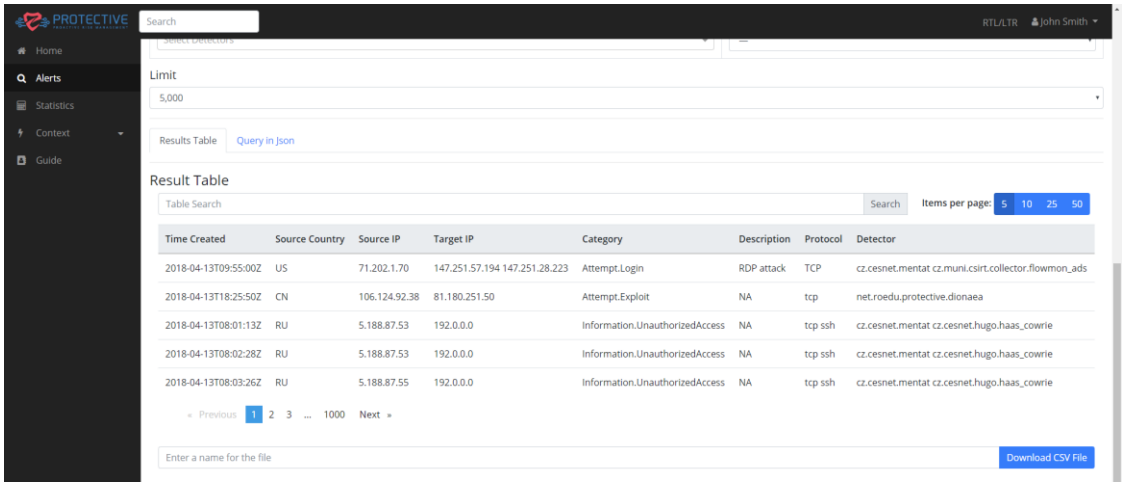


Figure 7: Database Search Results

4.4 SC-2: Reputation of Malicious Entities UI

SC-2 output is included in the Meta-alerts and so it does not have a dedicated Prot-Dash UI component.

4.5 SC-3: Time Series and Trend Monitoring UI

SC-3 is strongly interrelated with SC-1 in terms of what data is necessary to be collected, but pushes the requirements further as the scenario is mostly based on the presentation of information over time. Instead of the default dashboard time series plots that have a pre-configured time period and other pre-set query parameters, in SC3 we have full control over the parameters we include in our query.

	Description
What	- **Overview and trends of ongoing activities**
How	- **Time series with trend** - **Visual cues and alerts**

An illustration is provided in Figure 8. Notice that we have control several query parameters. We can specify which time field we use for the time period. We can specify the granularity of the data we wish to have returned. The user can group the data by a specified field. Filters can be added, as in 4.4, using any field available in the database.

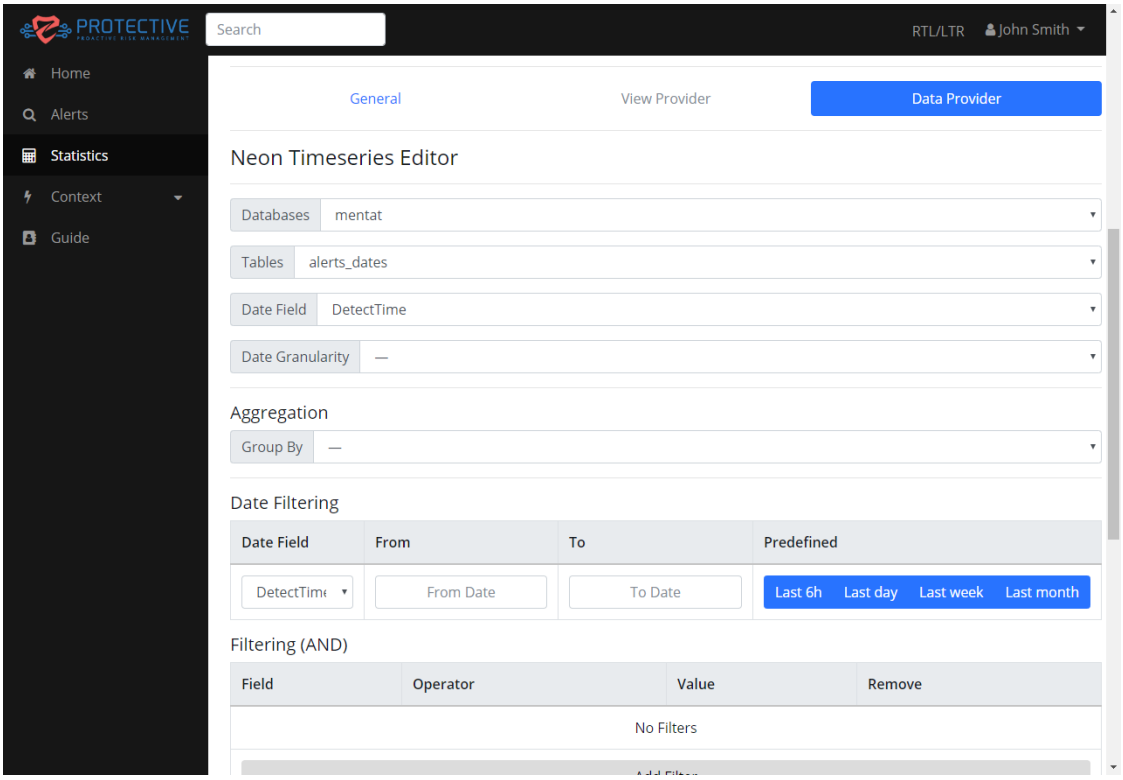


Figure 8: SC3 Time Series Query

The results of the query are then shown in Figure 9. In this example, we search alerts in a specified time period but group our query by IP address. We can see the IP addresses and colour codes along the right-hand-side.

