Horizon 2020 Programme
Instrument: Innovation Action

Proactive Risk Management through Improved Cyber Situational Awareness

Start Date of Project: 2016-09-01

Duration: 36 months

D5.4 Threat Intelligence Community v3

Deliverable Details	
Deliverable Number	**D5.4**
Revision Number	**E**
Author(s)	**TUDA/GMV/CES**
Due Date	**1018**
Delivered Date	**24/10/2018**
Reviewed by	**AIT, OXF, RoEduNet**
Dissemination Level	**PU**
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.

Partner Roles

Contributing Partners	
1.	**TUDA (contributor)**
2.	**GMV (contributor)**
3.	**CES (contributor)**
4.	**OXF (reviewer)**
5	**AIT (reviewer)**
6.	**RoEduNet (reviewer)**

Revision History

Revision	By	Date	Changes
E	TUDA	24/10/2018	
B4	AIT	24/10/2018	Additional review of the final deliverable
B3	TUDA	24/10/2018	Figure change, polishing
B2	TUDA	23/10/2018	Addressed comments of AIT
B1	TUDA	23/10/2018	Addressed comments from OXF, RoEduNet
A	TUDA	18/10/2018	Draft version for review
A6	TUDA	16/10/2018	Fixes and additions for libraries and TI-Admin
A5	TUDA, CES	15/10/2018	Added CES-related content, various fixes
A4	TUDA	12/10/2018	Added TUDA-related content
A3	TUDA, AIT	08/10/2018	Incorporated AIT content
A2	TUDA, GMV	22/02/2018	Added GMV text
A1	TUDA	01/10/2018	First Draft

Abbreviation List

CERT	Computer Emergency Response team
CSA	Cyber Security Awareness
CSIRT	Computer Security Incident Response Team
EML	TheEmailLaundry
FMP	Future Misbehaviour Probability
IEP	Information Exchange Policy
JSON	JavaScript Object Notation
NREN	National Research and Education Network
SIEM	Security Information and Event Management
SME	Small to Mid-size Enterprise
TI	Threat Intelligence

Executive Summary

This deliverable (Threat Intelligence Community v3) documents the final version of the PROTECTIVE threat intelligence community. More specifically the document mainly emphasizes in the updated implementations of *TI Admin* and *TI Trust* components. In addition, the deliverable provides updates on the *TI Distribution* and *TI Analytics* components. The first chapter includes a short description of the TI Sharing architecture and different components and their features being delivered. Chapter 2 highlights the scenarios, requirements and parts of the architecture fulfilled by the TI Distribution, TI Analytics, TI Trust and TI Admin components. Chapter 3 provides the implementation details and pre-requisites for executing the software components. It also includes a selection of updated screenshots of the executed software. Chapter 4 highlights any ethical related impacts. In the last chapter, IPR impacts are briefly discussed and all third party (open-source) libraries that are used by the various TI community components are acknowledged.

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

List of Tables

1 Introduction

This document describes the third version (v3) of the PROTECTIVE Threat Intelligence (TI) community implementation. Figure 1 depicts the architecture of the centralized version of the PROTECTIVE community (as presented in the PROTECTIVE deliverable D5.1) and as it will be provided in the PROTECTIVE framework version that will be used in the Pilot 2 period.

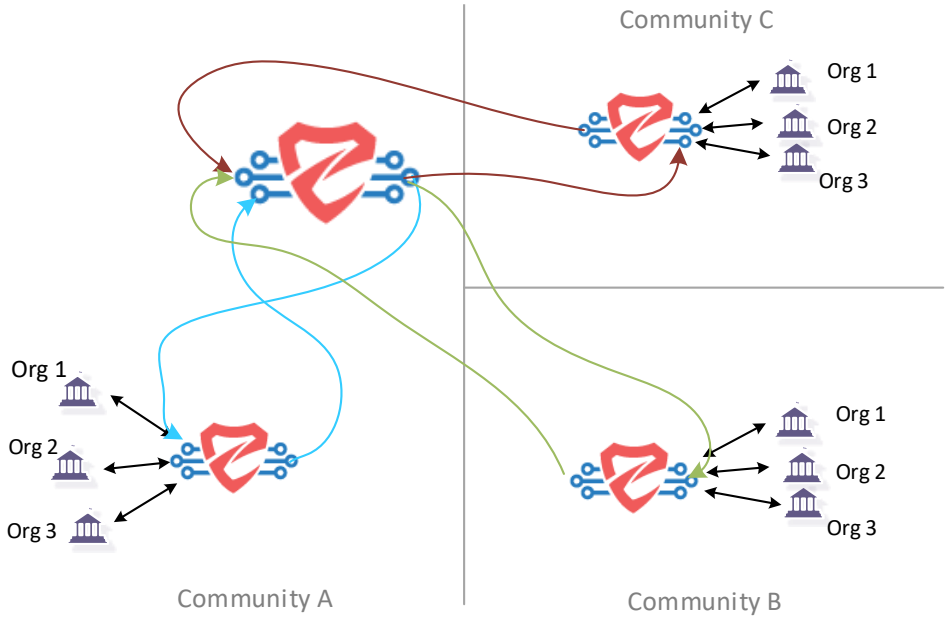

Figure 1: Centralized TI sharing architecture for PROTECTIVE Community

In particular, this deliverable provides the implementations of the following components that deal with the TI sharing among the PROTECTIVE communities.

- TI Distribution: this component includes a redefined sharing model, a set of connectors to ingest alerts from each of the constituencies in PROTECTIVE, and the implementation of certificate-based communication among the PROTECTIVE nodes.
- TI Analytics: this component is updated with a new data mediation and visualisation capability.
- TI Trust: this component includes the final implementation of the Trust component. The module is able to calculate the quality of the incoming alerts and the reputation of malicious entities (i.e., IP addresses).
- TI Admin: this component includes the implementation of the Admin component. This involves the implementation of Single Sign-On (SSO) for those parties who are authorized to access resources made available through the PROTECTIVE UI (i.e., PROT-Dash).

2 Scenarios, Requirements & Architecture

Scenarios, requirements and architecture are documented in Chapter 3 of the deliverable "D2.3 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

Figure 2 shows the scenarios for PROTECTIVE. Those (SC1, SC2, SC3, SC5 and SC6) highlighted in green have been completed.

