**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

# D3.1 Event correlation mechanisms report

| Deliverable Details | |
|---|---|
| **Deliverable Number** | **D3.1** |
| **Revision Number** | **E** |
| **Author(s)** | **GMV/PSNC/TUDA/UOXF** |
| **Due Date** | **0817** |
| **Delivered Date** | **31/08/2017** |
| **Reviewed by** | **SYNYO/AIT** |
| **Dissemination Level** | **PU** |
| **Contact Person EC** | **Alina-Maria Bercea** |

| Contributing Partners | |
|---|---|
| **1.** | **GMV (contributor)** |
| **2.** | **PSNC (contributor)** |
| **3.** | **TUDA (contributor)** |
| **4.** | **UOXF (contributor)** |
| **5.** | **AIT (reviewer)** |
| **6.** | **SYNYO (reviewer)** |

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.

# Revision History

| Revision | By | Date | Changes |
|---|---|---|---|
| E | GMV | 31/08/2017 | Version Submitted to Agency |
| B3 | AIT | 30/08/2017 | Updated with further comments from AIT |
| B1,B2 | SYNYO, AIT | 29/08/2017 | Updated with individual comments |
| A2 | PSNC | 01/07/2017 | Working version on Google Docs |
| A1 | PSNC, GMV | 25/05/2017 | Template applied, ToC proposed |

# Executive Summary

This deliverable contains an overview of the correlation process that can be applied in the PROTECTIVE project.

Alerts must first be put into a common format. Once this task is accomplished, the next step is to filtrate and aggregate, in order to discard unwanted alerts and to reduce, if possible, the disk space needed for the alerts that we do keep. Further down the pipeline, alerts are enriched, so that external information can be incorporated into the correlation process, which is the following step.

Alerts can be correlated using different approaches, each one having its advantages and drawbacks, which are briefly explored in this document. In addition, other topics intrinsic with the issue at hand, log format and storage and search engines, are briefly discussed.

The document has the following structure. Sections 2 and 3, specify the internal format of alerts and how they can be enriched. Next, in section 4, we describe the specific process of correlation, the pipeline proposed by some authors and how the volume of events to process can be reduced. We also briefly explain the different methods and compare them. In section 5 we give the details of some of the event sources used in PROTECTIVE. Furthermore, we provide insights into different normalization methods, the ways logs can be stored and the use of search engines. In section 6, we compare several correlation engines and select the one we will use in the 1st pilot.

Finally, we include three annexes:

- Annex A, in which there are several alert examples.
- Annex B, which contains some definitions.
- Annex C, where an example of alert correlation is given.

# Abbreviations List

| | |
|---|---|
| CEP | Complex Event Processor |
| CVSS | Common Vulnerability Scoring System |
| IDS | Intrusion Detection System |
| IPS | Intrusion Prevention System |
| KVP | Key-Value Pairs |
| LAMS | Local Adaptive Multivariate Smoothing |
| MA | Meta-Alert |
| NIDS | Network Intrusion Detection System |
| TC | Trust Computation |
| TI | Threat Intelligence |

**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

# List of Tables

# 1 Introduction

In this document, we are going to describe the process of alert correlation as designed in the PROTECTIVE project. To be able to correlate alerts, they must first be put into a common format. Once this task is accomplished, the next step is to filtrate and aggregate, in order to discard unwanted alerts and to reduce, if possible, the disk space needed for the alerts that we do keep. Further down the pipeline, alerts are enriched, so that external information can be incorporated into the correlation process, which is the following step.

Alerts can be correlated using different approaches, each one having its advantages and drawbacks, which are briefly explored in this document. In addition, other topics intrinsic with the issue at hand, log format and storage and search engines, are briefly discussed.

Finally, different tools for developing the correlation mechanism are considered, selecting the one considered to be the most suitable.

