Horizon 2020 Programme
Instrument: Innovation Action

Proactive Risk Management through Improved Cyber Situational Awareness

PROTECTIVE
PROACTIVE RISK MANAGEMENT

Start Date of Project: 2016-09-01

Duration: 36 months

D3.5 Correlation and Prioritisation Platform Component v3

Deliverable Details	
Deliverable Number	**D3.5**
Revision Number	**E**
Author(s)	**GMV, PSNC**
Due Date	**31/10/2018**
Delivered Date	**26/10/2018**
Reviewed by	**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.	**GMV (contributor)**
2.	**PSNC (contributor)**
3.	**RoEduNET (reviewer)**

Revision History

Revision	By	Date	Changes
E	PSNC	26/10/2018	Version submitted to REA
A4	PSNC	25/10/2018	Editorial corrections
A3	PSNC	24/10/2018	Addressing comments and minor corrections
A2	GMV	24/10/2018	Internal WP review
A1	PSNC	24/10/2018	Initial draft

Abbreviation's List

API	Application Programming Interface
ASN	Autonomous System Number
CA	Context Awareness
CEP	Complex Event Processor
DAS	Data Analytics Server
DNS	Domain Name Service
DRSA	Dominance-Based Rough Set Approach
DX.Y	Deliverable No. X.Y
FQDN	Fully Qualified Domain Name
FR	Functional Requirement
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDEA	Intrusion Detection Extensible Alert
IEP	Information Exchange Policy
IP	Internet Protocol
JRE	Java Runtime Environment
JSON	Javascript Object Notation
MAP	Meta-Alert Prioritisation Module
MCDA	Multi-Criteria Decision Analysis (or Aiding)
N/A	Mom Applicable
SC	SCenario
SME	Small to Mid-size Enterprise
TI	Threat Intelligence
UUID	Universally Unique Identifier
VM	Virtual Machine
WP	Work Package

Executive Summary

This deliverable D3.5: Correlation and Prioritisation Platform Component describes the version updates brought in for the two software components of the PROTECTIVE system, namely correlation and prioritisation modules [D3.5]. The prior versions have been described in deliverable D3.4. The first chapter is a short description of the feature(s) being delivered. Chapter two highlights the scenarios, requirements and parts of the architecture fulfilled by the software components. Chapter three contains the GitLab links for the source code and instructions for installation. It also includes a selection of screenshots of the executed software. Chapter four highlights any ethical related impacts.

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 features being delivered in addition to those described in D3.4 are described below on a per module basis. The modules have been implemented within separate project tasks.

Correlation Module which is composed of Correlation Pipeline handling module, WSO2 correlation engine, Events (Alerts) extractor (c.f. D3.4), and enrichment modules, has been extended by following functionalities:

- Incorporation of enrichment data provided by Trust Module into Correlation Pipeline – implementation of custom aggregation functions for WSO2.
- Incorporation of enrichment data provided by Context Awareness Module into Correlation Pipeline.
- Addition of new processing path to allow sharing of non- contextualised Meta-alerts through TI sharing mechanisms.
- Addition of new correlation rules to WSO2 CEP engine and adjustment of existing ones.
- Optimisation of custom modules for WSO2 to mitigate memory and performance issues.

Meta-Alert Prioritisation Module (MAP), which is composed of Criteria Mapper Module and Ranking Generation Module, has been extended by functionalities described below.

In the case of Criteria Mapper Module (criteria-mapper), which is a wrapper module placed between GUI and Ranking Generation Module and prepares input for classification algorithms, implementation has been extended by:

- Addition of API calls that allows to retrieve, in a single API call, classification results with the whole meta-alerts and criteria vectors constructed on the basis of user defined template.
- Ability to provide output not only in JSON but also as HTML document with Java Script. The output is produced on the basis of the user-defined template that can be updated ‘on-the-fly’ without the need for service restart.
- Addition of support for GET requests alongside with handling of POST requests, which aims in easier integration with frameworks for graphical user interfaces.
- Incorporation of redesigned web-forms that allows for testing of MAP API calls and can serve as part of a GUI.

Ranking Generation Module (ranking-generator) provides implementations of classification algorithms for prioritisation along with APIs that allows integration with other components of PROTECTIVE platform, in particular with Criteria Mapper Module that produces inputs for prioritisation. The module in its current version implements API that allows to fetch rankings in the short form (priority class and meta-alert id) or the long form (priority and vector of criteria), definitions of the rules used for prioritisation in textual form or in RuleML.

Additionally, Angular service has been implemented in order to show a possible way of integration of MAP with dashboards.

2 Scenarios, Requirements & Architecture

The scenarios for PROTECTIVE are shown in Figure 1The scenario SC4 highlighted in yellow has been partially completed. Note: the functionalities that have been delivered but not yet fully tested or integrated (which is under the scope of work package WP6) are considered as *partially* fulfilled or implemented within this document.

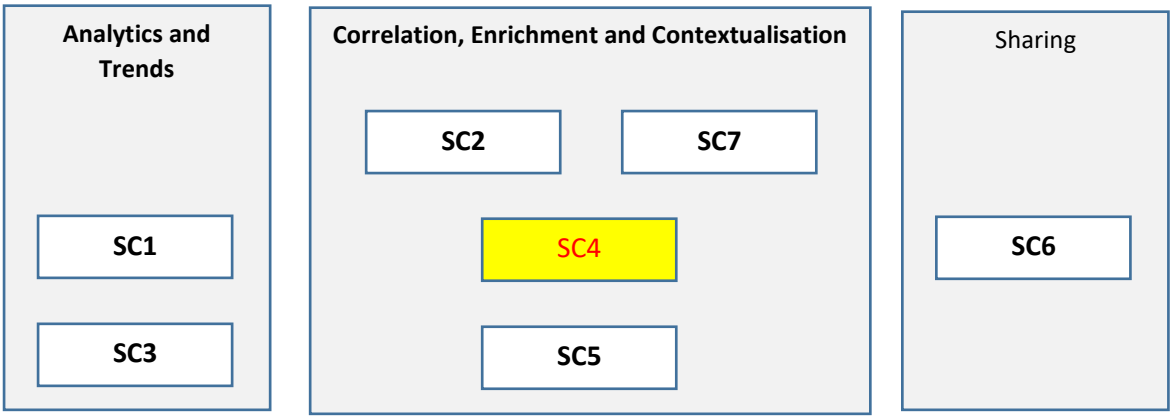


Figure 1: Scenario Status

Scenario 4 - Alert Correlation and Prioritisation

Referring to the description of SC4 (see D2.1 for details), the features implemented for this deliverable are:

- Read and process data from IDEA format alerts.
- Process data provided by Trust module.
- Contextualise meta-alerts using data provided by Context Awareness module.

Prioritisation module is able to use the following data from enriched meta-alerts in the ranking process:

- vulnerabilities,
- mission related data,
- risk related data,
- data quality related information,
- reputation data.

The subsystem can use all available data to correlate alerts based on both:

- Pre-defined rules
- Rules provided by the advanced user.

The detection of a particular pattern can lead to:

- Creation of new meta-alert,
- Addition of new data to existing meta-alert,
- Change of existing meta-alert rank/priority,
- Change in initial classification.

One detection mode has been completed - on-line (real-time) processing (short time window, possibly in-memory processing).

The time windows of events used for patterns discovery should be configurable. The user should be able to:

- Select alerts' attributes to be used in the correlation process,
- Select predefined rules or set of rules and related actions,
- Define own rules using Siddhi and define actions that should be conducted on meta-alerts in the case of conditions in a rule being fulfilled,
- Create groups of rules.

The actions executed based on the defined rules that allow:

- The creation of new meta-alerts,
- Discarding data,
- The addition of external data to meta-alert,
- The addition of alerts to existing meta-alerts,
- Reassignments of categories.

2.1 Requirements

The following requirements have been completed (marked Done) or partially fulfilled by this delivery. The functionalities that have been delivered but not yet fully tested or integrated are considered as partially fulfilled within this document.

Alert Prioritisation

ID	PR-01	Done
Type	FR	
Slogan	PROTECTIVE must support the relative prioritisation of threat data in terms of their value/importance to the organisation's mission	
Rationale	This will define the requirements and methodology to specify a meta-model and related modelling language in order to enable the definition of domain specific criticality taxonomies involving object types such as assets, organisational structures and roles etc.	
Related Scenarios	SC-4, SC-7	
Related Requirements	CA-01, CA-02, CA-03, CA-04, CA-05, CA-06, CA-07	
ID	PR-02	Done
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 impacts 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	
ID	PR-03	Done

Type	FR	
Slogan	PROTECTIVE MUST be able to condense related information to provide a holistic view of the current threat situation, thus raising the situational awareness	
Rationale	Raw alerts provide scattered information and require effort from the operator to understand, correlating related alerts will provide a high-level understanding of the current threat situation	
Related Scenarios	SC-4	
Related Requirements	CR-01, CR-02, CR-03, CR-04	
ID	PR-04	Partially
Type	FR	
Slogan	PROTECTIVE MUST be able to prioritise threat data in terms of the preferences/needs of the specific operator	
Rationale	To support the responsibilities of individual operators, the operators must be able to define which attributes/categories are most import for them, this needs to be reflected in the prioritisation and the visualisation of meta-alerts.	
Related Scenarios	SC-4	
Related Requirements	UI-01	

Table 1: Alert Prioritisation requirements fulfilled

Alert Correlation

ID	CR-01	Done
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-02	Done
Type	FR	
Slogan	PROTECTIVE MUST support the correlation of meta-alerts.	
Rationale	Meta-Alert correlation increases the information density over single events and provides a higher-level view on the current situation.	
Related Scenarios	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	
ID	CR-04	Done
Type	FR	
Slogan	PROTECTIVE must enable meta-alerts to be enriched with context related information.	
Rationale	Context related information gives a deeper understanding to the indicator and enables the analyst to clearly understand the threat the indicator is meant to detect e.g. to include the malware family and variant with a malware hash.	
Related Scenarios	SC-2, SC-4	
Related Requirements	N/A	

Table 2: Alert Correlation requirements fulfilled

2.2 Architecture

The architecture of the PROTECTIVE node in Figure 2 shows the components included in this delivery. The components highlighted in green have been completed for this delivery, those in yellow have been partially completed. The components provided within this delivery are fully functional, however not fully integrated with other recently developed components of PROTECTIVE node, thus they are marked as partially completed. They will be considered as completed after passing tests during integration in test-bed within the confines of work package WP6.