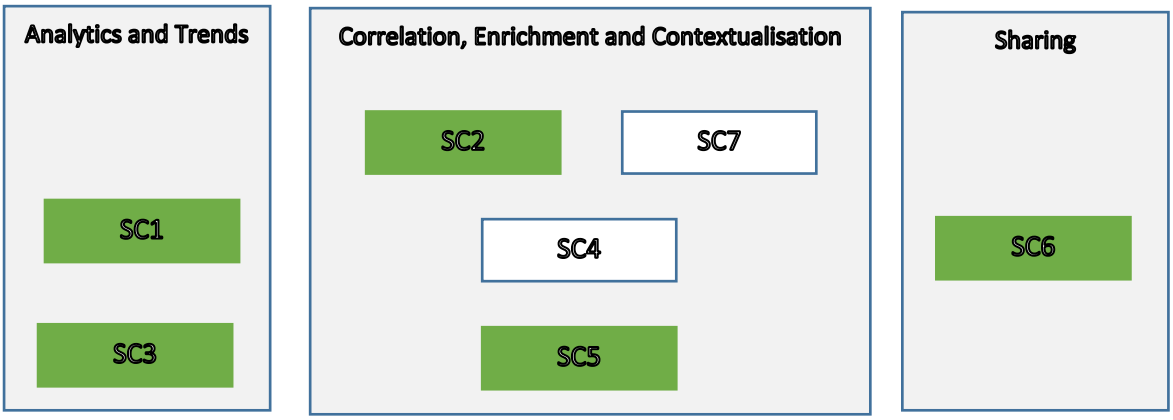

Figure 2: PROTECTIVE Scenarios

Scenario SC1 was delivered in deliverable "D5.2 Threat Intelligence Community v1". A new version of SC1 is now included as part of this deliverable to replace the original implementation which was based on the Mentat Hawat User Interface (UI). Regarding Scenario SC2, this deliverable provides the final version of the trust component. SC3 extends SC1 by providing a method for creating statistical trends for attacks based on historical data. Regarding SC5 (threat awareness), we have created a new module for an external system, NERD, which predicts future malicious behaviour of IP addresses based on advanced machine learning. PROTECTIVE nodes fetch results of this prediction from NERD. Lastly, regarding SC6, we have implemented several connectors to ingest alerts from different constituencies and enabled certificate-based alert sharing among the PROTECTIVE constituencies. With respect to SC6 in particular (but also relating to all scenarios SC1-SC7), we have integrated the UI with Keycloak and enforced Single-Sign-On with OIDC for authentication and OAuth2 for authorization.

2.2 Requirements

The following requirements have been completed (marked *done* or *partially* fulfilled) by this delivery.

2.2.1 Alert Prioritisation

ID	PR-02	Partially
Type	FR	
Slogan	PROTECTIVE MUST be able to prioritise threat data in terms of their reputation and trust level	
Rationale	The reputation of threat intelligence sources affects how the information should be treated. In particular, the reputation of a specific IP address may impact how the information related to said IP address has to be prioritised.	
Related Scenarios	SC-2, SC-4	
Related Requirements	TR-01	

Alert Correlation

ID	CR-01	Partially
Type	FR	
Slogan	PROTECTIVE MUST support the correlation of alerts sharing the same indicators	
Rationale	Alert correlation increases the information density over single events and provides a higher-level view on the current situation	

Related Scenarios	SC-2, SC-4	
Related Requirements	N/A	
ID	CR-03	Partially
Type	FR	
Slogan	PROTECTIVE must support enrichment of alerts by correlating them with internal and external information source	
Rationale	Ingested alerts are collected from various sources and formats, these may not provide full information (e.g., contain domain names instead of the IP), and PROTECTIVE must be able to complete this information to improve alert correlation accuracy.	
Related Scenarios	SC-2, SC-4	
Related Requirements	N/A	

Trust and Reputation

ID	TR-01	Done
Type	FR	
Slogan	PROTECTIVE must be able to estimate the reputation of "foreign" IPs	
Rationale	The reputation of a "foreign" IP may be used to prioritise which information needs to be processed first.	
Related Scenarios	SC-2	
Related Requirements	PR-02	

2.2.2 Analytics

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	

2.2.3 Interfaces for Ingestion, Sharing and User Interaction

ID	IF-03	Done
Type	NRF	
Slogan	PROTECTIVE MUST implement authentication to support that only authorised parties can receive and disseminate information into the system	
Rationale	Implementation of IF-09	
Related Scenarios	SC-6	
Related Requirements	N/A	

Ingestion of Information

ID	IF-04	Done
Type	FR	
Slogan	PROTECTIVE MUST support the addition and removal of information sources and sinks	
Rationale	To accommodate that the amount and type of sensors and threat intelligence sources will change over time, as well as the dissemination needs.	
Related Scenarios	SC-1, SC-2, SC-4	
Related Requirements	N/A	

Sharing of Information

ID	IF-08	Done
Type	NRF	
Slogan	PROTECTIVE MUST support push and pull mechanisms for information sharing	
Rationale	To be able to provide/dissemination the information as threat intelligence to different platforms	
Related Scenarios	SC-6	
Related Requirements	MP-10	

User Interfaces

ID	UI-01	Done
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	

2.3 Architecture

The PROTECTIVE node consists of three subsystems: i) Risk awareness subsystem, ii) Threat Intelligent Sharing (TI Sharing) subsystem, and iii) Alert flow processing platform. Figure 3, depicts the overview of a PROTECTIVE node, while Figure 4 shows the TI Sharing architectural overview. In this deliverable, we contributed implementations for the TI Sharing subsystem and alert flow-processing platform. As part of TI Sharing subsystem, we updated the implementation of the TI Distribution component. Additionally, we implement the TI Trust component as part of the Enrichment module. Outside of the PROTECTIVE node, we implemented a new module for the NERD system for threat awareness (realized by a threat prediction functionality). In the following, we provide the architectures and related documentations regarding the new implementations. The deliverable includes both updates of the Visualisation layer and the addition of a Keycloak server, which controls access to the node through SSO.