# 2 Events, Alerts and Meta-alerts: General Description and Assumptions

Incoming security alerts or events take the form of sets of Key-Value Pairs (KVPs). Among alerts, we can distinguish individual types of alerts - e.g. from the same version of the Network Intrusion Detection System (NIDS) - coming from one or more sources. The same type of alert can be produced by multiple sensors, of the particular kind, spread across the monitored environment. Each alert has a known source of origin. Information about the origin is part of the alert content, or the handlers processing incoming streams of events add it to its content. The same is the case with alert types. The keys present in the alerts are further also called attributes of the alert. The values have fixed data types so that we can assume that attributes are of the particular data type. It is important to note that various keys, in different types of alerts, may describe the same data. Thus a mapping of all those keys to unified representation is a need. To address that need a dictionary of attributes will be created along with a map of attributes correspondence. The dictionary will contain at least attribute name, its data type and unique identifier. The process of converting alerts to unified representation is a normalization. Besides of direct mapping of attributes application of additional conversions (e.g. string to integer or more complex ones) may be necessary during normalization step. It is worth to note that data types may also be a complex ones e.g. array of objects.

After normalization, alerts are further processed using techniques (Pouget & Dacier, 2003) like compression or aggregation (an aggregation is a form of lossless compression), filtering (reduction of information using values of particular features/attributes), selective suppression (stateful reduction), thresholding, modification (addition of missing attributes), generalization, specialization. The actual set of methods is subject to research and state of the art analysis within WP3 as well as technological constraints. We can call this step as a pre-processing phase. The choice of methods also depends on the volumes produced by alerts/events sources. In the case of a large volume of similar alerts/events, in-memory processing may be needed. The large volume of incoming data may appear in the case of local sources.

The next phase of processing is a correlation of normalized and pre-processed alerts that leads to the creation of meta-alerts. Meta-alerts are objects consisting of all common attributes of correlated alerts, a list of data sources and links to original alerts.

After initiation of the meta-alert object, the addition of data occurs as a result of an enrichment process. We can complement the initial meta-alert by, for example, providing statistics about the number of same events stored in the database and by adding assets criticality score for given IP/service and by adding information about possible attack classes, data from trust component. Enrichment phase is the phase where incorporation of a vast number of sources of Threat Intelligence (TI) into alert processing workflow is possible.

## 2.1 Alerts Definition

According to the NIST document (NIST, 2013), proposed as one of the main sources for terminology used in PROTECTIVE, an alert is "Notification that a specific attack has been directed at an organization's information systems". This definition has origins in the document provided by US Committee on National Security Systems (CNSS, 2015).

Alerts have different forms (cf. examples in the annexes), contain different types of information on disparate levels. Regarding alert definition, it leads to the conclusion that it should be general, which conforms to the definition provided above. Regarding alert processing (including building meta-alerts), it emphasizes the need to unify and normalize alerts before further analysis.

For further considerations, assume that the alert fulfils the above definition and that it contains a set of Key-Value Pairs (the keys are also called attributes). The figure below shows the high-level definition of an alert.

| Attribute (key) | Value |
|---|---|
| $A_1$ | $V_1$ |
| ... | ... |
| $A_n$ | $V_n$ |

Figure 1 General alert structure

A similar, and even more general, definition of alert is provided in the presentation provided by Aziz (Aziz, 2007), where it is assumed that alert is "generated by Intrusion Detection System (IDS) to notify of the interesting events". That definition does not limit alerts to inform about cyber-attacks. As the mentioned presentation concerns IDSs, it assumes that only this category of systems may generate alerts, which is not true for PROTECTIVE.

A more formal definition found in (Kruegel, et al., 2005) states that: an alert A is a list of pairs of attributes $a_i$ with their corresponding value sets $v_i$ that is, $A=\{(a_1,v_1),(a_2,v_2),...,(a_n,v_n)\}$. Each attribute $a_i$ of an alert describes a certain property (or feature) of the attack that this alert refers to (e.g., name and target of the attack).