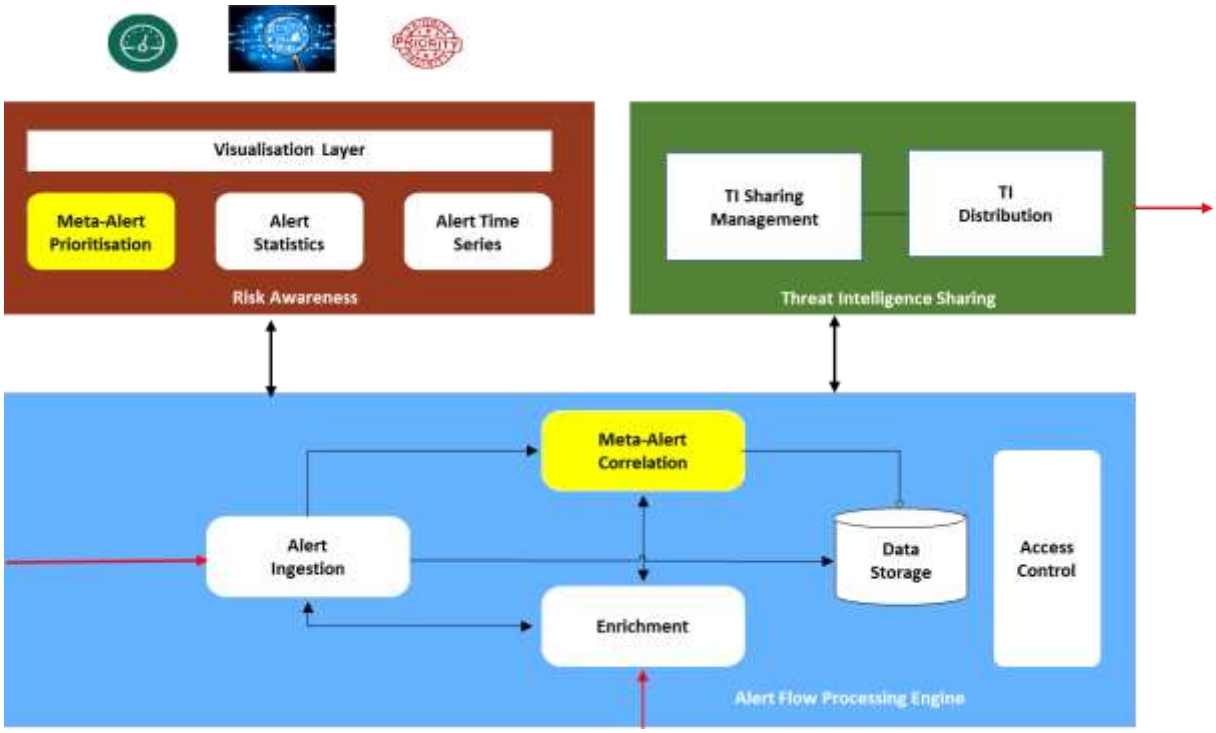


Figure 2: PROTECTIVE NODE Status

The architecture presented in Figure 3 and Figure 4 is an in-depth view of the modules being delivered, which correspond with the Meta-alert Correlation and Prioritisation modules. A flow of alerts through the modules is as follows. Mentat Enricher Extractor forks alerts' flow and sends individual alerts through Meta-alert Pipeline where alerts are correlated into meta-alerts and further contextualised

by Context Awareness Enrichment module, using data provided by Context Awareness modules implemented within work package WP4. Correlation Engine is also capable of sharing meta-alerts before contextualisation step with Threat Intelligence (TI) distribution components. Finally, meta-alert are stored in a database from where Meta-Alert Prioritization Module can fetch them (Figure 4), converted into vectors of criteria values by criteria-mapper and further prioritised by ranking-generator. Results of prioritisation are sent to GUI for visualisation. For further information about implementations of each component refer to section 3.

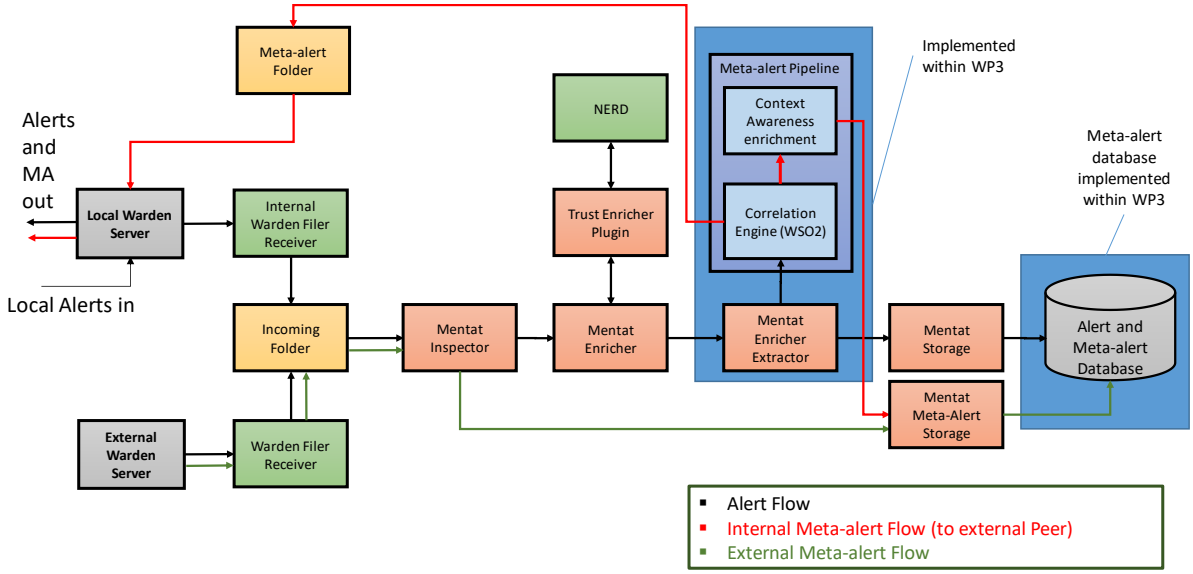


Figure 3: Alert and Meta-alert flow diagram: components delivered within this deliverable are highlighted using blue background

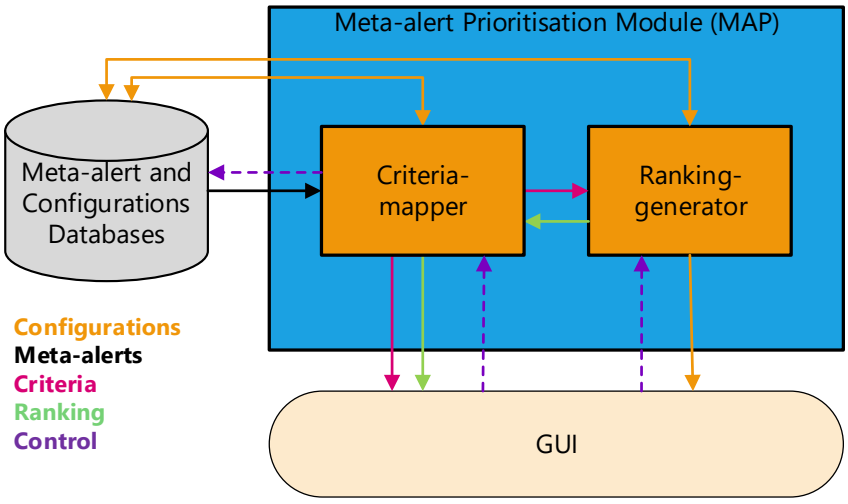


Figure 4: Meta-alert Prioritisation Module

3 Implementation

3.1 Prerequisites

Prerequisites do not differ significantly from those described in deliverable D3.4 for the previous release of the software components. The technological stack has not been altered and most of components can be run as standalone modules, however data from other PROTECTIVE components are needed to get meaningful results and discover, and use, a full potential of the correlation and prioritisation engines.

For the Correlation engine the machine (or VM) needs to have git, Docker, docker-compose installed and an internet connection in order to download all the Docker images and dependencies. There are no dependencies with PROTECTIVE nodes; it can be executed standalone. However, to contextualise Meta-alerts, it needs access to APIs provided by Context Awareness module (CA MIM) as described in deliverable D4.4. In order to compute quality for Meta-alert access to data provided by Trust Component (see deliverable D5.3 for details around Trust Component) is needed.

For Ranking Generation Module the machine (or VM) needs to have git, Java JRE version 1.8 or higher installed and an internet connection in order to download all the Gradle dependencies. There are no direct dependencies with Protective nodes. Although without output from other modules, incorporated into meta-alerts during correlation phase, the range of criteria that can be used to produce ranking becomes very limited.
The module implements classification and ranking generation algorithms based on Dominance-based Rough Set Approach (DRSA). An overview of DRSA has been presented in deliverable D3.1.

For the Criteria Mapper Module the machine (or VM) needs to have git, Node.js with STJS and MongoDB modules or, for a dockerised version, Docker (instead of Node.js) installed and an internet connection to download all the Docker images and dependencies. There are no dependencies with PROTECTIVE nodes; it can be executed standalone, only access to MongoDB (standalone or in a Docker) containing meta-alerts is needed. Alert and Meta-alert enrichment processes highly influence spectrum of possible criteria available for prioritisation as it was noted above.

3.2 Execution

The mentat-enriched-extractor module is placed in gitlab, https://gitlab.com/protective-h2020-eu/meta-alert-correlation/mentat-enriched-extractor

The correlation-pipeline feature is placed in gitlab, https://gitlab.com/protective-h2020-eu/meta-alert-correlation/correlation-pipeline

The correlation-engine module WSO2 CEP is placed in gitlab, https://gitlab.com/protective-h2020-eu/meta-alert-correlation/WSO2CEP

All of them have their Docker images uploaded in their Registry section.

To deploy the whole solution, some steps have to be followed as explained in:
https://gitlab.com/protective-h2020-eu/cp/Mentat/Mentat-config/tree/with-experimental-module

The criteria-mapper module is placed in gitlab, https://gitlab.com/protective-h2020-eu/meta-alert-prioritisation/criteria-mapper

The ranking-generator module is placed in gitlab, https://gitlab.com/protective-h2020-eu/meta-alert-prioritisation/ranking-generator

3.3 Mentat Enriched Extractor

This module is written in Python and can be found in the repository https://gitlab.com/protective-h2020-eu/meta-alert-correlation/mentat-enriched-extractor. It implements the extraction of IDEA events.