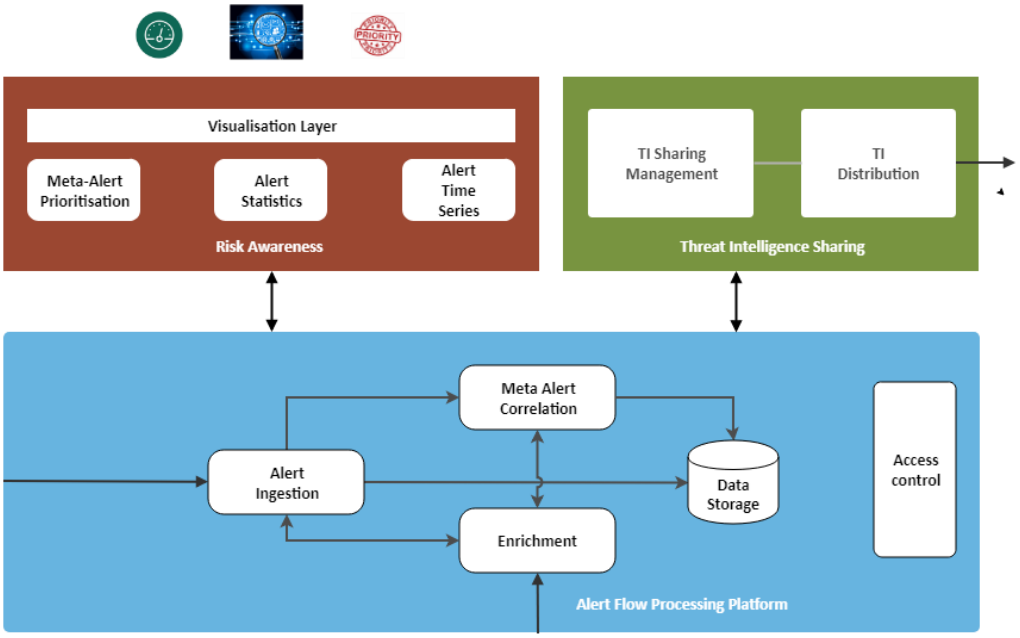

Figure 3: PROTECTIVE System Overview

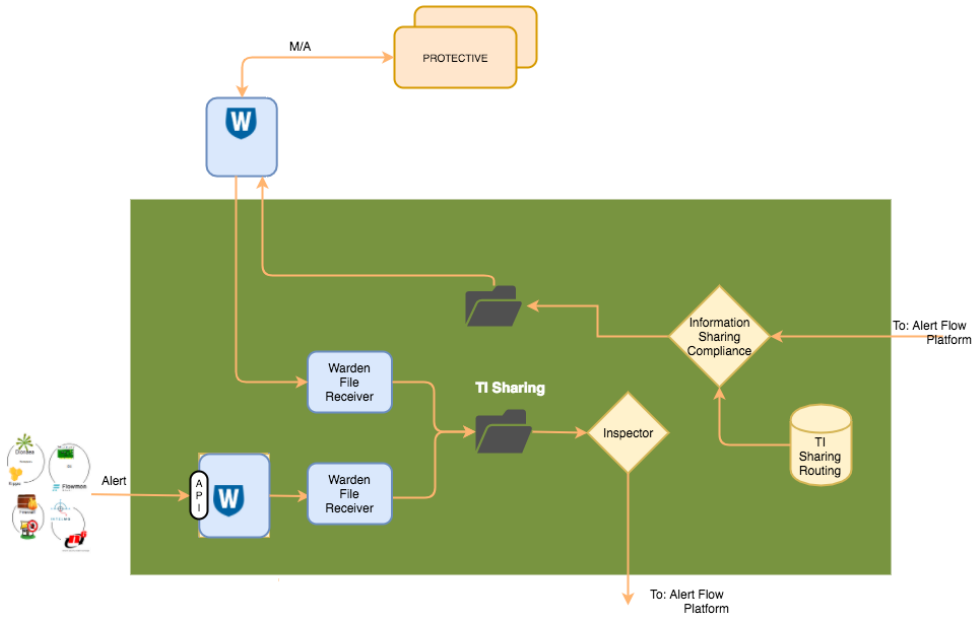

Figure 4: TI Sharing Overview

2.3.1 TI Distribution component

The first version of the TI Distribution has been delivered in deliverable "D5.2: Threat Intelligence Community v1" while the second version has been documented in "D5.3: Threat Intelligence Community v2". Here, a summary of the TI Distribution is given to assist the reader to acquire a holistic picture of the PROTECTIVE TI community.

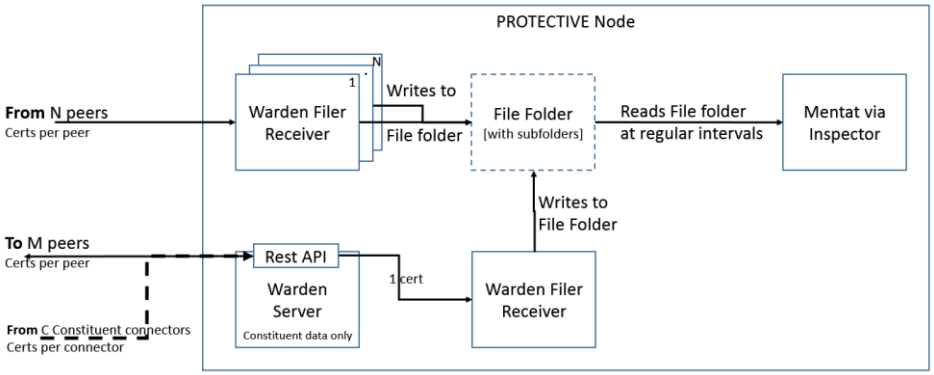

Figure 5: Alert flow architecture

Figure 5 shows how the alert flow is implemented in the PROTECTIVE System. The purpose of this architecture is to ensure that the events received from a peer node (i.e., member of the Community) or the centralized PROTECTIVE node (see Figure 1) are certainly the events of that peer node's connectors. Events received from other peer nodes are stored directly into the database. This ensures that no duplicated or same events are sent recursively through all the peers. This flow architecture is divided into local flow and centralized.

- Local/Constituency flow: the local flow consists of all the events from a constituency's connectors (described in Section 2.2 of deliverable "D6.2: PROTECTIVE Framework Implementation v1"). These events are ingested via the local Warden server API into local Warden Server and then, are sent to PROTECTIVE using a Warden Filer Receiver that is in charge of receiving the events from local Warden and storing them in a File Folder. Finally, PROTECTIVE reads this folder at regular intervals, picks the events, and stores them in its database.
- Centralized flow: in contrast to the previous versions of the PROTECTIVE architecture (cf. deliverable D5.3), the final version will illustrate a centralized architecture (see deliverable D5.1). In the centralized architecture of PROTECTIVE (see also Figure 1), a Community controls what is sent to the central PROTECTIVE server but not what is sent to other communities. At a glance, in the centralized flow, the events from the central PROTECTIVE server are received via Warden Client Receivers (one per peer). Received events are stored in the same File Folder as in the Local flow. Events are then read by PROTECTIVE and stored in its database.
- Implementation of certificates: we have enabled the secure (HTTPS) communication using self-signed certificates for the Warden server. This means that to be able to connect with other Peer nodes or to allow connectors to send data to the Warden Server, certificates are needed for the communication between the Warden server and each Peer nodes' or connector Warden Filer receiver/sender. The internal Warden file receiver that receives events from local Warden Server also works with certificates.

At the inspector module, there are rule to check against whether personal data exists in the candidate CTI that is about to be shared (prior to distribution). We refer to this part of inspector as the information sharing compliance module. The rule takes the form of the following:

- **Name** – this is the name of the rule.
- **Condition** – this needs to be met before an action can take place.
- **Action** – this enforces information to be shared as expected. The actions relate to anonymization, pseudonymization and dropping alerts entirely.

The rules are stateless in that they match a potential misuse. If the misuse is present, then the ISC enforces GDPR compliance through an action. For instance, in the case of a 'hotmail address' about to be shared, the email address would be anonymised.

2.3.2 TI Analytics component

The first version of the TI Analytics has been delivered in "D5.2: Threat Intelligence Community v1". The component has been improved with a new visualisation and data mediation framework; this new visualization framework will replace the previous interface used for SC1 implementation in the previous versions of this deliverable (cf. D5.2 and D5.3).

The new framework provides a more customizable users' dashboard, as it allows adding and removing widgets with custom graphs and perform custom queries to the database.

Threat awareness component

The threat awareness 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, NERD[1] (developed and operated by CESNET, a PROTECTIVE member). This is practical because NERD already keeps profiles of malicious IP addresses with all the information needed for the prediction, while that profiles are based on alerts from Warden – the same as used in PROTECTIVE (currently the CESNET's Warden server is used, but it will be connected to the PROTECTIVE's centralized Warden when ready). In addition, NERD also uses several other data sources (e.g., blacklists, DNS, WHOIS, geolocation, etc.). Figure 6 depicts the high-level architecture and communication between a PROTECTIVE Node and NERD.