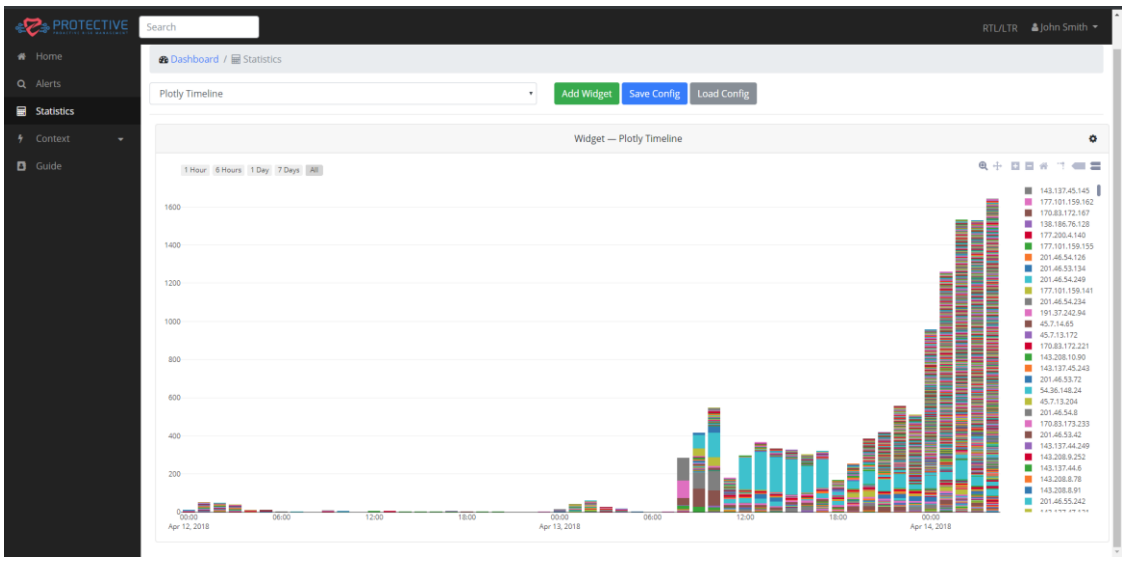


Figure 9: Results Time Series

4.6 SC-4: Alert Correlation and Prioritisation UI

Correlation and prioritisation consider two steps, namely the configuration of the correlation parameters, and the actual representation of the prioritised meta-alerts, tailored to the active operator, i.e., different operators may configure their priorities, which are in turn applied in the weighting of the priorities. Secondly, the prioritised meta-alerts will encapsulate relevant information that make it intuitive as to why they have been prioritised, severity of the alert, or other selected parameters that the operator has indicated are relevant for them.

	Description
What	- **The most relevant and critical incidents currently ongoing**
How	- **Prioritised list of relevant incidents (meta-alerts) and scores (for why they are important)**

A web query form has been provided for sending request to MongoDB with meta-alerts. This query form is shown in Figure 10. The user can enter the MongoDB filtering query, in this case a date/time-based query, in the “Query for ranking meta-alert” field. The example provided in Figure 10 is querying the database for all alerts detected on the 15th of June 2018. The “minIndex” and “maxIndex” fields allow the user to specify the slice of the data being returned and therefore the number of results in the returned table. Ticking the “Return as HMTL” box returns the results of the query as an HTML document that can be freely viewed on any browser. The default returned data type is JSON.

The result of the query is then shown in Figure 11. Note that only part of the returned table is shown because of space and visibility constraints. However, we can see that the meta-alerts have been ranked by priority, as provided by an internal ranking generator (based on MCDA model). Several pieces of information are returned including; the severity of the attack, the source and target

criticalities, the source and target ip addresses and the category of attack e.g. Abusive.Spam. Further details on the internals of Alert Correlation and Prioritization are detailed in D3.5.