The module just gets the events from the Mentat pipeline, sends them to the proper handlers managed by PipelineHandler.js and then forward the events to the next Mentat module.

The IDEA events are sent to PipelineHandler.js using the PUB/SUB pattern via the ZeroMQ library: http://zguide.zeromq.org/php:chapter5#toc1.

```
root@04924ab1a4e7:/usr/src/app# mentat-ideagen.py --count 100
2018-02-13 15:33:49,164 INFO: Executing script command 'generate_randon'
2018-02-13 15:33:49,678 INFO: [1] Message was generated and stored to file '/var/mentat/spool/mentat-inspector.py/incoming/testmsg-uo3cchafbhxqxqre9ha4.idea'
2018-02-13 15:33:49,682 INFO: [2] Message was generated and stored to file '/var/mentat/spool/mentat-inspector.py/incoming/testmsg-6n3vs9ta352upef8wrf8.idea'
2018-02-13 15:33:49,685 INFO: [3] Message was generated and stored to file '/var/mentat/spool/mentat-inspector.py/incoming/testmsg-33t45ql0tpnh7upqzom8.idea'
2018-02-13 15:33:49,687 INFO: [4] Message was generated and stored to file '/var/mentat/spool/mentat-inspector.py/incoming/testmsg-a6gkasgqdsmllqjq5zqq.idea'
```

Figure 5: IDEA alerts are generated and sent to NODE.js PipelineHandler by the mentat-enriched-extractor

3.4 PipelineHandler

The pipeline handler manages the lifecycle of the events forked from the Mentat pipeline. This lifecycle consists of the following steps:

- MentatClient.js - Parse the IDEA events received from mentat-enriched-extractor into a suitable format for the WSO2 CEP.


```
info: [Mentat Client] Received event: {
   "Format": "IDEA0",
   "ID": "testmsg-gt86lffmnntj4h1gjcvx",
   "DetectTime": "2018-02-13T16:13:22Z",
   "Category": ["Abusive.Spam","Test"],
   "ConnCount": 633,
   "Description": "Ping scan",
   "Source": [
      {
         "IP4": ["10.0.0.5"],
         "Proto": ["icmp"]
      }
   ],
   "Target": [
      {
         "Proto": ["icmp"],
         "IP4": ["10.10.0.5"],
         "Anonymised": true
      }
   ],
   "Node" : [
      {
         "SW" : ["warden_filer"],
         "Name" : "cz.cesnet.mentat"
      },
      {
         "SW" : ["Beekeeper"],
         "Name" : "cz.cesnet.rimmer"
      }
   ]
}
```


Figure 6: PipelineHandler receives IDEA Alerts

- IdeaEmitter.js - Send the formatted events to WSO2 CEP. The explanation of this format can be found at *3.5.1 Execution Flow*

Figure 7: PipelineHandler sends events to WSO2 CEP

- Wso2dasServer.js - Receive the meta-alerts generated at WSO2 CEP.

Figure 8: PipelineHandler receives Meta-alerts from WSO2 CEP

- Parse the meta-alerts into a suitable format.
- Meta-alerts follow two different flows:
  - The meta-alerts are sent to local warden server to be shared.
  - The meta-alerts are enriched by Context Awareness module and sent to the Mentat Meta-alerts storage module to be inserted into the database.

```
> use mentat
switched to db mentat
> show collections
alerts
meta-alert
meta_alerts
> db.meta_alerts.findOne({ID:testmsg-cd4iyrc8npqekywhipet})
2018-02-13T15:40:23.931+0000 E QUERY    [thread1] ReferenceErr
@(shell):1:25
> db.meta_alerts.findOne({ID:'testmsg-cd4iyrc8npqekywhipet'})
{
        "_id" : ObjectId("5a8305b4d393f800190e2d0e"),
        "ID" : "testmsg-cd4iyrc8npqekywhipet",
        "DetectTime" : "2018-02-13T15:34:47Z",
        "MACategory" : "Abusive.Spam",
        "AggrID" : "testmsg-cd4iyrc8npqekywhipet",
        "TopAssetCriticality" : " ",
        "TopMissionImportance" : " ",
        "Source" : {
                "Asset" : " ",
                "MaxAssetCriticality" : " ",
                "Mission" : " ",
                "MaxMissionImportance" : " ",
                "IP4" : "195.113.144.230",
                "Proto" : "icmp",
                "Hostname" : "undefined"
        },
        "Target" : {
                "Asset" : " ",
                "MaxAssetCriticality" : " ",
                "Mission" : " ",
                "MaxMissionImportance" : " ",
                "IP4" : "10.10.0.5",
                "Proto" : "icmp",
                "Hostname" : "undefined"
        }
}
>
```

Figure 9: MongoDB Correct Meta-alert Insertion Verification

3.5 WSO2 DAS

WSO2 DAS is the correlation technology engine used. It receives the input events from the PipelineHandler. The complete execution flow inside WSO2 DAS is as follows (Figure 10) and it is extended in comparison to the previous version of this component implementation.

Figure 10: WSo2 DAS Execution Flow

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.

WSO2 follows the IDEA message definition [2] with extensions added to incorporate data produced by Trust Module (implemented in work package WP5). The IDEA JSON format has some array type elements with a non-predefined length such as Source, Target or Node. Due to the fact that WSO2 does not support receiving variable length elements, a workaround has been made.

The correlation-pipeline splits each IDEA alert in (at least) four events, Target, Category, Quality, and Source, with as many Target (or Source) events as elements in the corresponding JSON array. These events are the input of the WSO2 module through an HTTP connection (TargetHTTP, CategoryHTTP, SourceHTTP, QualityHTTP receivers), which will then aggregate them (Generate_Event execution plan) by their correlation_id field in order to generate the original IDEA alert again. Once the distinct execution plans (or rules) are checked, the new created Meta-alerts are sent back to the PipeHandler module (through MetaAlert_HTTP).

For further information about the predefined rules (or execution plans) available in the current release and the structure of WSO2 DAS look at Annex A.

3.6 Criteria-mapper

This module can be found in the repository https://gitlab.com/protective-h2020-eu/meta-alert-prioritisation/criteria-mapper and implements mappings of values present in meta-alerts into criteria that can be used altogether in MCDA based ranking process. The criteria-mapper serves as a wrapper between GUI, ranking-generator and meta-alerts database (MongoDB). The input to the module is a template file that defines mappings (including time-dependent functions) and the query string that is passed to the database to filter relevant meta-alerts. The meta-alerts fetched from database are transformed into vectors of criteria values, processed by Ranking Generator (ranking-generator) module and send back to user in form of JSON or HTML. The default output is the list of JSON objects – one object per meta-alert, with values of criteria and priority class as returned by ranking-generator. Please refer to https://gitlab.com/protective-h2020-eu/meta-alert-prioritisation/criteria-mapper/blob/master/README.md for details.

The module currently support the following API calls:

Endpoint	Input	Output

/rank	**Parameters (POST or GET methods):**
	query – MongoDB query string
		JSON array of results provided by ranking-generator in short form (meta-alert ID, priority):
		`[{`
		`"ID": "b70eac70-b4ac-11e7-9460-002564d9514f",`
		`"Priority": 3`
		`},`
		…
		`]`

/rank-ext	**Parameters (POST or GET methods):**
	query – MongoDB query string
		JSON array of results provided by ranking-generator in long form (meta-alert ID, priority, values of criteria):
		`[{`
		`"ID":"437f5917-56d7-4653-b904-a61fb924eca7",`
		`"SourceAssetCriticality":"critical",`
		`"TargetAssetCriticality":"NA",`

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.

Endpoint	Input	Output
		"TimeToDueDate":"54598.526", "TimeFromDetectTime":"31801.474", "SeverityForAttackCategory":"med", "MAQuality":"0.5", "AttackSourceReputationScore":"0.5", "MaxCVE":"0.0", "Priority":"1" } , …]
/rank-meta-alert	**Parameters (POST or GET methods):** query – MongoDB query string MinIndex – the index of the first object from results' set to be returned (it allows to fetch results in slices) MaxIndex – the index of the last object from results' set to be retuned **POST only parameters (in addition to above mentioned):** HtmlOut – specify that output should be produced in HTML ReorderOut – sort object properties alphabetically by keys names	Two possible outputs: 1. JSON array with combined classification results, criteria and meta-alerts properties. 2. HTML table generated from JSON data.
/criteria	**Parameters (POST or GET methods):** query – MongoDB query string	The output is almost identical to the one provided for /rank-ext but the Priority is not given. The criteria vectors are not passed through ranking-generator.
/form	None	Form to test API
/form-gui	None	Simplified form to test functionality of MAP

Table 3: API calls supported by criteria-mapper

The API can be tested using the form provided under /form-gui as shown below in Figure 11.