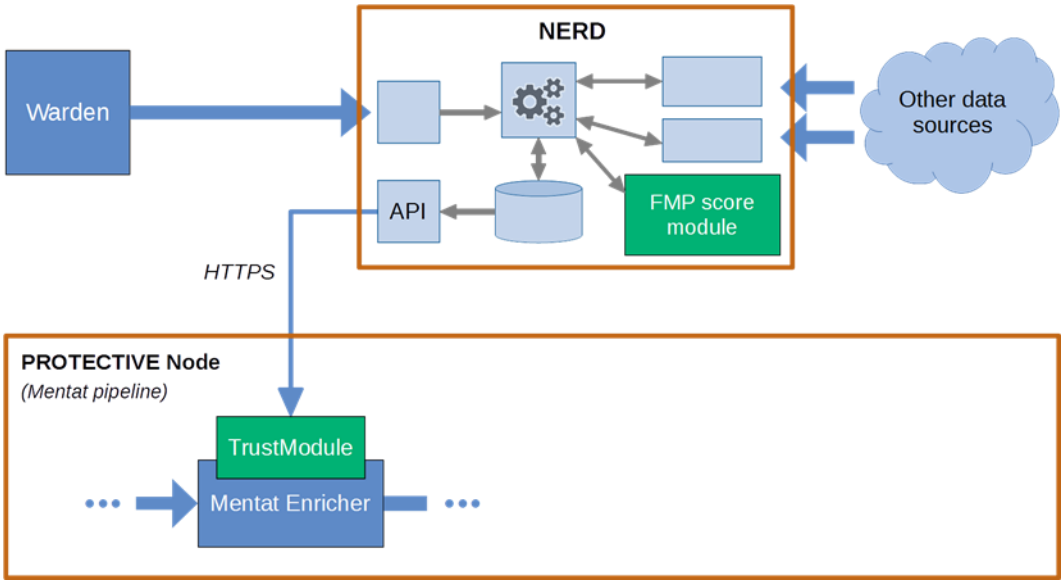

Figure 6: PROTECTIVE node and NERD communication pipeline

The new NERD module uses information about malicious IP addresses in NERD's database and a machine-learning model to compute the so-called Future Misbehaviour Probability (or FMP score). That is, the probability that a given IP address will perform a malicious activity (of a specific kind) in

[1] NERD – Network Entity Reputation Database, https://nerd.cesnet.cz

the next 24 hours. This score is stored as part of the IP address profile in NERD and thus made available via the existing HTTPS-based API.

The TrustModule in a PROTECTIVE Node fetches this score for each IP address and stores it as the *EntityReputation* field into the IDEA messages. The new *FMP score* thus replaces the *reputation score* mentioned in earlier deliverables, which was a similar concept, but used a simpler mechanism instead of machine learning and did not involve explicit prediction of future threats.

For the pilot, as well as for future practical usage, it is planned to connect all PROTECTIVE nodes to the main instance of NERD (operated by CESNET as a free service not only for the PROTECTIVE project), which receives data from the central Warden server. This saves both computational resources and number of requests made to external systems (like queries to blacklists, DNS, WHOIS databases, etc.). However, if necessary, it is also possible to deploy a separate instance of NERD along with each PROTECTIVE node and connect it to this instance instead.

The FMP score provides an estimation of the probability of future alerts. This is performed by using one of the best current machine-learning (ML) methods, Gradient Boosted Decision Trees (GBDT). The input for the ML model is a feature vector, representing an IP address, consisting of various features. These include the number of alerts about an IP address received within the last day and week, the number of detectors that reported this IP address in the last day/week, total volume of the reported attacks, average time interval between previous alerts and time from the last alert. The features are computed separately for different categories of alerts and include information about neighbouring IP addresses (same /24 prefix). Information about presence of the IP on several public blacklists is included as well together with several tags guessing the type of IP (statically or dynamically assigned, NAT, VPN, Tor exit node, etc.).

Given all this information, the model is trained to predict the probability that there will be an alert (of specific type) reporting the given IP address within the next 24 hours. The training runs on historical data from three months and it is planned to periodically re-train the model on recent data[2].

2.3.3 TI Trust component

The TI Trust component, depicted in Figure 7, is implemented as part of the Enrichment subsystem under the Alert Flow Processing platform. The specification of the TI Trust Component was proposed in deliverable "D5.1: Threat Intelligence Sharing: State of the Art and Requirements". The component consists of a plugin (TrustEnricher) for the "mentat-enricher.py" and a Python module called TrustModule. It provides the functionality for calculating the alert quality as well as the entity reputation. The first version of the TI Trust component provides the following features.

The TrustEnricher plugin extends each alert within the pipeline by parsing it and appending two separately calculated scores using the functionality of the TrustModule. This adds the AlertQuality and EntityReputation fields to the IDEA format of the alert. Afterwards the enriched alert is forwarded to the next step of the pipeline for further processing.

[2] Details regarding the specifics of the machine learning algorithms, as well as the evaluation of the model, will be published on a scientific journal, as part of the PROTECTIVE dissemination plans.

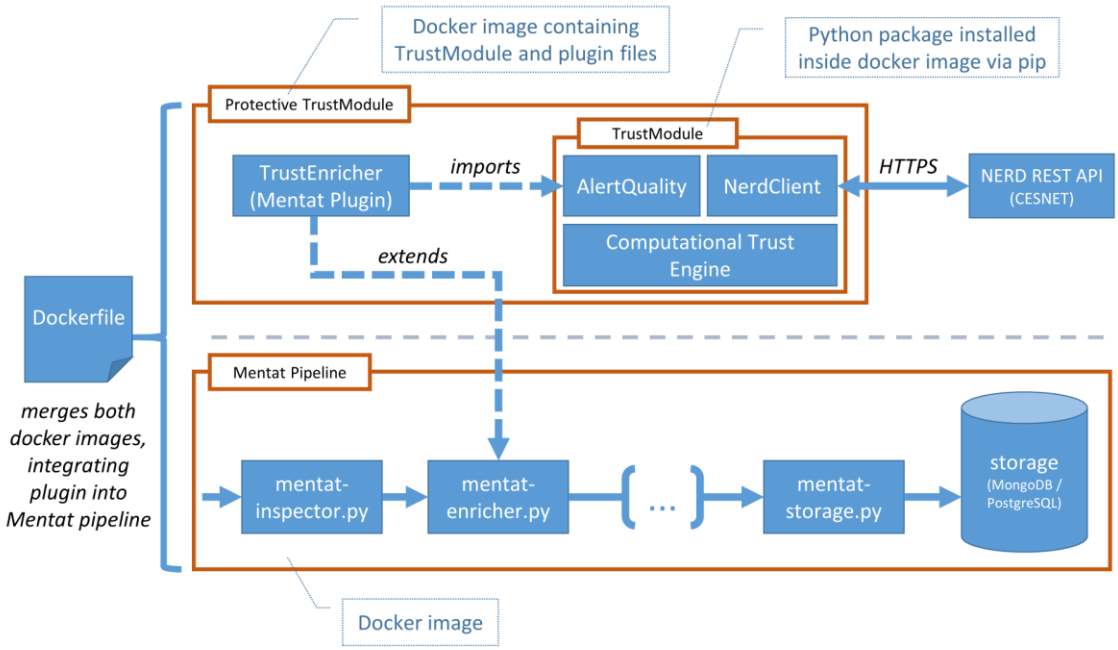

Figure 7: TI Trust Component

The python TrustModule consists of the following three submodules:

- **AlertQuality:** it parses the incoming alert and tracks persistent data regarding IP address recurrence and source trustworthiness in order to calculate the alert quality score using CertainTrust.
- **NERDClient:** it is a HTTP REST Client, which queries the NERD API at CESNET (PROTECTIVE Constituency) to receive the entity reputation score.
- **CertainTrust** (Computational Trust Engine): it provides the functionality to calculate the alert quality.