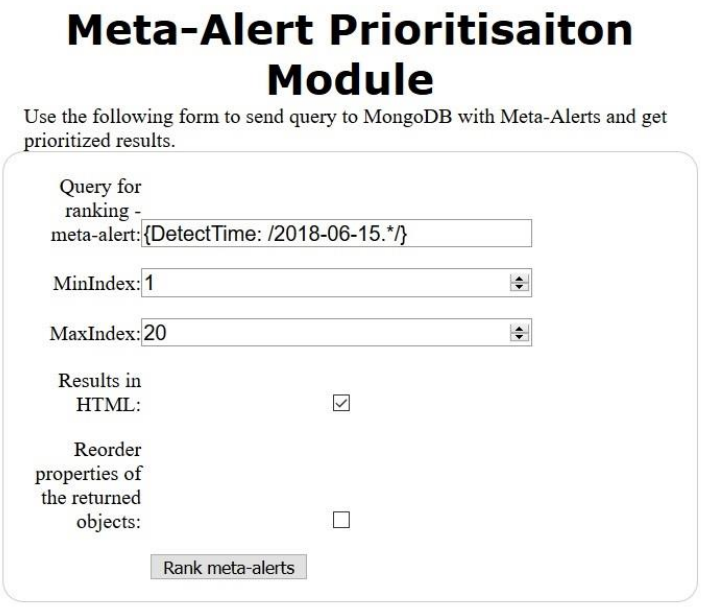
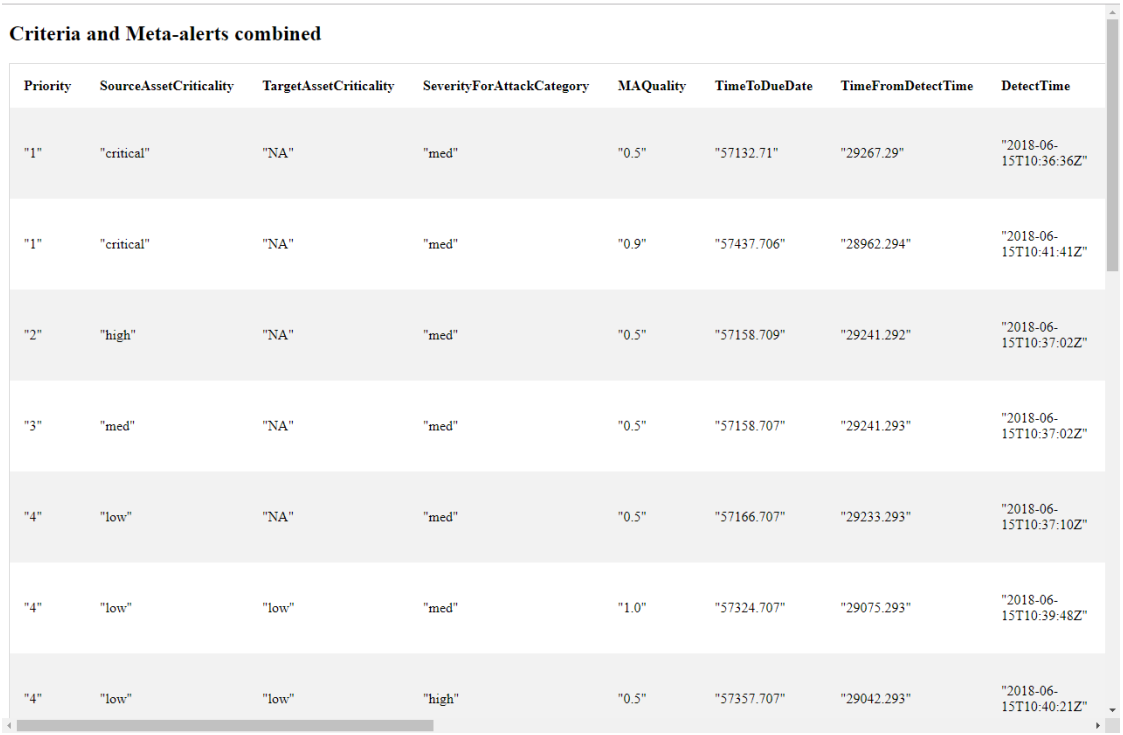


Figure 10: Meta-Alert Prioritization Query



Criteria and Meta-alerts combined

Priority	SourceAssetCriticality	TargetAssetCriticality	SeverityForAttackCategory	MAQuality	TimeToDueDate	TimeFromDetectTime	DetectTime
"1"	"critical"	"NA"	"med"	"0.5"	"57132.71"	"29267.29"	"2018-06-15T10:36:36Z"
"1"	"critical"	"NA"	"med"	"0.9"	"57437.706"	"28962.294"	"2018-06-15T10:41:41Z"
"2"	"high"	"NA"	"med"	"0.5"	"57158.709"	"29241.292"	"2018-06-15T10:37:02Z"
"3"	"med"	"NA"	"med"	"0.5"	"57158.707"	"29241.293"	"2018-06-15T10:37:02Z"
"4"	"low"	"NA"	"med"	"0.5"	"57166.707"	"29233.293"	"2018-06-15T10:37:10Z"
"4"	"low"	"low"	"med"	"1.0"	"57324.707"	"29075.293"	"2018-06-15T10:39:48Z"
"4"	"low"	"low"	"high"	"0.5"	"57357.707"	"29042.293"	"2018-06-15T10:40:21Z"

Figure 11: Meta-Alerts Query Table

4.7 SC-5: Prediction of Future Events

The alert prediction component represents an implementation of SC5 – prediction of future events. It predicts the probability of future alerts for known malicious IP addresses. We decided not to include this functionality directly into the PROTECTIVE Node, but rather as a new module into an external system, NERD2 (developed and operated by CESNET, a PROTECTIVE member). The rationale for this is described further in D5.4.

SC-5 output is included in the Meta-alert and so it does not have a dedicated Prot-Dash UI component

4.8 SC-6: Sharing of Threat Intelligence

SC-6 output is included in the Meta-alerts and so it does not have a dedicated Prot-Dash UI component.

4.9 SC-7: Context Awareness Development UI

Data is input to the CA system via automatic fetching of inventory data and vulnerability data from external databases and Internet systems. It is also possible to input the definition of the relationships in the dependency graph for services, mission and asset. This information will be entered via a table and in the first release, Excel will be used for this purpose. The entered relationship data will be uploaded as a comma separated variable (CSV) file to the CA system. It is also possible to use the Web UI to upload data from a file straight into the CA system. Visualisation of the entered data can be useful for a number of purposes including assessing situational awareness. The particular data to be visualised is:

1. Inventory -Mission, service and asset- entities and their relationship;
2. Asset state management service and network ontologies;
3. Detailed information on entities and relationships;

To meet this criterion, a D3JS Force Directed Graph (FDG) View Provider has been implemented. The D3JS Data Provider retrieves data directly from the CA MAIR API. The Mapping Provider parses and formats the data to make it compatible with the FDG library. The graph is then rendered with a number of customisations to provide extra functionality for the user.