To summarize, we will consider alert as a notification with a known structure, composed of a list of attribute-value pairs, indicating that an organization's information systems have been a target of a specific attack. The indication of compromise or attack can be indirect, for example when systems of an organization are a source of an attack.

## 2.2 Alert Unification / Normalization

As shown in the "Annex A: Examples of Alerts", alerts may have a significantly different scope and form (in certain cases it may not be possible to unify two arbitrary alerts). To be able to process further a set of alerts from different sources, need for their unification to the maximum possible extent arises.

For instance, attribute denoting the system that the attack is originating from may be described as "src IP" in the alert A and "source IP address" in the alert B. These names need unification. Another example may be that two systems, providing data from their log systems, work in different time zones (or just experience slight time differences in their system clocks). To preserve the timeline of particular events, that may concern both systems, the timing of all events need unification (not necessarily to the timing of one of these two systems, but rather to a timing universal for the correlation system).

An example of attributes normalization/unification is provided below.

| Attribute (key) | Value |
|---|---|
| src ip | 1.2.3.4 |
| dest ip | 11.22.33.44 |
| time | 10:12:34.567 |
| timezone | GMT |

| Attribute (key) | Value |
|---|---|
| src | 1.2.3.4 |
| dest | 11.22.33.45 |
| time | 11:12:38.234 |
| timezone | CET |

Figure 2 Alert 1 (left) and Alert 2 (right) from different alert sources

| ID | source | destination | time GMT |
|---|---|---|---|
| 1 | 1.2.3.4 | 11.22.33.44 | 10:12:34.567 |
| 2 | 1.2.3.4 | 11.22.33.45 | 10:12:38.234 |

Figure 3 Two Alerts 1 and 2 normalized, ready to be integrated

### 2.2.1 Dictionary / Ontology of Attributes

To appropriately understand all processed alerts, their internal structure must be recognized, for all types of alerts, and defined in the form of a dictionary or ontology. The attribute dictionary might look as follows:

| ID | Name | Type | Format |
|---|---|---|---|
| 1 | Sender | String | Enum {IDS, FW, SIEM, ...} |
| 2 | Attack dst | IP | Is_array |
| 3 | CVSS | Float | {0.0..10.0} |

Figure 4 Alert attributes dictionary

The dictionary should contain the following entries:

- ID – unique identifier of attribute for further reference.
- Name – attribute name (if it is unique, the ID column might not be necessary).
- Type – data type of the attribute.
- Format – a generalized field describing the possible values of the attribute. For instance, it may contain information like:
    - Possible values of enumerative types (e.g. names or identifiers of all possible source of alerts).
    - Scopes of numeric types (e.g. CVSS score).
    - Flag indicating if the attribute value may be an array (e.g. an alert may contain the list of internet protocol addresses that are attacked instead of a single address).
    - Regular expression defining the textual representation of the attribute value (e.g. definition of email address of the e-mail alert sender).
    - Flag indicating if the attribute value may be empty.
    - The default value of the attribute.

It is necessary to assure that the dictionary is extensible as the alert structure may vary over time due to standard modifications, alerting system upgrades, and similar issues.

### 2.2.2 Attributes Correspondence Mapping

As alerts from different sources will have different ontologies, there is a need to provide correspondence mapping to build a common ontology that will facilitate further processing of the alerts. The common ontology is crucial during normalization of the events.

Similarly to the extensibility of alert attribute ontology, it is obvious that the catalogue of all defined alerts (and thus the correspondence mapping) must be extensible to enable acquiring new sources of alerts. The need for the common ontology within the normalisation task is further elaborated in Section 5.2.

## 2.3 Meta-alert General Concept Including Visual Form

### 2.3.1 Meta-alert General Concept

Meta-Alert (MA), by intuition, is a structure that contains information from multiple alerts (but most probably only a relevant subset of that information). More formally, the meta-alert is a result of the merging of two or more related alerts, merged as a part of the alert correlation process (Kruegel, et al., 2005).