As part of the TrustEnricher, the following fields are added to the IDEA-formatted alert.

```
"AlertQuality": {
   "Score": <float>,
   "Inputs": {
      "Completeness": <float>,
      "SourceRelevance": <float>,
      "AlertFreshness": <float>
   },
   "Opinion": {
      "Quality": <float>,
      "Certainty": <float>,
      "SourceTrustworthiness": <float>
   },
   "IpRecurrence": <int>
},

"EntityReputation": <float>
```

Note that all the floating-point numbers are in the range between zero and one.

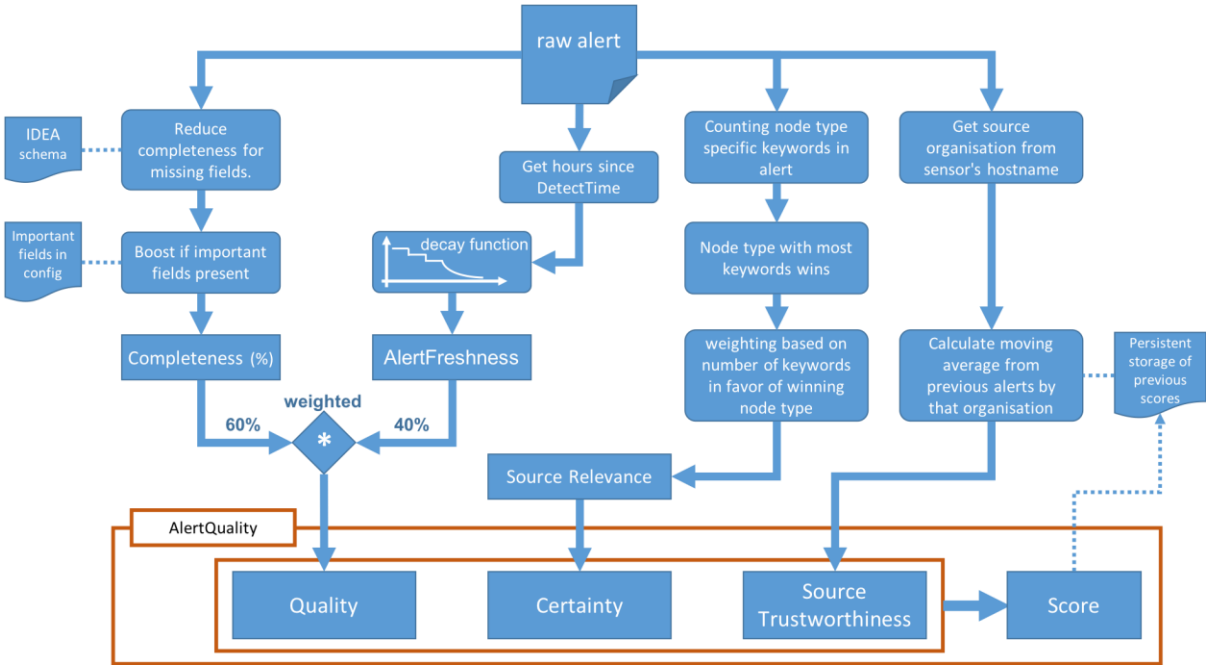

Figure 8: TI Trust Computation Model

Before the alert quality is determined, three intermediate results are calculated as described in the following:

- **Quality**
  The alert's basic quality is a weighted mean of the completeness and the alert's freshness. The weights are set in the module's configuration[3].

  - **Completeness**
    Upon comparing the raw alert to the IDEA schema definition, completeness is reduced for each missing field. Important fields specified in the configuration (e.g., source IP address, category, node type) boost up the completeness value if they are present in the alert.

  - **AlertFreshness**
    The DetectTime field of the alert is used to convert the number of hours since detection to a value between zero and one that is named AlertFreshness. A decay function is used which consists of a step function for relatively short timespans (i.e., more recent alerts) followed by an exponential decay function. Based on the number of intervals set in the configuration the dynamic interval sizes are calculated:

    $$IntervalSize_x = round(StartIntervalSize \cdot x^{1.25})$$

    Where $1 \leq x \leq IntervalCount$ is the number of the interval and $IntervalSize_x$ is its upper limit. Within these time intervals, the AlertFreshness is calculated as follows:

    $$AlertFreshness_{steps} = 1 - 0.5\left(\frac{x}{IntervalCount - 1}\right)^2$$

    Where x represents the hours passed since detection time.

    All alerts with a detection time outside the calculated time intervals receive an

[3] The default configuration and further documentation can be found in the GitLab repository: https://gitlab.com/protective-h2020-eu/enrichment/TI-trust/trustmodule

AlertFreshness value assigned according to the exponential decay function:

$$AlertFreshness_{exponential} = 0.5^{\,DecayFactor \cdot x + 1}$$

- **Certainty**
  For each possible node type a list of specific keywords (e.g., “honeypot”) is specified in the module’s configuration along a value for this type’s relevance. Based on the number of keyword occurrences in the alert’s node field, the source relevance is determined and then combined with the result of a voting. This results in the certainty value.

- **SourceTrustworthiness**
  The alert’s source organisation is determined by the prefix of the sensor’s hostname, which is the reverse top-level domain. Scores from previous alerts are persistently stored per organisation to calculate a moving average named SourceTrustworthiness, which is used as weight for the computation of the final score.

The final score representing the alert quality is calculated from the following equation:

$$AlertQuality = Quality \cdot Certainty + SourceTrustworthiness \cdot (1 - Certainty)$$

Intermediate results and the final score are added to the alert in the last step all inputs.

2.3.4 TI Admin Component

The TI Admin component, pictured in Figure 9, shows an overview of how the authentication and access control mechanisms operate. It is an implementation of the access control component illustrated in Figure 3.

- **Authentication:** From the browser, the user enters their credentials in to the UI. These credentials are sent to the Keycloak server. On authentication success, Keycloak responds with a bearer token.
- **Authorization:** The user will have access to various parts of the UI, depending on what their access level is. Keycloak defines roles and access levels. Users’ access level is verified when the bearer token is sent to Keycloak’s Entitlement API. This API responds to the UI with a Requesting Party Token (RPT). This contains details on what the user is permitted to access (or not) as the case may be.