Under the “Context” link in the navigation menu of Prot-Dash, we can see two nested links. If we click on the first one “CDG Table”, we get a list of all of the context directed graphs that CA MAIR has access to. If we wish to create a widget to view the graph, we need not manually create it as was done in SC3. The user can just click on the button with the name of the graph that they wish to view. Looking at the

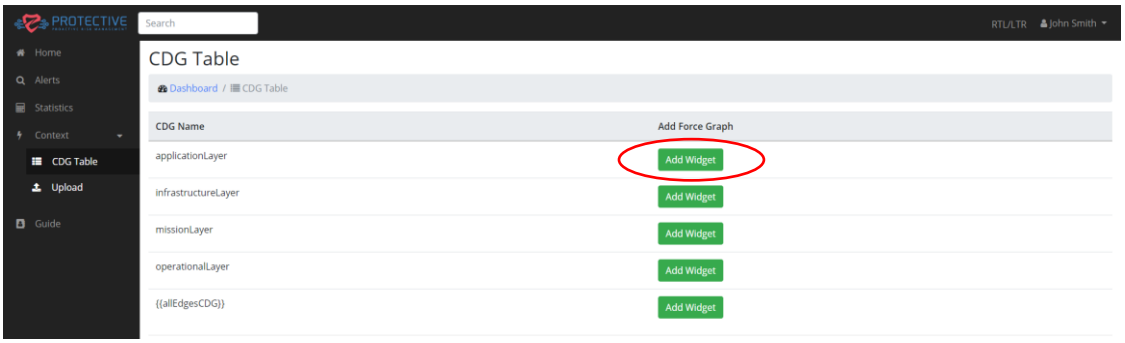


Figure 12: CDG Table

names of the graphs in the CDG Table, we can see that separate data about mission, infrastructure, application and operation is available. Clicking will cause a widget to be created with the content of the graph. We can see this in Figure 12.

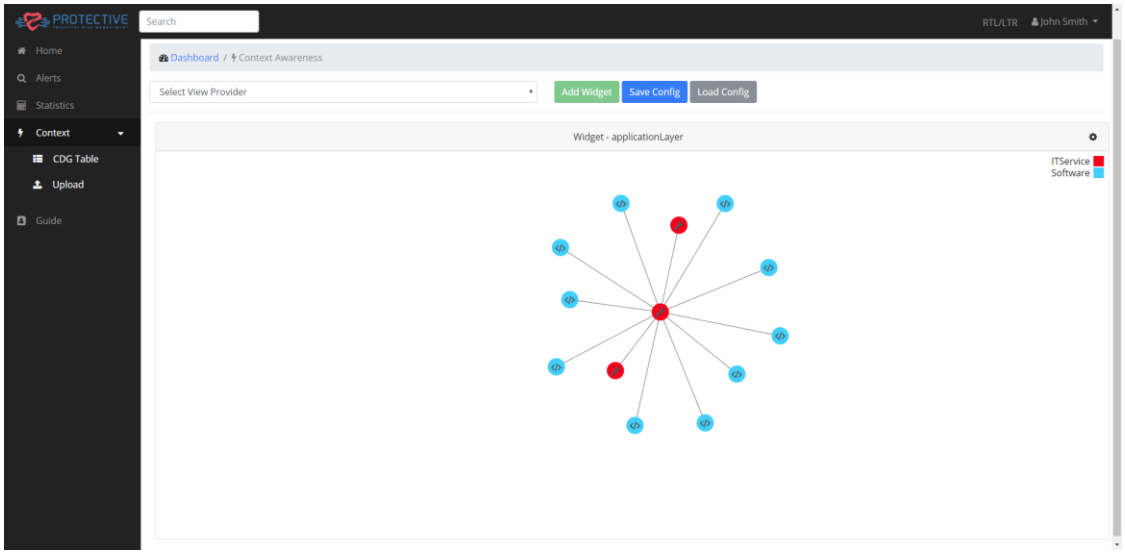


Figure 13: Force Directed Graph

The automatically rendered FDG is then shown in Figure 13. We have rendered the application layer of the system graph. The blue nodes represent software and the red nodes represent an IT service. The links show how they are dependent on each other.

The user can interact with the graph in a number of different ways. It is possible to drag the nodes into different positions. This is often useful if a node is obscured and we wish to isolate it. Using the mouse wheel, we can zoom into and out of the graph. This allows the user to have an overall look at the graph scape or have a much closer look. Notice also that there is a legend in the top-right of the graph. Clicking on the legend collapses all the nodes with the type clicked-on. We can see the result of clicking on the "Software" title in the legend in Figure 14. If we look closely, we can see that "Software" is "greyed-out". Clicking it again will undo the collapse.

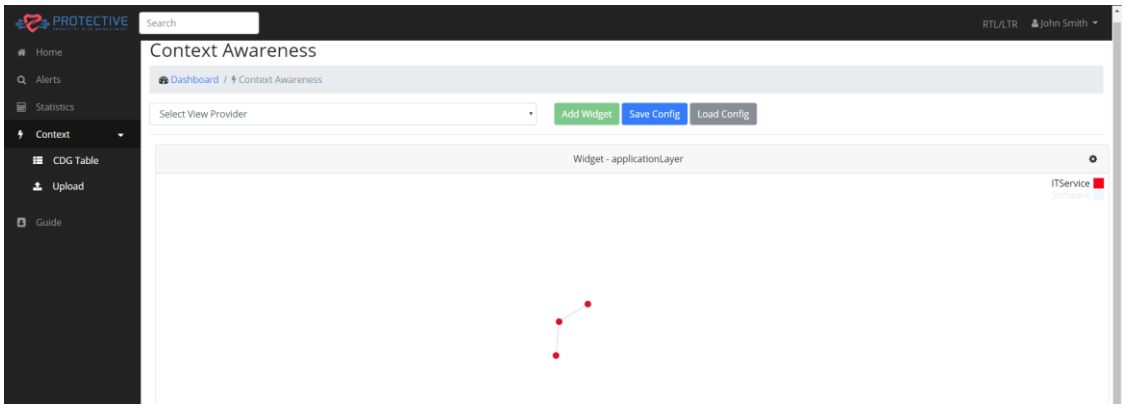


Figure 14: Collapsed Nodes

The act of hovering over the node or the link will display basic information about them. The name and type of the node is displayed. The source, target and weight of the link is displayed. Hovering will also trigger filtering of the nodes. If we hover over a node, only those nodes and links that are connected to this node will be focussed. The other nodes and links will be "greyed-out".