Figure 11: Web-form for testing criteria-mapper API

An exemplary execution case of the module:

1. Service start:

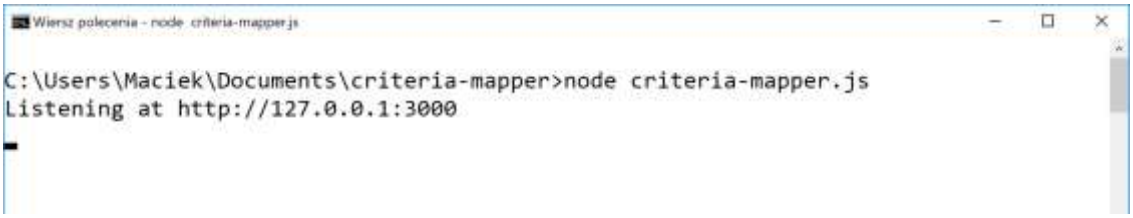


Figure 12: Initialisation of criteria-mapper.js as Node.js application

2. Inputs:
   Inputs are provided using web-forms accessible through /form and /form-gui URLs or through direct API calls. The meta-alerts to be prioritised are fetched directly from database.
   The inputs are following:
   - Query string – allowing selection of meta-alerts

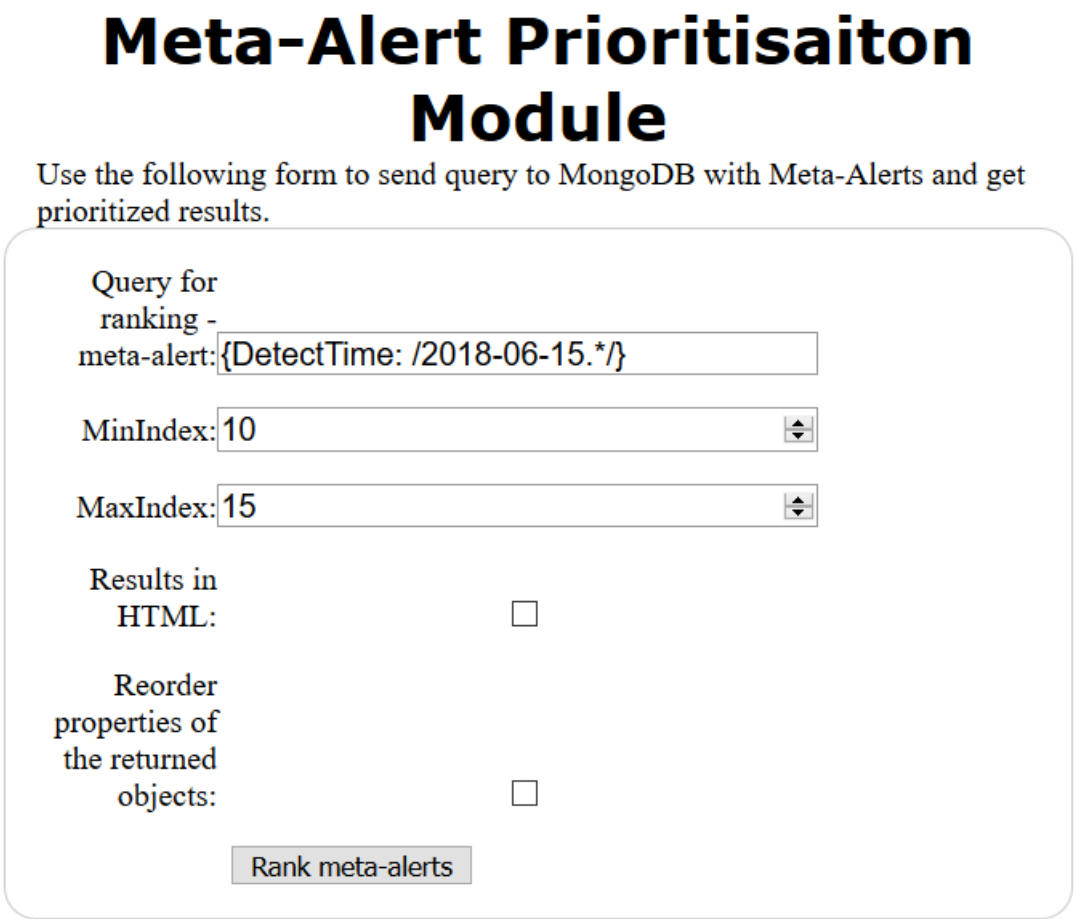


Figure 13: Example query string used to filter meta-alerts from MongoDB

   If the query string is not provided then default ({}) is used.

   - Additional parameters as described in Table 3 (see also Figure 13)
   - Meta-alerts in MongoDB (shown in MongoDB Compass)

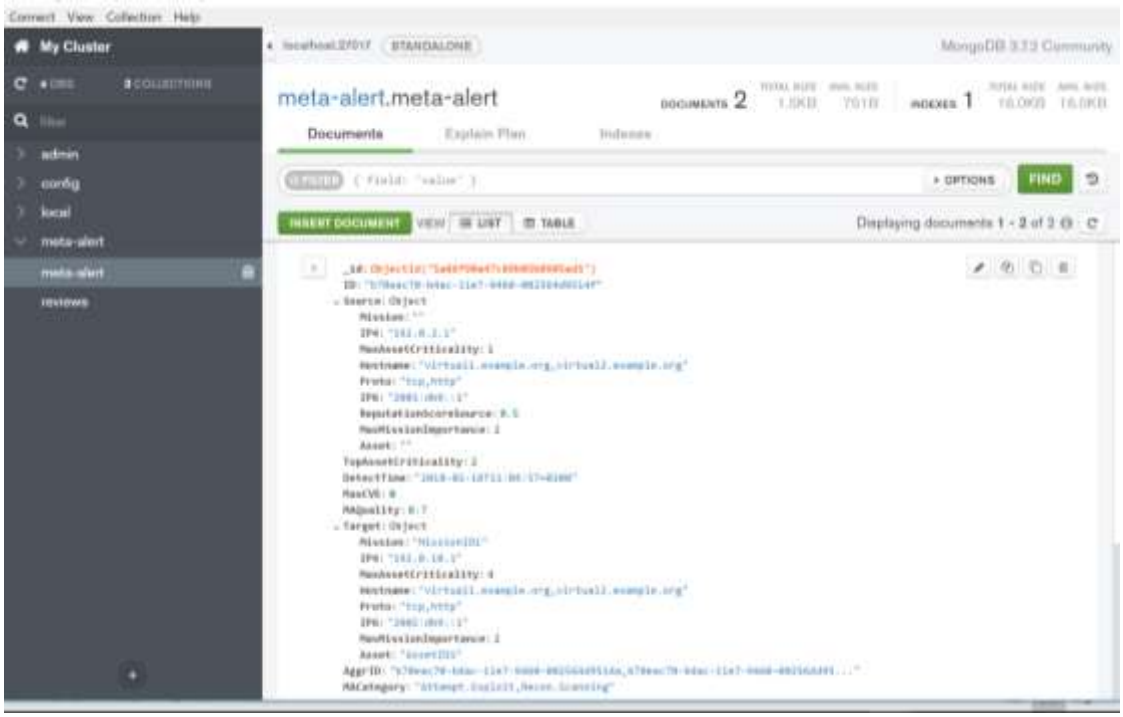


Figure 14: Example of meta-alert in MongoDB

- Template file – defines transformation of meta-alert into values of criteria, it allows also to specify temporal (dependent from current time) criteria

```
{
   "{{#each items}}": {
      "TargetAssetCriticality": "{{Target.MaxAssetCriticality}}",
      "SourceAssetCriticality": "{{Source.MaxAssetCriticality}}",
      "TimeToDueDate": "{{{{Date.parse(DetectTime)+_MaxHandlingTime_}-Date.now()}/1000}}",
      "TimeFromDetectTime": "{{{Date.now()-Date.parse(DetectTime)}/1000}}",
      "MAQuality": "{{MAQuality}}",
      "AttackSourceReputationScore": "{{Source.ReputationScoreSource}}",
      "SeverityForAttackCategory": "{{%
        (function(category) {%
          if (RegExp('Attempt.Exploit*').test(category)) {return 4;} %
          if (RegExp('Recon.Scanning*').test(category)) {return 1;} %
         return 2;})(MACategory)}}",
      "MaxCVE": "{{MaxCVE}}",
      "ID": "{{ID}}"
   }
}
```

Figure 15: Template file for the criteria-mapper

- Parameters file

```
{
"_MaxHandlingTime_": 86400000
}
```

Figure 16: File with parameters for the mapping template

3. Results

   Output in HTML form is shown in Figure 17.

Figure 17: Example prioritisation results in HTML format

Priority	SourceAssetCriticality	TargetAssetCriticality	SeverityForAttackCategory	MAQuality	TimeToDueDate	TimeFromDetectTime	DetectTime	MACategory	Source	Target
"1"	"critical"	"NA"	"med"	"0.5"	"57132.71"	"29267.29"	"2018-06-15T10:36:36Z"	"Attempt Login"	IP4:109.248.9.105,5.188.86.195,5.188.87.35 Proto: Port: Hostname:	IP4:81.180.251.50 Proto:tcp,ssh Port:22 Hostname:
"1"	"critical"	"NA"	"med"	"0.5"	"57437.706"	"28962.294"	"2018-06-15T10:41:41Z"	"Abusive Spam"	IP4:66.118.187.34 Proto:tcp,smtp Port:undefined Hostname:undefined	IP4:undefined Proto:undefined Port:undefined Hostname:undefined
"2"	"high"	"NA"	"med"	"0.5"	"57158.709"	"29241.292"	"2018-06-15T10:37:02Z"	"Attempt Login"	IP4:109.248.9.103,5.188.86.211,109.248.9.105 Proto: Port: Hostname:	IP4:81.180.251.50 Proto:tcp,ssh Port:22 Hostname:
"3"	"med"	"NA"	"med"	"0.5"	"57158.707"	"29241.293"	"2018-06-15T10:37:02Z"	"Attempt Login"	IP4:109.248.9.103,5.188.86.167,81.180.73.214 Proto: Port: Hostname:	IP4:81.180.251.50 Proto:tcp,ssh Port:22 Hostname:
"4"	"low"	"NA"	"med"	"0.5"	"57166.707"	"29233.293"	"2018-06-15T10:37:10Z"	"Abusive Spam"	IP4:80.211.28.148 Proto:tcp,smtp Port:undefined Hostname:undefined	IP4:undefined Proto:undefined Port:undefined Hostname:undefined

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.