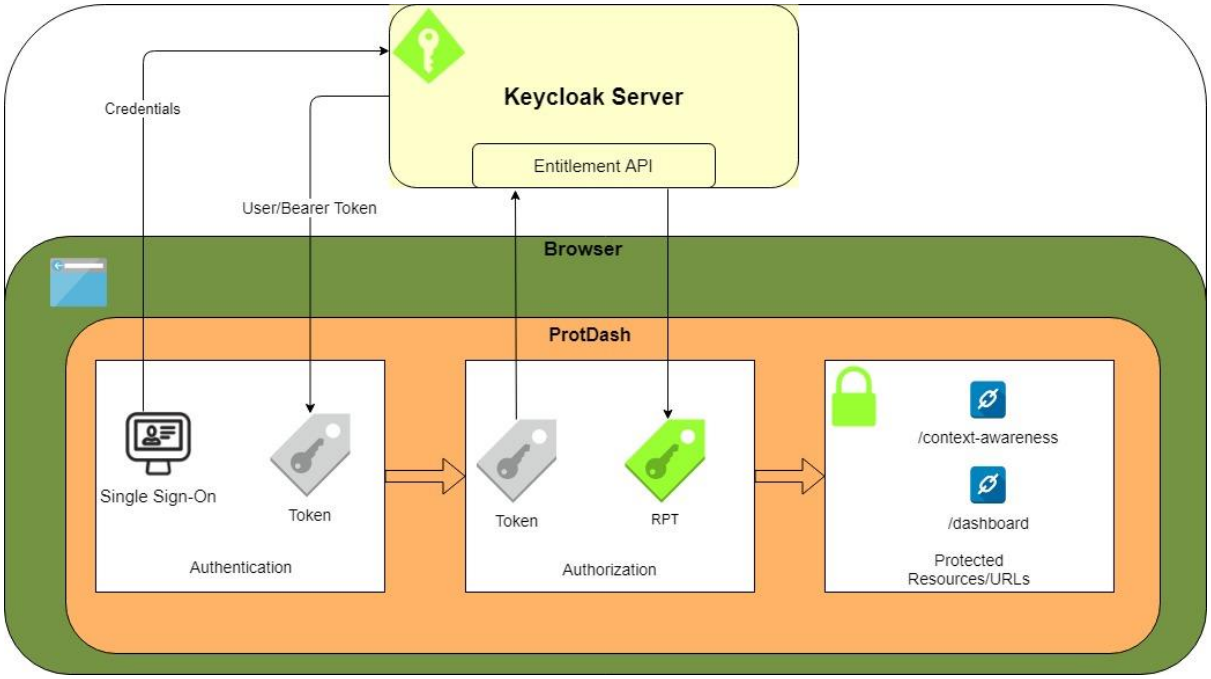

Figure 9: TI Admin Component

3 Implementation

3.1 Prerequisites

TI Distribution

In order to run the framework we recommend the following minimum hardware specifications:

- Operating System: Ubuntu 16.04 LTS Server
- HDD: 320 GB
- RAM: 34 GB

The framework can be ran on both bare-metal hardware and a Virtual Machine (VM). The solution also supports Apple MacOS and Microsoft Windows 10. In order to install all components, the machine (or VM) must have git, docker and docker-compose already installed, as all the components are stored in GitLab and most of them run as docker containers. The machines that are going to run the connectors must have Python 2.7+ installed.

TI Trust

The trust enrichment functionality for IDEA alerts is provided as a python module named TrustModule, which can be installed and used stand-alone. It requires Python 3 and the installation of some third party dependencies via the python package manager (pip). A full list of these dependencies can be found in the GitLab repository (see Section 3.2).

For integration of the trust enrichment in the pipeline a plugin named TrustEnricher can be deployed as part of any Mentat installation. The plugin imports and calls the required python functions of the TrustModule to enrich all alerts processed within the pipeline according to the default configuration. A docker image of the bare plugin is provided as well as a modified Mentat docker image that

integrates the aforementioned image. The installation and optional configuration are specified in the installation instructions (see Section 3.2) as well as the TrustEnricher repository.

Note that the EntityReputation score can only be added if an access token for the NERD API at CESNET (PROTECTIVE Constituency) is obtained and placed in the configuration file before rebuilding the docker container according to sample configuration file and the installation instructions in the repository. This does not affect the computation of the AlertQuality score.

As part of the trust enrichment, the number of recurrences of any source IP addresses seen from incoming alerts are tracked and stored on the file system. This is also the case for the source trustworthiness, which is used as a baseline specific to each organisation and is calculated from previous alerts. To ensure persistent storage of this data both file paths can be changed in the configuration file in order to save them in a docker volume.

TI Analytics

In order to successfully run the new improved TI Analytics software for SC1, the main requirements are to have an application server (e.g., Tomcat) where we must have the compiled package that will provide us the tools to connect to the database and to the queries we want to make to the database. In addition, we will also need to have another application running on the same application server or another (e.g., node) that will be in charge of providing the frontend (visualization component) to the user.

In order to use the **threat awareness** (threat prediction**)** functionality, an instance of the NERD system with the new threat-prediction module has to be deployed and available to the PROTECTIVE node.

The machine-learning model used in the module, GBDT, is implemented by a library called xgBoost (https://github.com/dmlc/xgboost). Besides this, NumPy and some additional Python packages are required by the module. NERD itself requires a lot of other Python packages and software to be installed, as documented at https://github.com/CESNET/NERD/wiki/Installation-and-running, but this is normally irrelevant to the operator of a PROTECTIVE node, as NERD is considered to be an external system to which PROTECTIVE node connects. The only requirement on the side of PROTECTIVE Node is the TrustModule installed and configured with a NERD API key (see above).

TI Admin

The Keycloak server is implemented in the same manner as the other elements of the Protective node. It is designed to be run alongside the other components and is dockerized. One requirement is that the UI environment be configured to know where the Keycloak server is deployed in order to interact with it for access control purposes.

3.2 Execution

TI Distribution

For the PROTECTIVE system, an automated installer has been developed. This installer runs Mentat, Warden, Context Awareness and the receivers for the data flow. This installer is available in PROTECTIVE's GitLab. An installation guide for the PROTECTIVE node is available in the repository where the installer is: https://gitlab.com/protective-h2020-eu/protective-node-installer . The guide includes instructions for installation of all the components.

The modules included in this second version of the system are:

- For the core of PROTECTIVE, Mentat and Warden:
  https://gitlab.com/protective-h2020-eu/cp/Mentat/Mentat-config
  https://gitlab.com/protective-h2020-eu/ti-s/Warden/Warden
- For the connectors, there is a folder for each one connector all the necessary to run them:
  https://gitlab.com/protective-h2020-eu/ingestion/extraction

TI Trust

The source code and installation instructions for the TrustModule and the TrustEnricher plugin as well as a sample configuration are available in the following GitLab repositories and can be installed as part of the PROTECTIVE node according to the installation guide.

- **TrustModule** (stand-alone python module):
  https://gitlab.com/protective-h2020-eu/enrichment/TI-trust/trustmodule
- **PROTECTIVE TrustModule** (including TrustEnricher plugin):
  https://gitlab.com/protective-h2020-eu/enrichment/TI-trust/protective-trustmodule

The TrustEnricher plugin provides a docker image. The installation guide before building the docker container and integrating it in Mentat container.

TI Analytics

The visualization layer used by PROTECTIVE for this phase has been grouped into a single component, PROT-Dash. This component is a web dashboard that provides the tools to accomplish with SC1 and SC3. This dashboard allows searching alerts, making custom graphs or time-series statistics and calculating the trend following the Exponentially Weighted Moving Average (EWMA) algorithm. Below, in Figure 10, PROT-Dash home is shown.