There will be cases where the application user requires more detailed information about the nodes and links in the graph. This is also possible using Prot-Dash. We must select "edit" from the widget dropdown menu and then click on the "View Provider" tab. A grey blank space with and empty JSON

string ({}) should be visible below the graph. Clicking upon a node or link populates this JSON string with detailed information about whatever asset was clicked. This is shown in Figure 15.

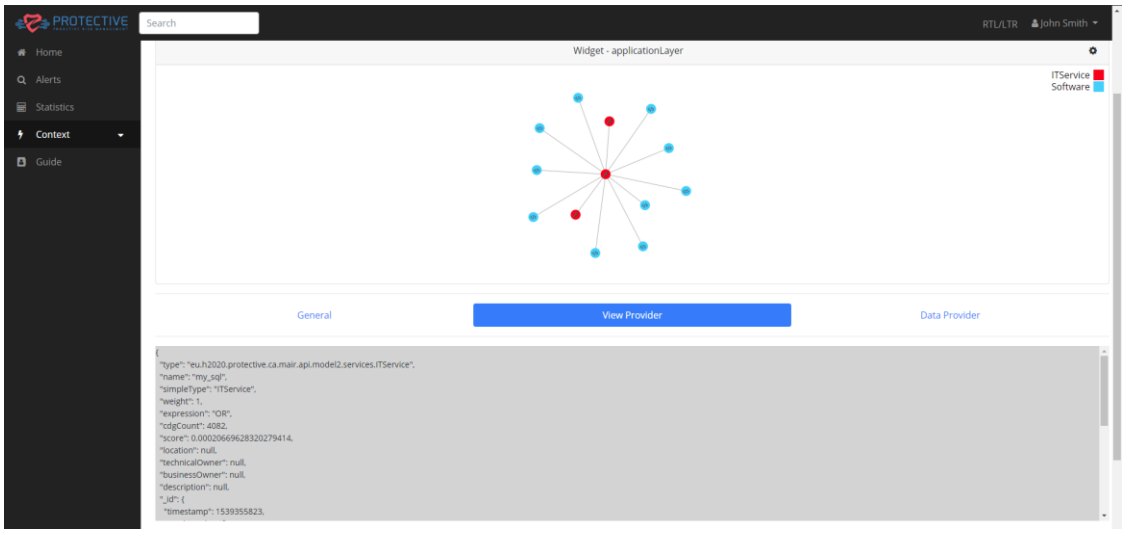


Figure 15: CDG JSON Detail

Adding data to CA MAIR is also possible from Prot-Dash. However, the data that is added must be in the form of a file with either the .json or the .cdgs extension. The file to be uploaded should contain information about a directed graph in valid syntax. If the syntax or graph structure is invalid, errors will be returned upon attempted uploading of the file. The errors are very useful and point to specific line numbers where the errors are present. The upload section of Prot-Dash can be by under Context > Upload. This is shown in Figure 16.

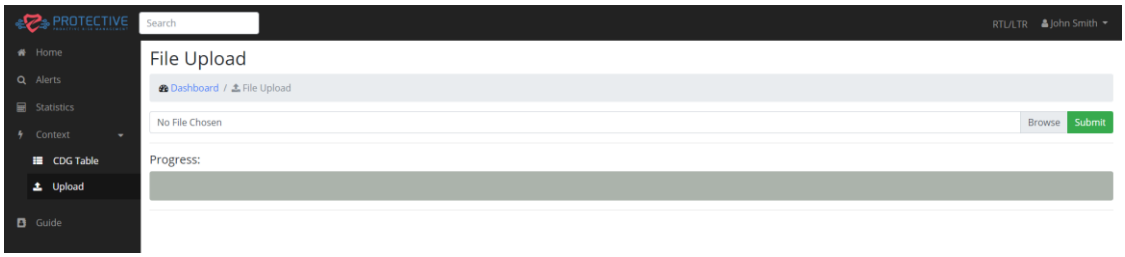


Figure 16: CDG file Upload

4.10 Scenario Independent Supporting Visualisations

To complement the search results, e.g., in SC-1, and to enable the operators to see the full alert information and to prototype meta-alerts visualisation, further UI views were developed and discussed with the operators during Pilot 1. The focus of these discussions was mainly to motivate input regarding which contextual information they would benefit from, before implementing them across the full system stack. The visualisations described below are using actual data from the local Mentat instance, but do not necessarily utilize all PROTECTIVE features, since they were developed in parallel. Further, certain annotations are requested from NERD, an external service which is used across PROTECTIVE as well.

The screenshots in Figure 17 shows some of the components that are visualised as part of a specific alert, as well as various details that have been annotated in order to contextualize the alert. The information extracted directly from the IDEA message, which represents alert, is organized as follows:

- Summary: Provides the operator with key fields of the alert, e.g., source, target and category

- Node: Information about the node that generated the alert and the software its running
- Description and Note: Short and long information about the alert