- Output from the console showing the status of processing

Figure 18: The criteria-mapper module receives POST with the query string

- Response

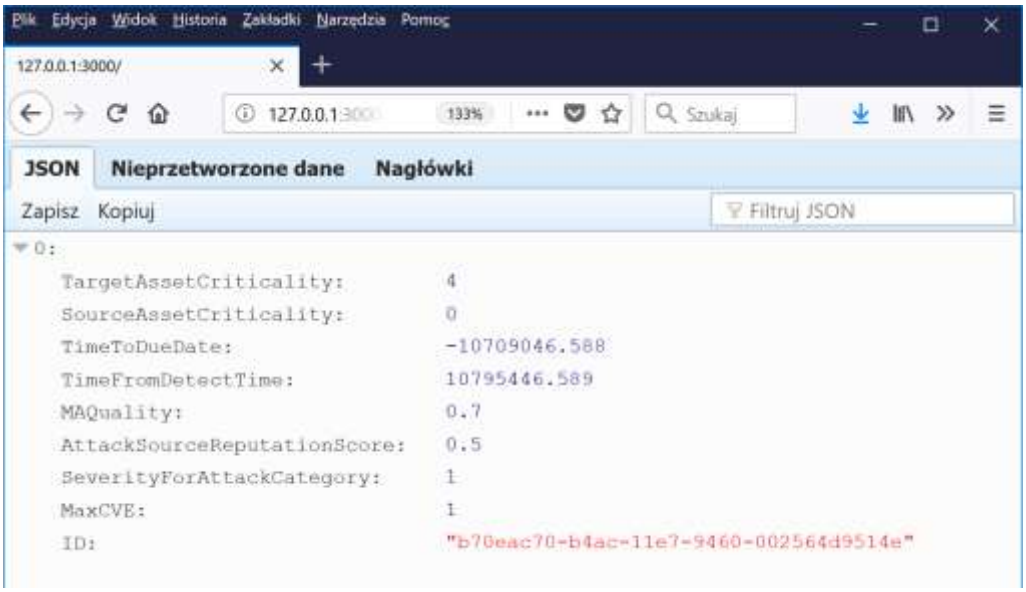


Figure 19: An example of the response in JSON

- Response in the raw form:

```
[{"TargetAssetCriticality":4,"SourceAssetCriticality":0,"TimeTo
DueDate":-
10709046.588,"TimeFromDetectTime":10795446.589,"MAQuality":0.7,
"AttackSourceReputationScore":0.5,"SeverityForAttackCategory":1
,"MaxCVE":1,"ID":"b70eac70-b4ac-11e7-9460-002564d9514e"}]
```

4. Exemplary invocation of the service using command line only:

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.

Figure 20: An example invocation of the service from the command line

3.7 Ranking Generation Module

This module can be found in the repository *https://gitlab.com/protective-h2020-eu/meta-alert-prioritisation/ranking-generator* and implements generation of meta-alerts' ranking [1].

The classification is produced using Dominance-based Rough Set approach as described in deliverable D3.2.

Ranking-generator supports following API calls (Table 4).

Endpoint	Input	Output
/rank	**Parameters (POST):** JSON array with criteria vectors in request body.	JSON array of results in short form (meta-alert ID, priority): [{ "ID": "b70eac70-b4ac-11e7-9460-002564d9514f", "Priority": 3 }, …]

/rank-ext	**Parameters (POST):** JSON array with criteria vectors in request body.	JSON array of results in long form (meta-alert ID, priority, values of criteria):
		[{
		"ID":"437f5917-56d7-4653-b904-a61fb924eca7",
		"SourceAssetCriticality":"critical",
		"TargetAssetCriticality":"NA",
		"TimeToDueDate":"54598.526",
		"TimeFromDetectTime":"31801.474",
		"SeverityForAttackCategory":"med",
		"MAQuality":"0.5",
		"AttackSourceReputationScore":"0.5",
		"MaxCVE":"0.0",
		"Priority":"1"
		}
		,
		…
		]

/meta-data	**Parameters (GET or POST):** None or Rule-set id	JSON array with definitions of criteria
/rules-text	**Parameters (GET or POST methods):** None or Rule-set id	Prioritisation rules in textual form.
/rules-ruleml	**Parameters (GET or POST methods):** None or Rule-set id	Prioritisation rules in RuleML

Table 4: API calls supported by Ranking-Generator module

The module is configured by criteria definitions (Figure 21), and rule prioritisation model in RuleML (Figure 22, for textual representation see Figure 23) constructed on learning data.

```
[    {
       "name": "SourceAssetCriticality",
       "active": true,
       "type": "condition",
       "valueType": "enumeration",
       "domain": ["NA", "low", "med", "high", "critical"],
       "preferenceType": "gain"
    }, {
       "name": "TargetAssetCriticality",
       "active": true,
       "type": "condition",
       "valueType": "enumeration",
       "domain": ["NA", "low", "med", "high", "critical"],
       "preferenceType": "gain"
    }, {
       "name": "TimeToDueDate",
       "active": true,
       "type": "condition",
       "valueType": "real",
       "preferenceType": "cost"
    }, {
       "name": "TimeFromDetectTime",
       "active": true,
       "type": "condition",
       "valueType": "real",
       "preferenceType": "cost"
    }, {
       "name": "SeverityForAttackCategory",
       "active": true,
       "type": "condition",
       "valueType": "enumeration",
       "domain": ["NA", "low", "med", "high", "critical"],
       "preferenceType": "gain"
    }, {
       "name": "MAQuality",
       "active": true,
       "type": "condition",
       "valueType": "real",
       "preferenceType": "gain"
    }, {
       "name": "AttackSourceReputationScore",
       "active": true,
       "type": "condition",
       "valueType": "real",
       "preferenceType": "gain"
    }, {
       "name": "MaxCVE",
       "active": true,
       "type": "condition",
       "valueType": "real",
       "preferenceType": "gain"
    }]
```


Figure 21: Criteria definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model
href="http://deliberation.ruleml.org/1.01/relaxng/datalogplus_min_relaxed.rnc"?>
<RuleML xmlns="http://ruleml.org/spec">
  <act index="1">
    <assert>
      <implies>
        <if>
          <atom>
            <op>
              <rel>ge</rel>
            </op>
            <ind>critical</ind>
            <var>SourceAssetCriticality</var>
          </atom>
        </if>
        <then>
          <atom>
            <op>
              <rel>le</rel>
            </op>
            <ind>1</ind>
            <var>Priority</var>
          </atom>
        </then>
        <evaluations>
          <evaluation measure="Strength" value="0.19689119170984457" />
          <evaluation measure="Confidence" value="1.0" />
        </evaluations>
      </implies>
    </assert>
    <assert>
      <implies>
        <if>
          <and>
            <atom>
              <op>
                <rel>le</rel>
              </op>
              <ind>med</ind>
              <var>SourceAssetCriticality</var>
            </atom>
            <atom>
              <op>
                <rel>le</rel>
              </op>
              <ind>high</ind>
              <var>TargetAssetCriticality</var>
            </atom>
          </and>
        </if>
        <then>
          <atom>
            <op>
              <rel>ge</rel>
            </op>
            <ind>3</ind>
            <var>Priority</var>
          </atom>
```

```
        </then>
        <evaluations>
          <evaluation measure="Strength" value="0.4905008635578584" />
          <evaluation measure="Confidence" value="1.0" />
        </evaluations>
      </implies>
    </assert>
  </act>
</RuleML>
```

Figure 22: Exemplary rules from rule prioritization model in RuleML

```
1: (SourceAssetCriticality >= critical) =>[c] (Priority <= 1)
2: (TargetAssetCriticality >= critical) & (TimeToDueDate <= 3.17) =>[c]
(Priority <= 1)
3: (SourceAssetCriticality >= high) =>[c] (Priority <= 2)
4: (TargetAssetCriticality >= critical) & (TimeToDueDate <= 15.48) &
(AttackSourceReputationScore >= 0.93) =>[c] (Priority <= 2)
5: (SourceAssetCriticality >= med) & (TimeFromDetectTime <= 15.7) &
(SeverityForAttackCategory >= med) & (MAQuality >= 0.35) =>[c] (Priority <= 2)
6: (SourceAssetCriticality >= med) & (TargetAssetCriticality >= critical) &
(SeverityForAttackCategory >= high) =>[c] (Priority <= 2)
7: (TargetAssetCriticality >= critical) & (TimeToDueDate <= 18.59) &
(AttackSourceReputationScore >= 0.75) & (MaxCVE >= 8.2) =>[c] (Priority <= 2)
8: (SourceAssetCriticality >= med) =>[c] (Priority <= 3)
9: (SourceAssetCriticality >= low) & (TargetAssetCriticality >= critical) &
(SeverityForAttackCategory >= low) =>[c] (Priority <= 3)
10: (SourceAssetCriticality >= low) & (TargetAssetCriticality >= critical) &
(TimeFromDetectTime <= 17.41) =>[c] (Priority <= 3)
11: (SourceAssetCriticality >= low) & (TargetAssetCriticality >= critical) &
(TimeToDueDate <= 25.45) =>[c] (Priority <= 3)
12: (SourceAssetCriticality >= low) =>[c] (Priority <= 4)
13: (TargetAssetCriticality >= critical) & (SeverityForAttackCategory >=
critical) =>[c] (Priority <= 4)
14: (SourceAssetCriticality <= NA) & (SeverityForAttackCategory <= high) =>[c]
(Priority >= 5)
15: (SourceAssetCriticality <= NA) & (TargetAssetCriticality <= high) =>[c]
(Priority >= 5)
```

Figure 23: Exemplary rules in textual form

Once Ranking Generation Module receives meta-alerts description produced by Criteria Mapper Module, it constructs and returns a ranking of these meta-alerts as shown in Figure 24 and Figure 25.