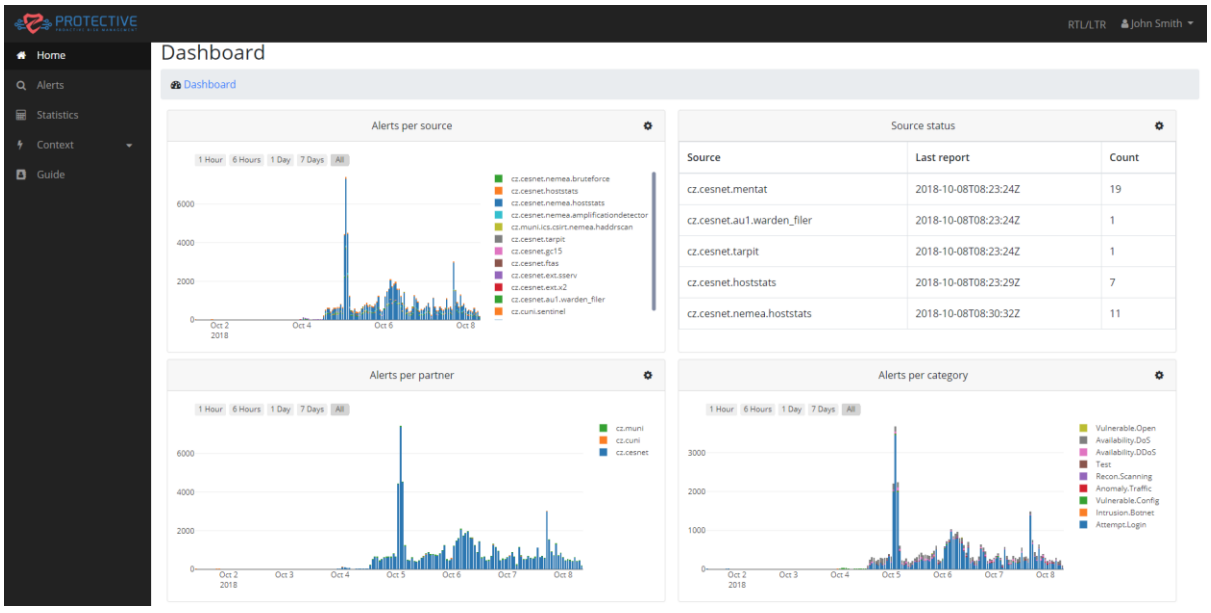

Figure 10: PROT-Dash Home

Figure 10 contains a customized graph showing the alerts per day received in 1 month, with a trend line computed by the EWMA algorithm.

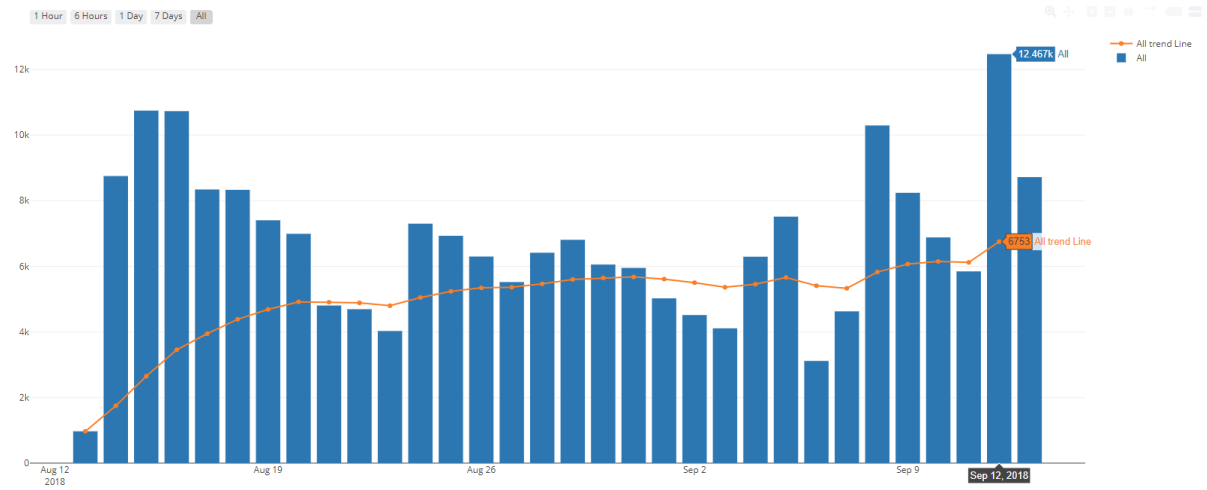

Figure 11: Custom Trend Line graph example

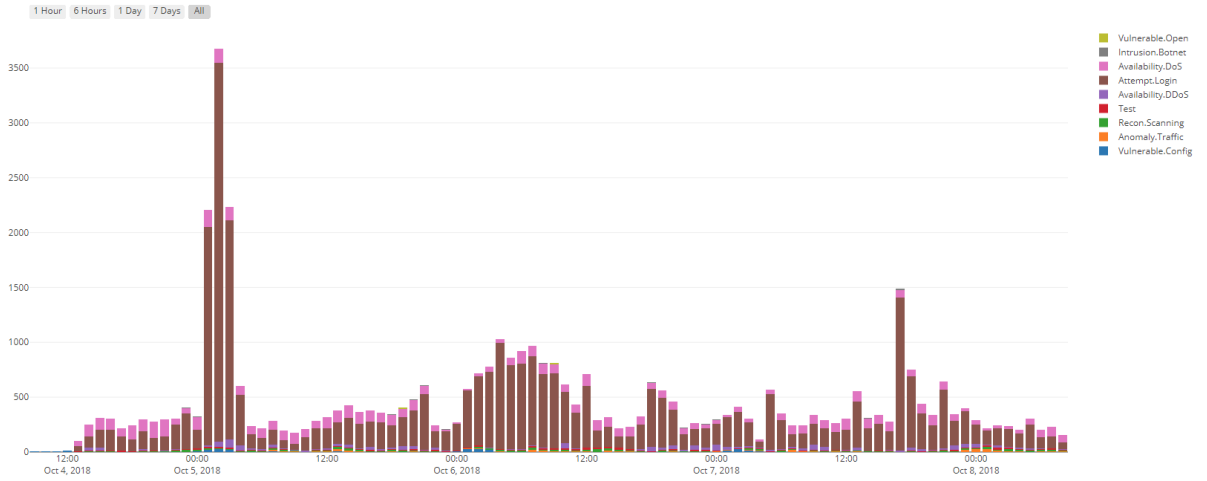

Figure 12: Custom grouped graph example

Finally, in Figure 12 shows alerts grouped by Category and per hour. For further information about all the functionality and configuration of PROT-Dash, see D6.9 PROTECTIVE System v2.

Threat Awareness

The module for threat awareness (threat prediction) is included in NERD repository at https://github.com/CESNET/NERD/.

It is not part of PROTECTIVE node installation as it is normally deployed as part of the main NERD instance operated by CESNET and each PROTECTIVE node should connect to it as to an external data source. In case a PROTECTIVE node operator want to deploy its own instance of NERD, including the threat prediction module, it can be done by following the NERD's documentation (https://github.com/CESNET/NERD/wiki).

Normally, the only thing needed to do on the PROTECTIVE side is to configure the API key in configuration of the TrustModule. The FMP score is then added to IDEA messages and can be consumed by the prioritization module or shown to the user in GUI.

TI Admin

Most of the work in TI Admin so far has been in the investigation and integration of Keycloak as both a SSO and authorization solution for both the UI and Protective.

Authentication

Keycloak uses OpenID Connect (OIDC) for authentication. This authentication protocol is an extension of OAuth 2.0. OIDC makes use of JSON Web Tokens (JWT) as illustrated in Figure 9. JWT's serve to encrypt data in a web-friendly and compact way.