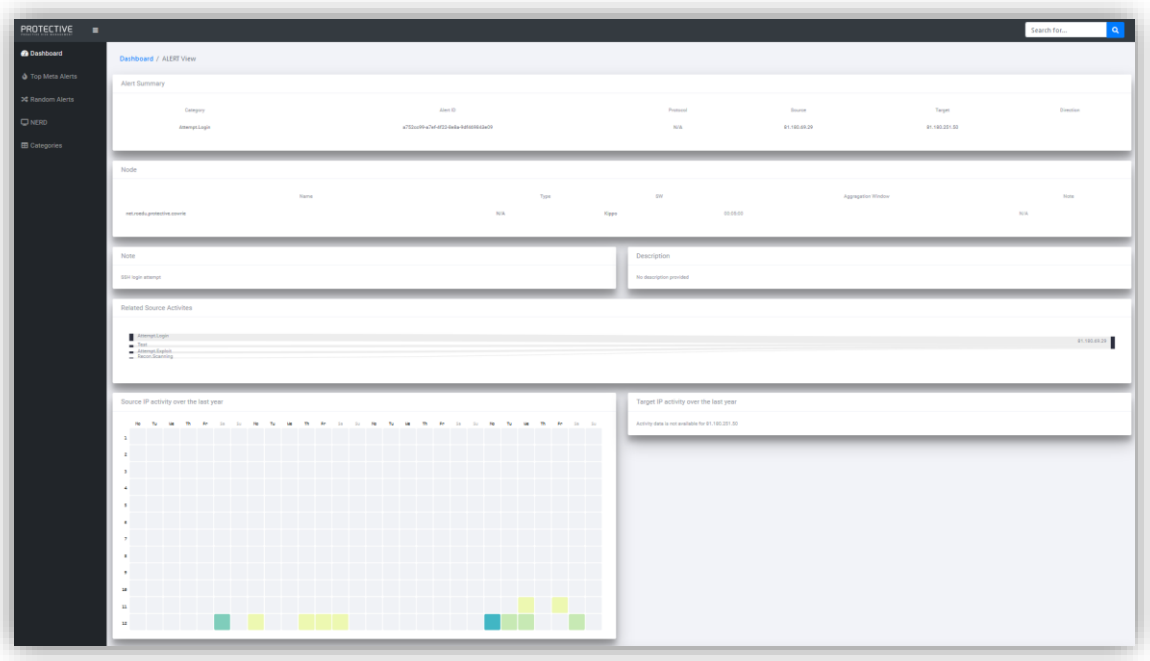


Figure 17: Alert view with annotations, zoomed out to capture more components

The annotations that are added to e.g., an alert overview, consists of, in the case that the information is available, additional information regarding the following aspects:

- Other activities performed by the source and destination IPs to provide the operators with a visual impression of how active either are. This information goes beyond the correlation time window in order to provide the operator with a broader view of e.g., past activities.
- Quality metrics of the alert, as calculated by the Trust module.

Detailed views for alerts, meta-alerts and IP addresses share the main blocks of information, except from the current focus. E.g., an alert includes information regarding relevant IP addresses or meta-alerts, while a meta-alert would reference the alerts that have been used to create it. The most relevant visualizations are elaborated below, and provide a description what they communicate.

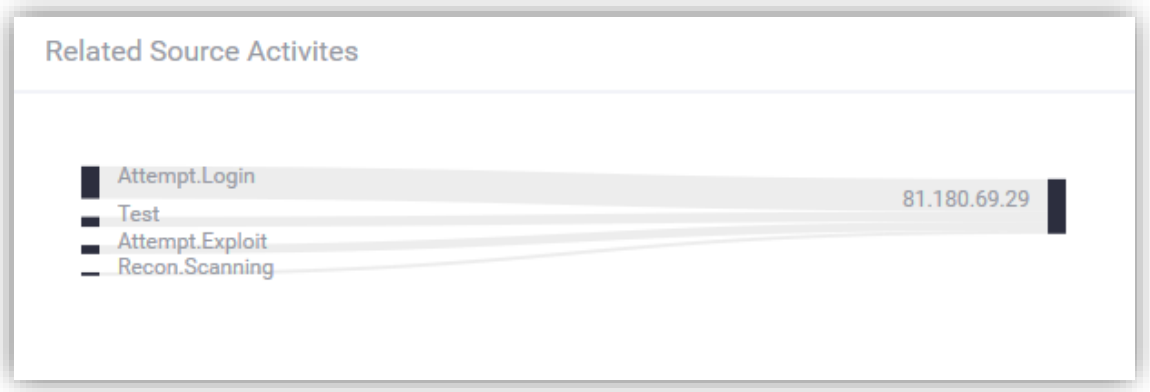


Figure 18: Detailed view of categories in which the source has been involved

Figure 18 shows which categories a given IP has been involved in and their relative frequency. E.g. in this case the shown IP frequently attempts to login into a remote host. This expresses the scope and type of activities the specific IP has been involved in, beyond the current alert.

In a similar way, Figure 19 and Figure 20 show which specific IP addresses the source **attacked** and which sensor **detected** the activity.

Figure 19: Relative IP activity across categories and the targets

While the main difference between Figure 19 and Figure 20 is where the information is coming from, i.e., the local Mentat database and NERD, they each emphasize which service (represented by the IP) the operator should investigate, and which sensor detected the events, respectively. Thus, one informs the operator about assets under attack, the latter informs the operator about the source.