```
[
  {
  "TargetAssetCriticality": "critical",
  "SourceAssetCriticality": "Low",
  "TimeToDueDate": -100896.193,
  "TimeFromDetectTime": 187296.193,
  "MAQuality": 0.7,
  "AttackSourceReputationScore": 0.5,
  "SeverityForAttackCategory": "critical",
  "MaxCVE": 0,
  "ID": "b70eac70-b4ac-11e7-9460-002564d9514f"
  },
  {
  "TargetAssetCriticality": "critical",
  "SourceAssetCriticality": "NA",
  "TimeToDueDate": -10641696.193,
  "TimeFromDetectTime": 10728096.193,
  "MAQuality": 0.7,
  "AttackSourceReputationScore": 0.5,
```

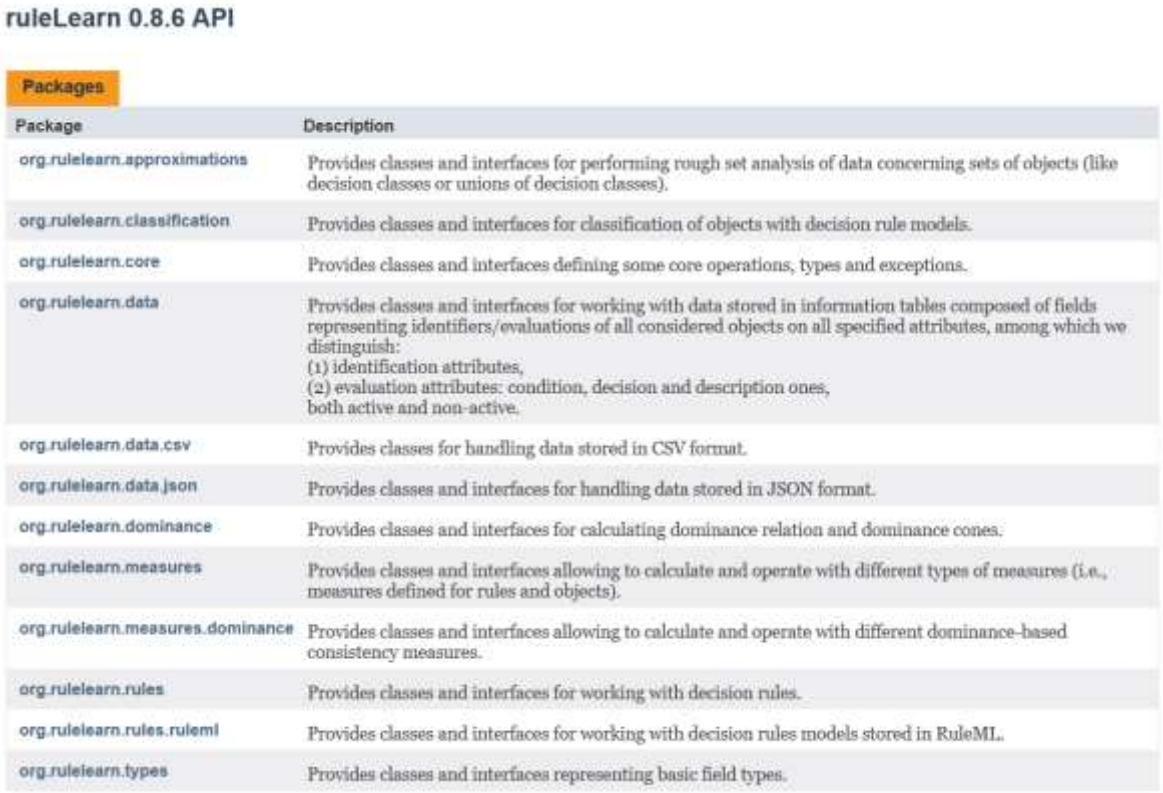
```
  "SeverityForAttackCategory": "Low",
  "MaxCVE": 1,
  "ID": "b70eac70-b4ac-11e7-9460-002564d9514e"
  }
]
```


Figure 24: Exemplary meta-alerts to be ranked