To introduce more flexibility to that definition, for further considerations let us assume that MA may not be only a combination of two or more alerts, but two or more objects that may be alerts or other MA.

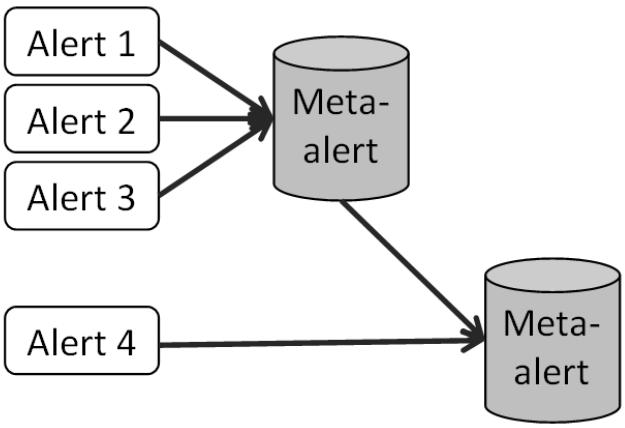


Figure 5 Meta-alerts integrating information from alerts or alerts and other meta-alerts

MAs are objects consisting of all common attributes of correlated alerts and MAs, a list of data sources and links to original alerts and meta-alerts, as well as additional data derived as a function of the attributes of the merged alerts. The general purpose of meta-alerts is to aggregate information of related attacks and at the end present a single alert instance that summarizes all the relevant information to a human analyst.

It is worth to note that internal representation of MAs is oriented towards efficient machine processing and robust application of multi-criteria decision aiding algorithms. Thus for the sake of clarity and high end-user experience, it is necessary to propose consistent visualization of all combined data. General example of the visual structure of MA is described in the next chapter.

The MA representation is consistent with alert representation (see figure below). Please note that all attributes present in MA must have a description in the attributes dictionary/ontology (cf. Section 2.2.1 Dictionary / Ontology of Attributes).

In case of temporal meta-alerts, the time stamp of the MA will be the minimum time stamp of source alerts and for multi-value fields it is the union of all values (Kruegel, et al., 2005).

| Attribute (key) | Value |
|---|---|
| ID | unique identifier of MA |
| AssetID | identifier of the local asset involved |
| AssetCriticality | Asset Criticality Score provided by AIT |
| TrustScores | table of Trust Scores for data sources of combined alerts (partially provided by TUDA) |
| SourceAlerts | References to alerts combined within MA (actual method of implementation to be determined) |
| Common attribute $A_1$ | $V_1$ [values determined/extracted in correlation process] |
| ... | ... |
| Common attribute $A_n$ | $V_n$ [values determined/extracted in correlation process] |

Figure 6 General concept of meta-alert (MA)

### 2.3.2 Conceptual Proposition of Meta-alert Visual Form

The internal representation of MA is data-centric as it must facilitate efficient and robust correlation and enrichment process. However, this does not have to be consistent with what should be shown to the system operator (as it has been mentioned earlier, the assumed intention of merging alerts into meta-alerts is to present human-readable information about an attack).

The following figure shows the proposed concept of a Meta-Alert (MA) visualization, containing general information necessary to be provided with MA.