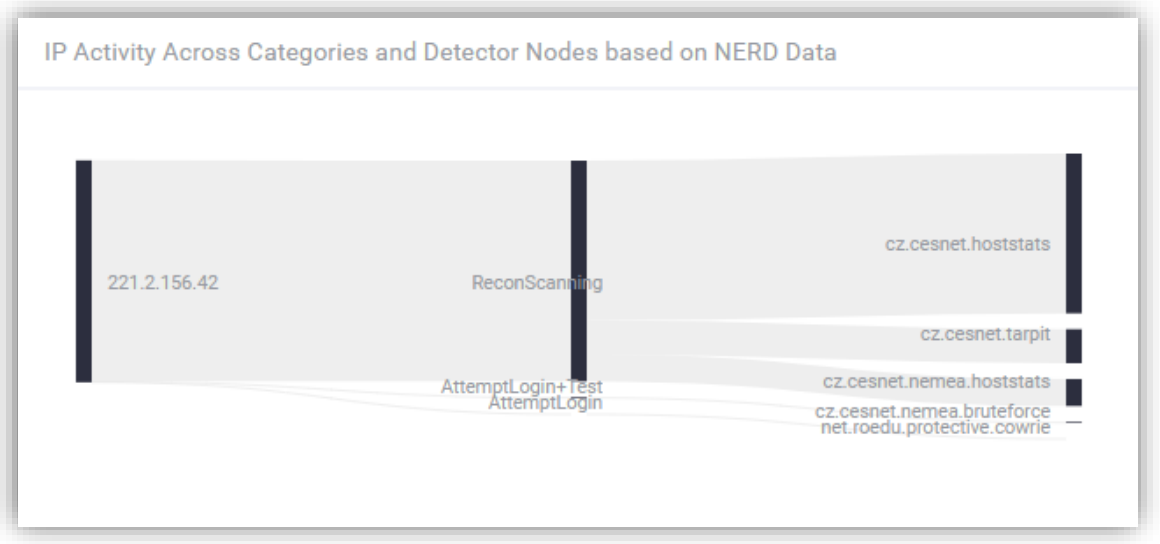


Figure 20: Relative IP activities across categories and the detectors

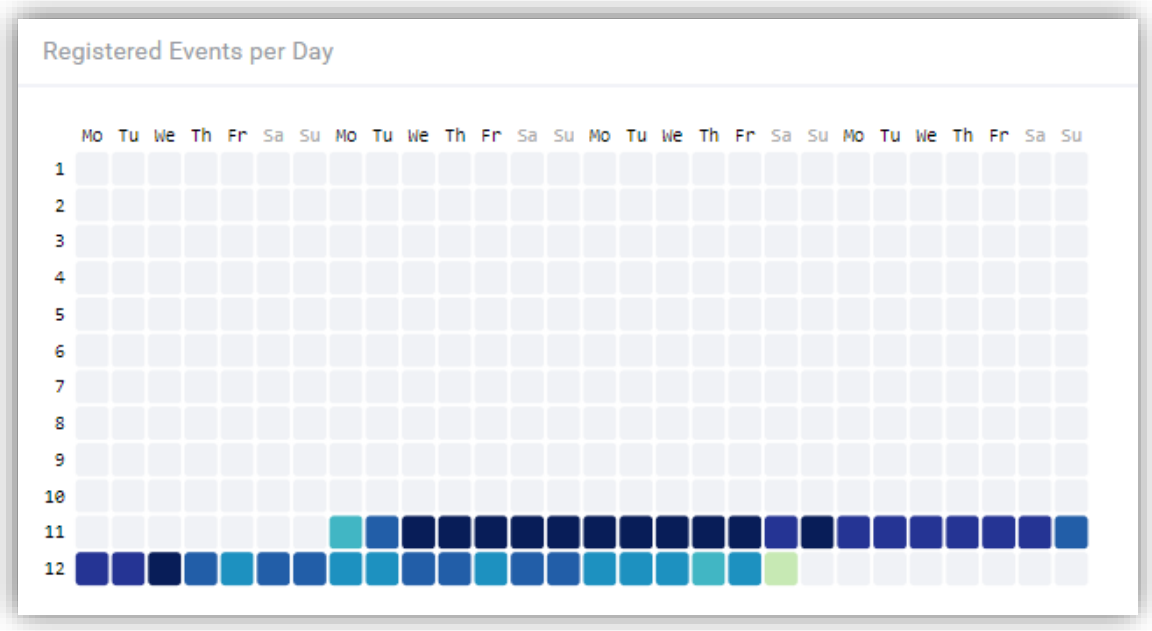


Figure 21: Frequency of IP activity over time

While the previous visualisation shows the type of activities, the activity information of a specific IP is also visualized over time, as exemplified in Figure 21. The motivation is to show the overall activity, when the IP was first registered as well as whether the activities have ceased. It gives the operator an impression of the duration of the malicious activities as well as if they are increasing or decreasing.

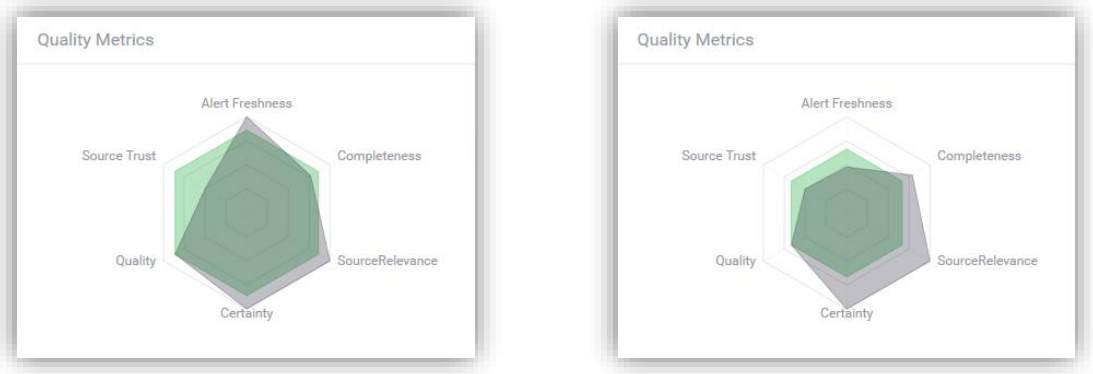


Figure 22: Main trust value - green - and the individual values - grey - plotted on a spider chart

One of the parameters which is used when prioritizing is the trust score. The trust score is comprised into a single value, from various other parameters. The visualisations in Figure 22 show how the detailed alert view can include the trust score and at the same time explain from which parameter the main impact is coming from. If needed, this allows the operators to refine their selection on which alerts to focus on, in particular if shown across multiple alerts, in a simplified manner.