```
[
  {
    "ID": "b70eac70-b4ac-11e7-9460-002564d9514f",
    "Priority": 3
  },
  {
    "ID": "b70eac70-b4ac-11e7-9460-002564d9514e",
    "Priority": 5
  }
]
```


Figure 25: Calculated ranking of exemplary meta-alerts

The ranking-generator source code has been documented using Javadoc, below there is shown an excerpt from documentation showing implemented packages (Figure 26).



ruleLearn 0.8.6 API

Packages

Package	Description
org.rulelearn.approximations	Provides classes and interfaces for performing rough set analysis of data concerning sets of objects (like decision classes or unions of decision classes).
org.rulelearn.classification	Provides classes and interfaces for classification of objects with decision rule models.
org.rulelearn.core	Provides classes and interfaces defining some core operations, types and exceptions.
org.rulelearn.data	Provides classes and interfaces for working with data stored in information tables composed of fields representing identifiers/evaluations of all considered objects on all specified attributes, among which we distinguish: (1) identification attributes, (2) evaluation attributes: condition, decision and description ones, both active and non-active.
org.rulelearn.data.csv	Provides classes for handling data stored in CSV format.
org.rulelearn.data.json	Provides classes and interfaces for handling data stored in JSON format.
org.rulelearn.dominance	Provides classes and interfaces for calculating dominance relation and dominance cones.
org.rulelearn.measures	Provides classes and interfaces allowing to calculate and operate with different types of measures (i.e., measures defined for rules and objects).
org.rulelearn.measures.dominance	Provides classes and interfaces allowing to calculate and operate with different dominance-based consistency measures.
org.rulelearn.rules	Provides classes and interfaces for working with decision rules.
org.rulelearn.rules.ruleml	Provides classes and interfaces for working with decision rules models stored in RuleML.
org.rulelearn.types	Provides classes and interfaces representing basic field types.

Figure 26: Excerpt from Javadoc of ranking-generator

4 Ethical Impacts

There are no applicable ethical impacts during the correlation and prioritisation phase. Although, the contextualised meta-alerts need additional compliance checks in the case of sharing them with partners. As mitigation of potential sharing implications two following decisions has been made: there will be no sharing of meta-alerts containing data provided by CA module, inspector module will be implemented within work package WP5 to allow compliance checking of incoming and shared data.

In , the content of meta-alert has been analysed from the perspective of sharing possibilities.

The following color codes have been used:

- Red – the key should not be shared
- Yellow – the data needs filtering (e.g. pseudonymisation / conversion to network range) depending if it is local or not and internal policies
- Green – it can be shared freely.

Key	SubKey	Type	Description
ID		ID	Unique message (meta-alert) identifier.
EventTime		Timestamp	Timestamp of the moment of the event
DetectTime		Timestamp	Earliest timestamp of the DetectTime from the correlated events
Category		String of Categories	Category of meta-alert. Presumably, the set of categories will be the superset of already defined categories for conventional IDEA alert.
AggrID		String of ID	Identifiers of messages, which are aggregated into more concise form by the correlation engine.
TopAssetCriticality		Integer (Integer in range [0-4] (0-ND, 1-low, 2-med., 3-high, 4-critical))	Max asset criticality over source and target assets engaged in the reported attack/issue
TopMissionImportance		Integer (Integer in range <0-4> (0-ND, 1-low, 2-med., 3-high, 4-critical))	The highest importance of the mission influenced by the reported attack/issue
MaxCVE		CVE score	Maximal CVE score of known vulnerabilities that an impacted host has (retrieved from CA module)
MAQuality		Float (in the range of 0 to 1)	Summarizes the quality (trustworthiness) of a Meta-alert
Source			

Key	SubKey	Type	Description
	ReputationScoreSource	Float (in the range of 0 to 1)	Reputation/badness of the IP address of the attacker (i.e., “how probable is it to encounter this attacker in the future?”)
	Asset	String of AssetID (UUID)	Identifiers of the relevant assets as provided by CA module.
	MaxAssetCriticality	Integer	The maximal value of asset criticality over sources.
	Mission	String of MissionID (UUID)	Identifiers of the relevant missions related to the assets as provided by CA module.
	MaxMissionImportance	Integer	The maximal value of mission importance overall missions supported by the asset being identified as the source of an attack or issue.
	IP4	String of Net4	IPv4 addresses of this source (could be subnet address)
	IP6	String of Net6	IPv6 addresses of this source (could be subnet address)
	Proto	String of the protocols involved	Protocols, concerning connections from this source
	Hostname	String with the different hostnames	Hostnames describing the source. Should be FQDNs. An empty array can be used to explicitly indicate that values have been inquired and not found (missing DNS names).
	ASN	String of numbers	Autonomous system numbers of the sources
Target			
	Asset	String of AssetID (UUID)	Identifiers of the relevant assets as provided by CA module.
	MaxAssetCriticality	Integer	Maximal value of asset criticality over targets
	Mission	String of MissionID (UUID)	Identifiers of the relevant missions related to the assets as provided by CA module.
	MaxMissionImportance	Integer	The maximal value of mission importance overall missions supported by the asset being identified as the target of an attack or affected by some issue.
	IP4	String of Net4	IPv4 addresses of this target (could be subnet address)
	IP6	String of Net6	IPv6 addresses of this source (could be subnet address)
	Proto	String of the protocols involved	Protocols, concerning connections to this target

Key	SubKey	Type	Description
	Hostname	String with the different hostnames	Hostnames describing the target. Should be FQDNs. An empty array can be used to indicate that values have been inquired and not found explicitly (missing DNS names).
	ASN	String of numbers	Autonomous system numbers of the targets

Table 5: Meta-alert sharing – ethical considerations

5 References

[1] J. Błaszczyński, R. Słowiński, and M. Szeląg, “Sequential covering rule induction algorithm for variable consistency rough set approaches,” Information Sciences, vol. 181, no. 5, pp. 987–1002, 2011

[2] Intrusion Detection Extensible Alert, https://idea.cesnet.cz/en/index

Annex A: Additional WSO2 Details

WSO2 Data Analytics Server combines real-time, batch, interactive, and predictive (via machine learning) analysis of data into one integrated platform. One of the modules available in it is WSO2 CEP (Complex Event Processor), specifically designed for real-time analysis. For the current delivery only the CEP module has been used, so that, in this section we focus on specific WSO2 CEP capabilities.

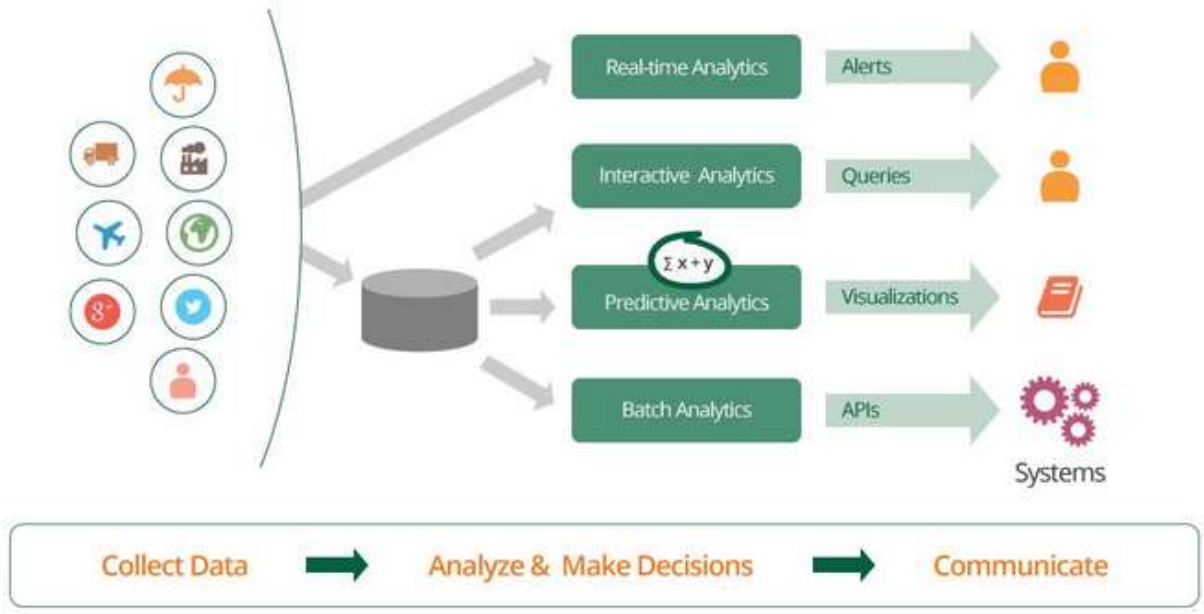


Figure 27: WSO2 DAS Architecture

Figure 28: WSO2 DAS Management Console

WSO2 CEP Architecture

The CEP engine is composed of distinct agents listed as follows:

- Event Receiver

Event receivers receive events that are coming to the CEP. WSO2 CEP supports the most common adapter implementations by default. For specific use cases, you can also plug custom adapters.

- Event Streams

Event streams contain unique sets of attributes of specific types that provide a structure based on which the events processed by the relevant event flow are selected. Event streams are stored as stream definitions in the file system via the data bridge stream definition store.

- Execution Plans

Event processor handles actual event processing. It is the core event processing unit of the CEP. It manages different execution plans and processes events based on logic, with the help of different Siddhi queries. Event Processor gets a set of event streams from the Event Stream Manager, processes them using Siddhi engine, and triggers new events on different event streams back to the Event Stream Manager.

- Event Publishers

Event publishers publish events to external systems and store data to databases for future analysis. Like the event receivers, this component also has different adapter implementations. The most common ones are available by default in the CEP. One can implement custom adapters for specific use cases.

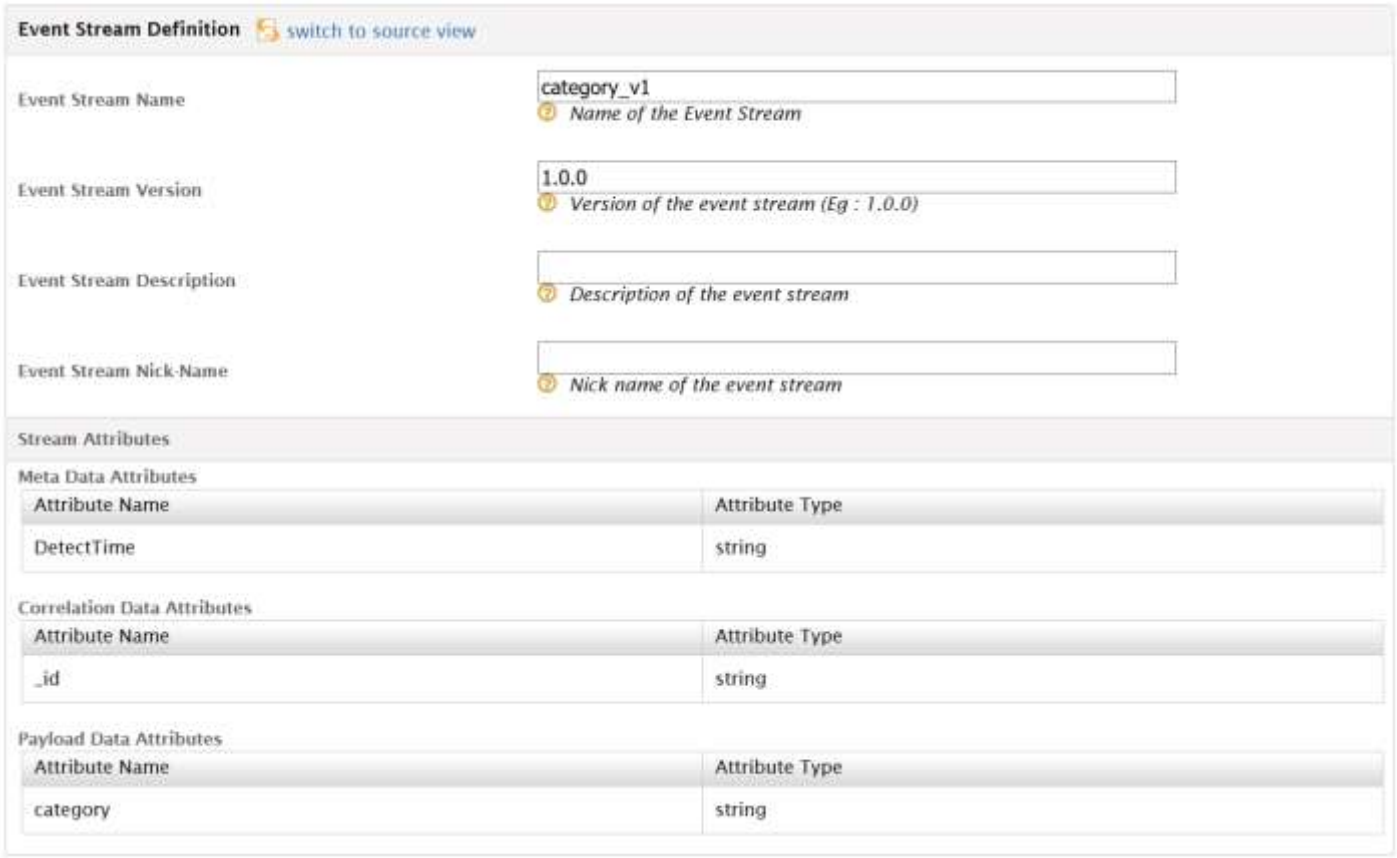


Figure 29: WSO2 DAS Category Stream Graphic Definition

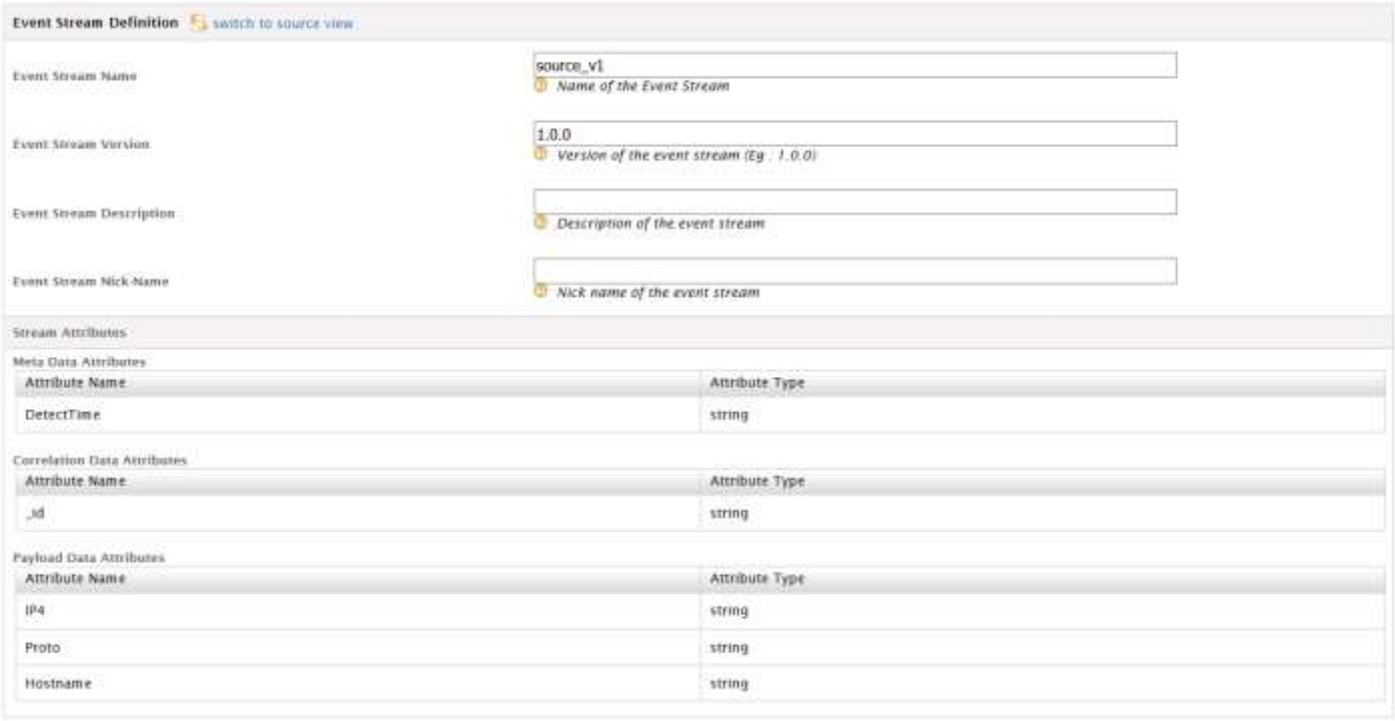


Figure 30: WSO2 DAS Source Stream Graphic Definition

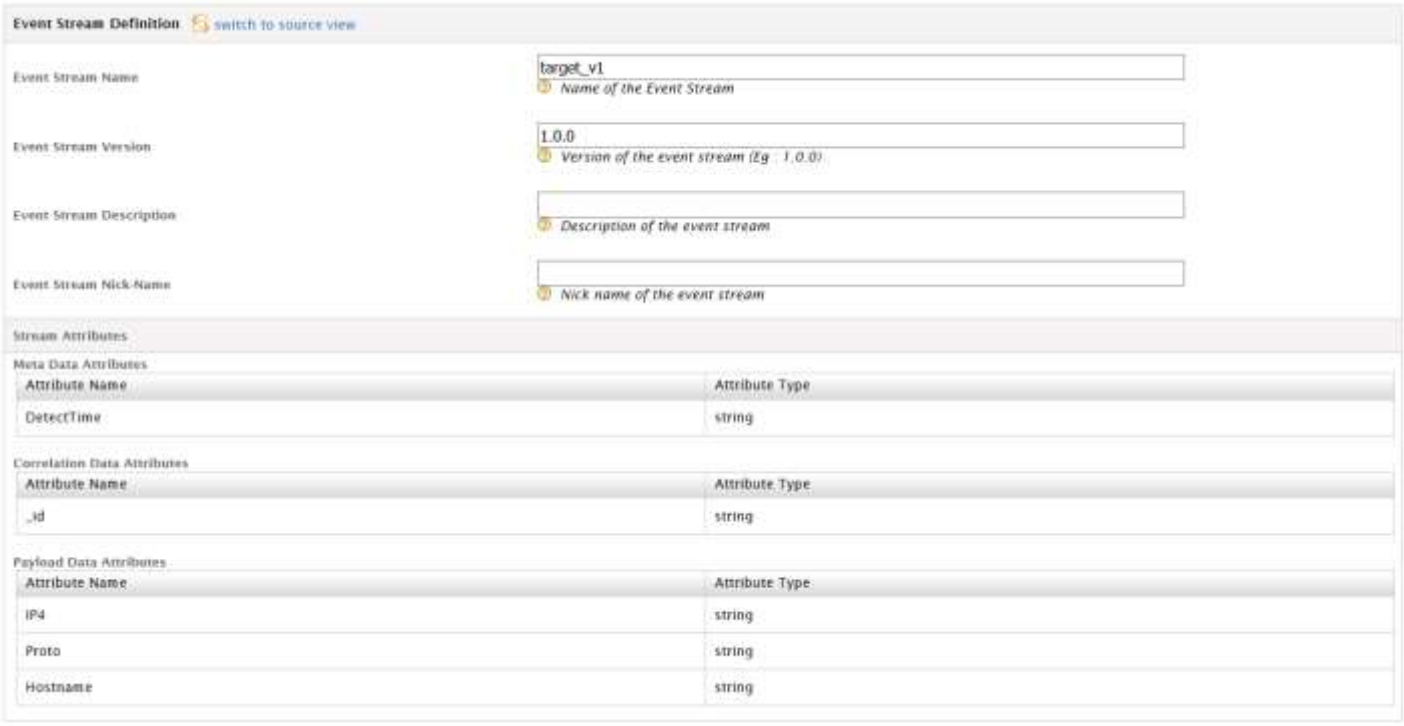
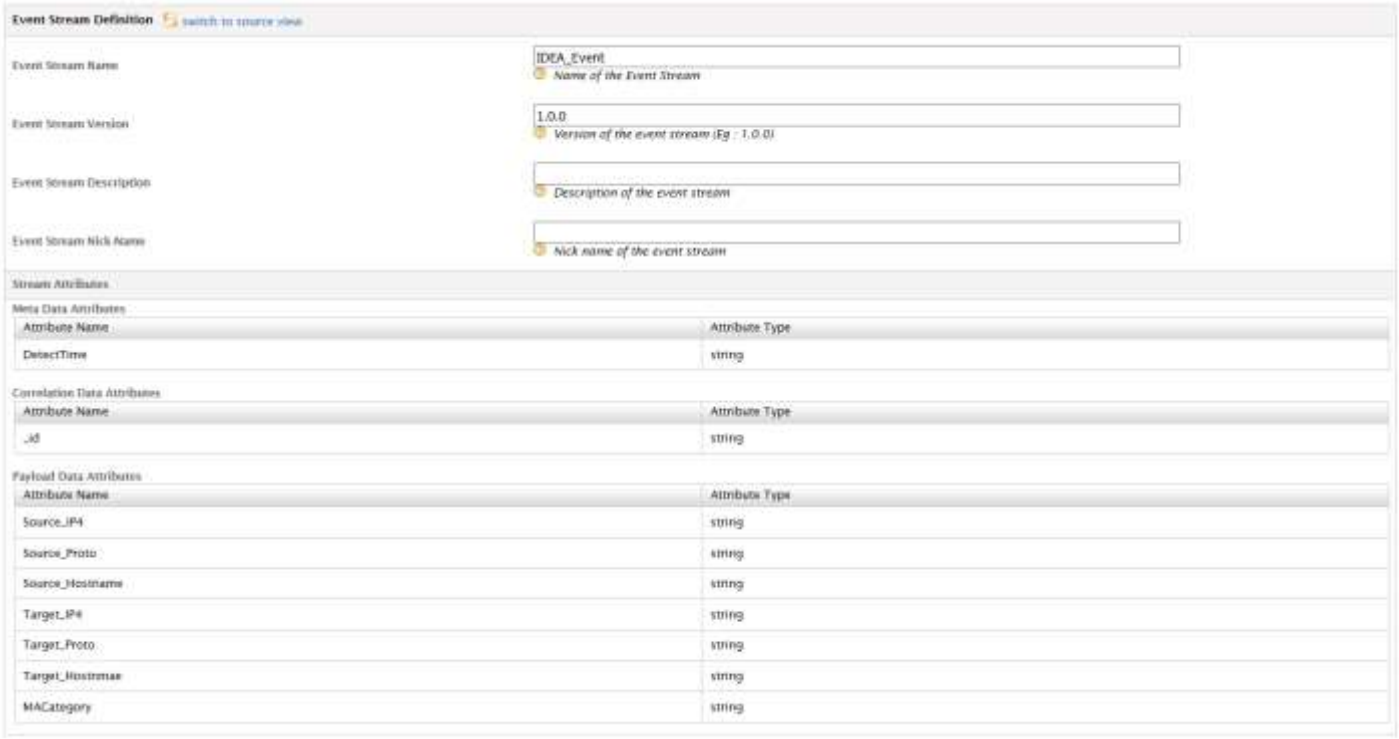


Figure 31: WSO2 DAS Target Stream Graphic Definition

Figure 32: WSO2 DAS IDEA_Event Stream Graphic Definition



Execution plans

Once the IDEA_alerts are generated, it is time to correlate them through the predefined execution plans (or rules).

In the current implementation twenty execution plans has been provided with rules allowing for correlation of alerts and producing output stream of meta-alerts. The meta-alerts are further correlated on temporal basis using time-windows.

Currently implemented rule-based execution plans for correlating alerts:

1. DetectSabotage
   Warn if some Sabotage action is preceded by some potential system compromise activity (Intrusion, unauthorized action, attempt of exploiting or anomaly connection) within 10 minutes
2. Intrusion_and_malware
   Warn if an Intrusion or Attempt to a Target IP is followed by a Malware events from the same Target_IP
3. Multiple_Recon_Scanning_SameSource
   Warn if a Source performs 3 Scans events within a minute
4. Multiple_Recon_Scanning
   Warn if a target receives 3 Scans events within a minute
5. Exploit_and_UnauthorizedAction
   Warn whether an Attempt.Exploit or Attempt.NewSignature event is followed by some unauthorized action (Information category event) for the same target within 10 minutes
6. UnusualUDP_Traffic
   Report Anomaly.Traffic events using UDP protocol
7. Exploit_and_Anomaly
   Warn if a Target generates an Anomaly category event after an attempt exploit event within 10 minutes
8. LoginBruteforce_and_Connect
   Warn whether multiple login attempts are followed by an anomaly connection
9. Attacked_Target_Compromised
   Warn whether the source of an attack was previously the destination of an attack (within 10 minutes)
10. Multiple_AnomalyConnection
    Warn if multiple anomaly connections to a same Target IP are detected
11. Scanning_and_Connect_Backward
    Warn if a host scan is made by an IP and then if a successful connection is established (Anomaly.Connection, Information.UnauthorizedAccess or Intrusion event category) by the same IP and then backward connection is established from connected IP to connecting IP within 10 minutes
12. Multiple_AnomalyConnection_DistinctPort
    Warn if multiple anomaly connections to a same Target IP and distinct ports are detected
13. Fake_DHCPServer
    Warn if an anomaly traffic event, using the UDP protocol and the port 67 or 68 is detected
14. Bruteforce_Login_SameTarget
    Warn if three login attempts events are registered from the same source to the same target within 10 minutes

15. Phising_and_MalwerInfection
    Warn if a Phising event to a Target IP is followed by a Malware event from the same Target IP (It could mean that the phising worked and the target is now infected) within 10 minutes
16. UnauthorizedAction_OpenBackdoor
    Warn if an unauthorized action events is followed by a vulnerable alert event. (It could means that the attacker opened a backdoor) within 10 minutes
17. Scanning_and_Exploit
    Warn if a Scan event is followed by an exploit attempt from same source within 10 minutes
18. Scanning_and_Connect
    Warn if a Scan event is followed by an anomaly connection from same source within 10 minutes
19. Bruteforce_login_DistintTarget
    Warn whether a Source performs three login attempts within a minute to distinct Targets
20. Scanning_and_Login
    Warn if a Scan event is followed by a Login attempt from same source within 10 minutes

In addition to above mentioned correlation rules, aggregation rules for alerts and meta-alerts has been implemented.

Publisher

The returned Meta-alert format is as follows.