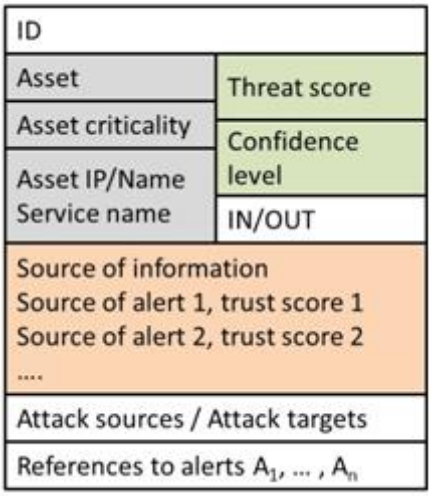


Figure 7 General concept of meta-alert (MA) visualization

The fields shown in the figure (on a very general level) are as follows:

- ID – unique identifier of the MA, for any further reference and analysis.
- Information about the Asset concerned (being attacked).
  - Asset criticality – importance of the asset concerned, which – if the asset is located in the protected infrastructure – will be provided by the inventory module.
  - Asset IP/Name or Service name – further description of an asset, e.g. IP address, port number, service name, TCP/UDP indicator, custom description.
- Threat Score – aggregated threat score, calculated appropriately, basing on properties of alerts that compose the MA.
- Confidence Level (overall trust) – the level of confidence or trust associated with MA. This level must be accurately correlated by combining trust scores associated with the individual alerts composing MA. Trust levels will be provided by the PROTECTIVE component implemented by TUDA.
- IN/OUT – indicator whether the MA concerns an attack aimed at the protected infrastructure (IN) or an attack originating from the protected infrastructure (OUT).
- Key information on alerts composing the MA:
  - Identifier of the source of the alert $A_i$
  - Trust score of the alert $A_i$

    - Attack source – description of the origin of the attack. Please note that if the attack is outgoing, it may be the IP address (or another reference) of an internal system.
- References to alerts $A_i$ – information that allows reaching the composing alerts to obtain more information about the associated alerts, when necessary (e.g. for a more detailed analysis by a system operator). Storing all relevant information in MAs seems infeasible as not every MA will be thoroughly analysed and an excess information would cause a loss of efficiency of storing and processing MAs. On the other hand, it must be assured that information on the underlying alerts is not lost in time.

The proposed structure is a subject to be extended, for instance with contextual information about the MA. MA may contain both low-level information (like IP addresses, URLs, etc.) and high-level contextual information (for instance, suggested steps of mitigation copied from composing alerts).

# 3 Enrichment of alerts and meta-alerts

Enrichment process is one of the most important steps within PROTECTIVE workflow as it is shown in Figure 8 Enrichment phase within PROTECTIVE workflow.