In summary, the combination of having the detailed information of each alert, based on the content generated by the sensor itself, with annotation, enable the operators to get both the detailed view of the event and potentially relevant context such as the activity of an attacker, which complements the numerical values of the Trust component by providing details regarding what may have impacted the values. However, this approach has two limitations; 1) while additional context is a benefit, it tends to show more for trivial malicious events, e.g., random scanning or spam, 2) it may overshadow more

serious malicious events, as they are presented with less information. For the latter, the importance of being able to provide a metric regarding the criticality of a specific event. At the same time, the visualisations emphasize the dependency on having a broad range of information available, since this significantly emphasize certain activities, and leads to further insights. That said, some visualisations seem to be counterproductive, such as visualisations of the relative time of the events occurring in the alerts, since the timings contributed with limited information besides when the event occurred e.g., Figure 23, shows how the detection time of the event, the creation time of the alerts and the cease time of the events are practically identical.

Figure 23: Relative event time of events in an alert

Finally, one of the aspects that have been a strong focus in the detailed view and visualisations is the interlinkage of the provided information, enabling the operators to seek out additional information about different components. E.g., when investigating a specific (meta-)alert, the operator can go directly to the source or target IP information page, see related alerts based on category, the sensor that produced the alert as well as going backwards, such as when looking at a specific, easily be able to identify and investigate alerts where the IP was mentioned. The main relations are illustrated in Figure 24, where each top view may refer to either prioritized views, as discussed in SC-4, or operator configured components showing information relevant to them.

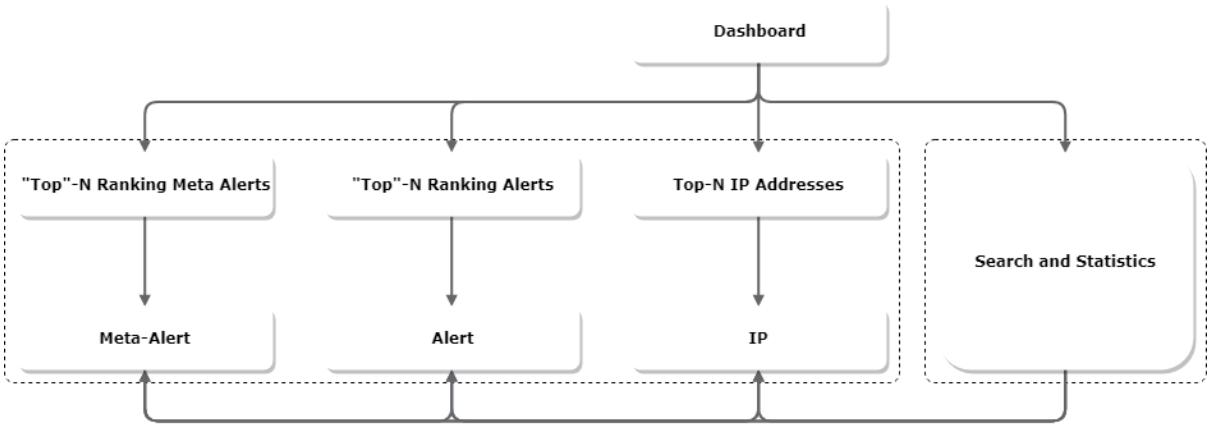


Figure 24: Relation of detailed views of alerts, meta-alerts and IPs in relation to search and analytics

5 Pre-requisites

Prot-Dash is delivered with the following components. It is designed to consume from both CA_MAIR and NEON.

- Prot-Dash Angular Web App source.
- Prot-Dash Angular Web App Docker images.
- Keycloak Server Docker image.

Prot-Dash source code, Prot-Dash Docker images and Keycloak Docker images are available at https://gitlab.com/protective-h2020-eu/viz/Prot-Dash-Angular

6 IPR Management

No background IPR has been used as input to this work and no inventions have been identified during the development.

7 Ethical Impacts

There are no ethical impacts arising from this function as none of the data is shared with other partners.

8 References

Angular. (2018). Angular. Retrieved October 22, 2018, from https://angular.io/

Bootstrap. (2018). Bootstrap · The most popular HTML, CSS, and JS library in the world. Retrieved October 22, 2018, from https://getbootstrap.com/

chart.js. (2018). Chart.js | Open source HTML5 Charts for your website. Retrieved October 22, 2018, from https://www.chartjs.org/

D3JS. (2018). D3.js - Data-Driven Documents. Retrieved October 22, 2018, from https://d3js.org/

Facebook. (2012). Facebook Social Graph. Retrieved October 27, 2016, from http://snap.stanford.edu/data/egonets-Facebook.html

GitHub. (2018). GitHub. Retrieved October 22, 2018, from https://github.com/

Google. (2018a). google mail - Google Search. Retrieved October 22, 2018, from https://www.google.ie/search?q=google+mail&rlz=1C1CHBF_enIE748IE748&oq=google+mail&aqs=chrome.0.69i59j69i60j69i65j0l3.3895j0j4&sourceid=chrome&ie=UTF-8

Google. (2018b). Google Maps. Retrieved October 22, 2018, from https://www.google.com/maps/@53.4157092,-7.9070578,15z

Keycloak. (2018). Keycloak. Retrieved October 22, 2018, from https://www.keycloak.org/

Neon, N. C. (2018). NEON | Next Century. Retrieved October 22, 2018, from https://nextcentury.com/project/neon/

plotly. (2018). Modern Analytics Apps for the Enterprise - Plotly. Retrieved October 22, 2018, from https://plot.ly/

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., & Mortimore, C. (2014). OpenID Connect Core 1.0 incorporating errata set 1. *The OpenID Foundation, Specification*.