```
{
 "name": "MetaAlert",
 "version": "2.0.0",
 "nickName": "",
 "description": "",
 "metaData": [
  {    "name": "ID", "type": "STRING"   },
  {    "name": "DetectTime", "type": "STRING"   }
 ],
 "payloadData": [
  {    "name": "Category", "type": "STRING" },
  {    "name": "AggrID",  "type": "STRING"   },
  {    "name": "TopAssetCriticality", "type": "STRING"   },
  {    "name": "Rule", "type": "STRING"   },
  {    "name": "MAQuality", "type": "FLOAT"   },
  {    "name": "Source_Asset",  "type": "STRING"   },
  {    "name": "Source_MaxAssetCriticality", "type": "STRING"   },
  {    "name": "Source_Mission", "type": "STRING"   },
  {    "name": "Source_MaxMissionImportance", "type": "STRING"   },
  {    "name": "Source_IP4",  "type": "STRING"   },
  {    "name": "Source_Proto", "type": "STRING"   },
  {    "name": "Source_Port",  "type": "STRING"   },
  {    "name": "Source_Hostname", "type": "STRING"   },
  {    "name": "Target_Asset", "type": "STRING"   },
  {    "name": "Target_MaxAssetCriticality", "type": "STRING"   },
  {    "name": "Target_Mission", "type": "STRING"   },
  {    "name": "Target_MaxMissionImportance", "type": "STRING"   },
  {    "name": "Target_IP4",  "type": "STRING"   },
  {    "name": "Target_Proto",  "type": "STRING"   },
  {    "name": "Target_Port",  "type": "STRING"   },
  {    "name": "Target_Hostname",  "type": "STRING"   }
 ]
}
```

Table 4: WSO2 Meta-alert publisher format

The fields with no content can't be fulfilled by the correlation module, they are competence of the next modules (enrichment, context awareness or priorisation), hence they are being returned with no value. The Meta-alerts are shown in the console (MetaAlert_Logger publisher) and also sent to the correlation-pipeline (MetaAlert_HTTP publisher).