As regards OIDC, there are generally two types of use cases. In the first of these, an application asks the Keycloak server to authenticate a user for them. After a successful login, the application will receive an identity token and an access token. The identity token contains information about the user such as username, email, and other profile information. The access token is digitally signed by the realm and contains access information (like user role mappings) that the application can use to determine what resources the user is allowed to access on the application. The second type of use cases is that of a client that wants to gain access to remote services. In this case, the client asks Keycloak to obtain an access token it can use to invoke on other remote services on behalf of the user. For our purposes, we implement the first use case.

As shown in Figure 13, we add a user. When creating that user, we add a credential for them in the form of a password. We then add a client representing the resource or group of resources we wish to protect. Both the user and the client (resource) should be created within the same Keycloak realm.

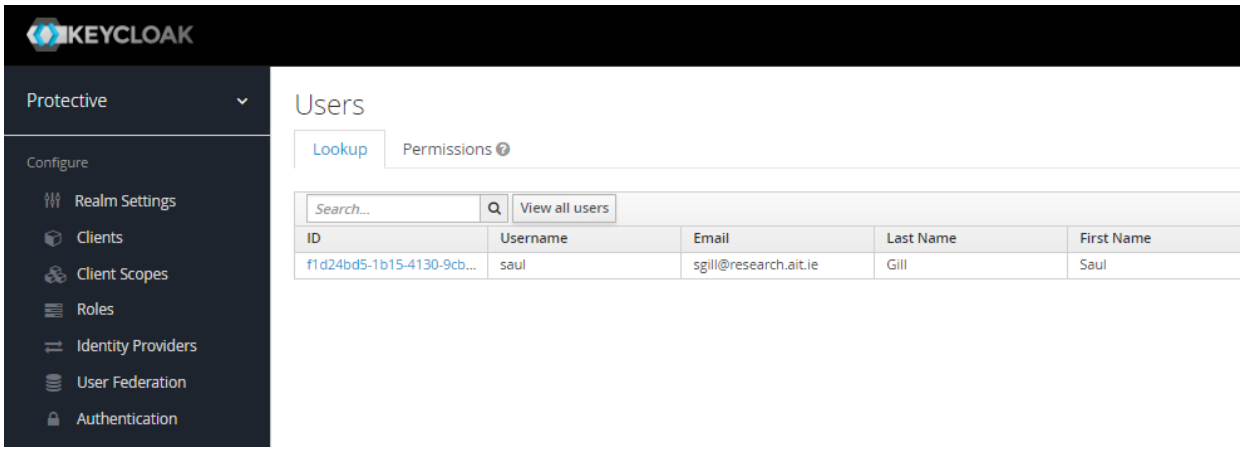

Figure 13: Keycloak Admin Users

Once the steps are carried out, the JSON configuration provided by the Keycloak admin console can be used to point the application at the Keycloak server.

Additionally, we added what is called a "guard" to the routes of the Angular UI. This guard prevents all users not authenticated by Keycloak from accessing the UI. When the user tries to access the UI, they are redirected to the Keycloak login page. Once they enter their username and password, they are permitted entry to the dashboard, as long as they have the correct authorization access to view the dashboard.

Authorization

Authorization is implemented using OAuth2 Keycloak for the resources in the PROTECTIVE node. Following on from the Authentication section above, once the user credentials are provided to the Keycloak server, the server responds with a JWT token. In our authorization implementation, we use this token to gain access to Keycloak's Entitlement API. The Entitlement API provides a 1-legged protocol for obtaining authorization data from the server, where the authorization data represents the result of the evaluation of all permissions and authorization policies associated with the resources being requested.

If the token being passed to the Entitlement API has the correct permissions/roles and the token itself is valid, the Entitlement API replies with a Requesting Party Token (RPT). The RPT contains details of the roles to which the user is entitled to access. We then use the information contained in the RPT to enforce authorization within the PROTECTIVE node UI.

This means that only those users/clients that have the correct roles assigned to them in Keycloak will be able to access certain protected parts of the UI. For example, only a user with the role, "statistics" will be able to gain access to the statistics section of the UI. Of course, these roles are fully configurable within the Keycloak admin console.

Code base

All code to allow the integration of Keycloak has been made available in the following repository:

https://gitlab.com/protective-h2020-eu/viz/ProtDash-Angular

This repository has a README that contains a Keycloak setup section. The section includes a guide to setting up Keycloak with a demo configuration. This means that demonstration realms, clients, resources, users, roles and permissions are included in the README.

All elements of the authorization and authentication are dockerized and the images have been made available in the docker registry located at:

registry.gitlab.com/protective-h2020-eu/viz/protdash-angular

4 Ethical Impacts

Computation of trust scores can be considered sensitive to a PROTECTIVE node (i.e., the NREN), and is not shared with other nodes. The security of the PROTECTIVE node though has the potential to affect the ethical impacts. **Assuming the system is secure against unauthorised users, then there are no direct ethical impacts arising from this function as none of the data is shared with other PROTECTIVE nodes.** Indirect ethical impact or potential ones in the face of successful attacks or data leakage include:

- **TI Distribution**: if data regarding the distribution is leaked, it would deliver insight into which TI is shared with, as well as access to the TI content itself. This could include personal data – should any exist prior to leaving the node (which in itself is unlikely), or insight about how the CSIRT acts on TI received. If the integrity of the system is compromised, false TI can be shared with other nodes or other actors.
- **TI Analytics**: if data about analytics is leaked, it would provide insight into how TI is analysed and used. This can include insight into mission priorities of an NREN, as well as computed trust scores. Furthermore, if the integrity of the system is compromised, false TI can be shared with other nodes or attacker actors.

- **TI Admin**: This component is responsible for authentication of users and access control based on user roles and permissions. Its aim is to make the PROTECTIVE node secure and thus minimize the dangers of data leakage using OIDC and OAuth2.0 specifications. Even if user credentials from the Keycloak server leak, it is still non-trivial to obtain access to protected resources. However, in the unlikely event of a data breach, access to the data produced by other PROTECTIVE components is possible.

We actively use the six-layered ethics framework discussed in D2.4 and D2.2 to continually review ethical impact, including data handling and security concerns that may rise in the implementation and deployment of the system (including all is implementation components). Moving forward, the project will discuss how ethical impacts play out in live environments in future deliverables. Through the pilot, we aim to identify additional impacts that we would be unable to observe in isolated development.

5 IPR Impacts

In this delivery, we have not had to face any problem or inconvenience with the IP. We have carefully analysed the licenses utilized by the modules described in this deliverable (see Table 1) and concluded that no IPR challenges exist.

Table 1: An overview of TI-Trust and TI-Admin library licenses

Library/Module	Version	License
jmespath	0.9.3	MIT License (MIT)
numpy	1.14.0	OSI Approved (BSD)
requests	2.18.4	Apache Software License (Apache 2.0)
python-dateutil	2.6.1	Apache Software License, BSD License (Dual License)
mmh3	2.5.1	CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (License :: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication)
idea-format	0.1.13	ISC License (ISCL) (ISC)
Keycloak	4.0.0	Apache Software License (Apache 2.0)
Keycloak-angular	2.0.2	MIT License (MIT)

6 References

IDEA, 2017. *IDEA.* [Online]
Available at: https://idea.cesnet.cz/en/index
[Accessed 26 2 2018].

Mentat, 2017. *Mentat.* [Online]
Available at: https://mentat.cesnet.cz/en/index
[Accessed 26 02 2018].

Warden, 2014. *WARDEN.* [Online]
Available at: https://warden.cesnet.cz/en/index
[Accessed 26 02 2018].