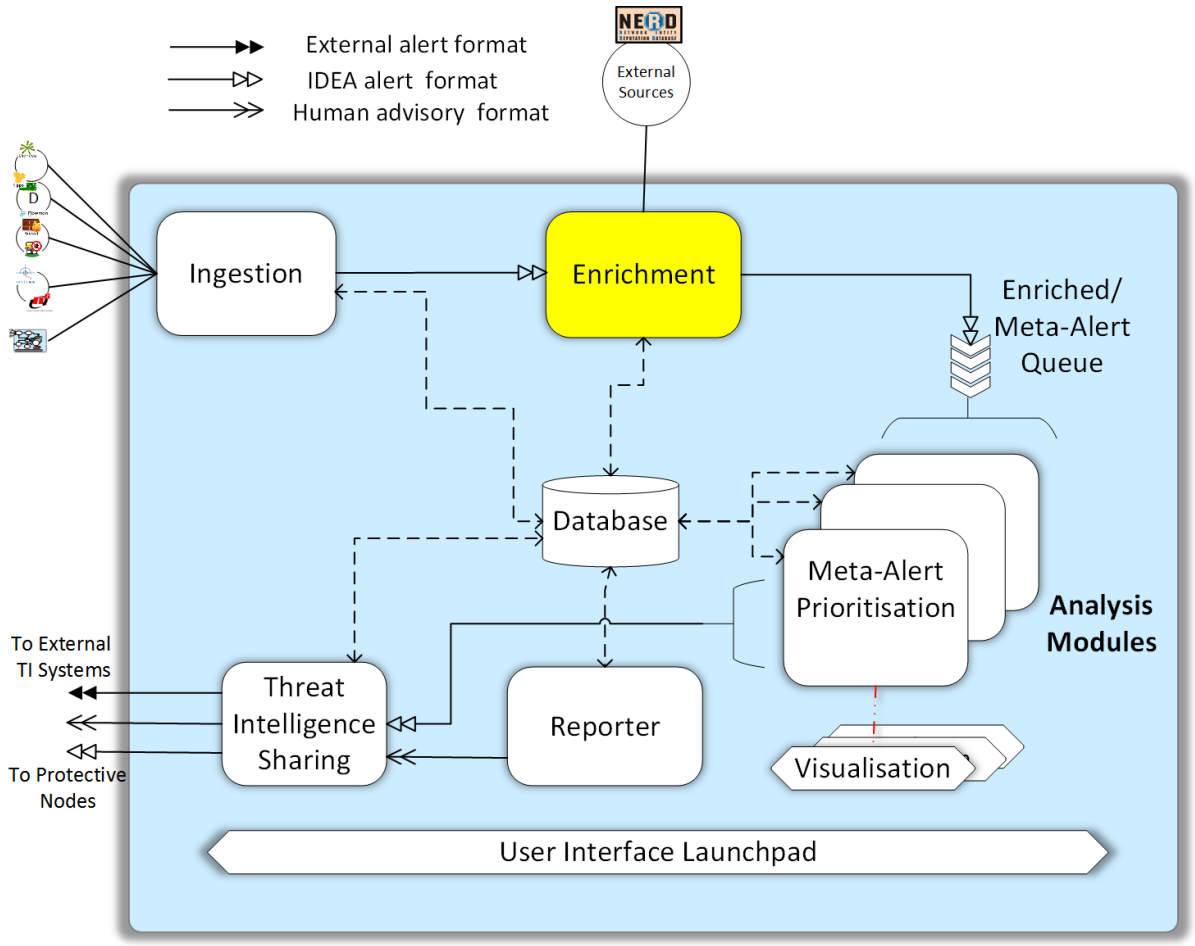


Figure 8 Enrichment phase within PROTECTIVE workflow

The actual process of enrichment takes place in two phases of the PROTECTIVE workflow (c.f. Figure 8):

- While processing alerts during correlation.
- During processing of correlated data in form of meta-alerts.

Enrichment, as it was defined in D2.1, is realised by correlation with multiple databases using various elements of the collected information like addresses and identifiers. These databases can be internal to the organisation or access to them can be provided by an external service. Within enrichment subsystem additional data is annotated to the alerts to aid with their further processing. Much such information is intended to capture the state of the current security context and therefore has a short lifetime as the security situation can change rapidly. There is also a trade-off with as to how long such enrichment data should be persisted.

Part of the data usable in enrichment process (c.f. Figure 9 Enrichment phase of PROTECTIVE workflow - detailed view) will be delivered internally within PROTECTIVE by the Context Awareness module (c.f. D4.1) and Trust Component (c.f. D5.1). Internal data will be complemented by external sources. Within

the deliverable D6.1, several components useful in the enrichment process have been identified. Among those components, the gathering of data from external sources is done by the following solutions and information sources:

- Mentat-enricher - – enables the enrichment of incoming data through the following sequence of tasks: IDEA notification validation, resolving target abuse's contact (for the reporting purposes), detection of event's specific type (to enable notification formatting). Implementation of further operations is planned: hostname/IP resolving, passive DNS, geolocation etc.
- Cortex engine (being part of TheHive project) – this engine has a plug-in architecture that enables various "Analyzers" to be added to automate observable analysis: geolocation, VirusTotal lookups, DNS lookups. In terms of the ENISA processing pipeline these analysers provide an enrichment function. Currently Cortex is capable to acquire data from 24 sources.
- Apache Spot – still in incubation phase.
- OpenSoc – discontinued.

Therefore, Mentat-enricher and Cortex seem to be most appropriate candidates to be implemented within the pilot.

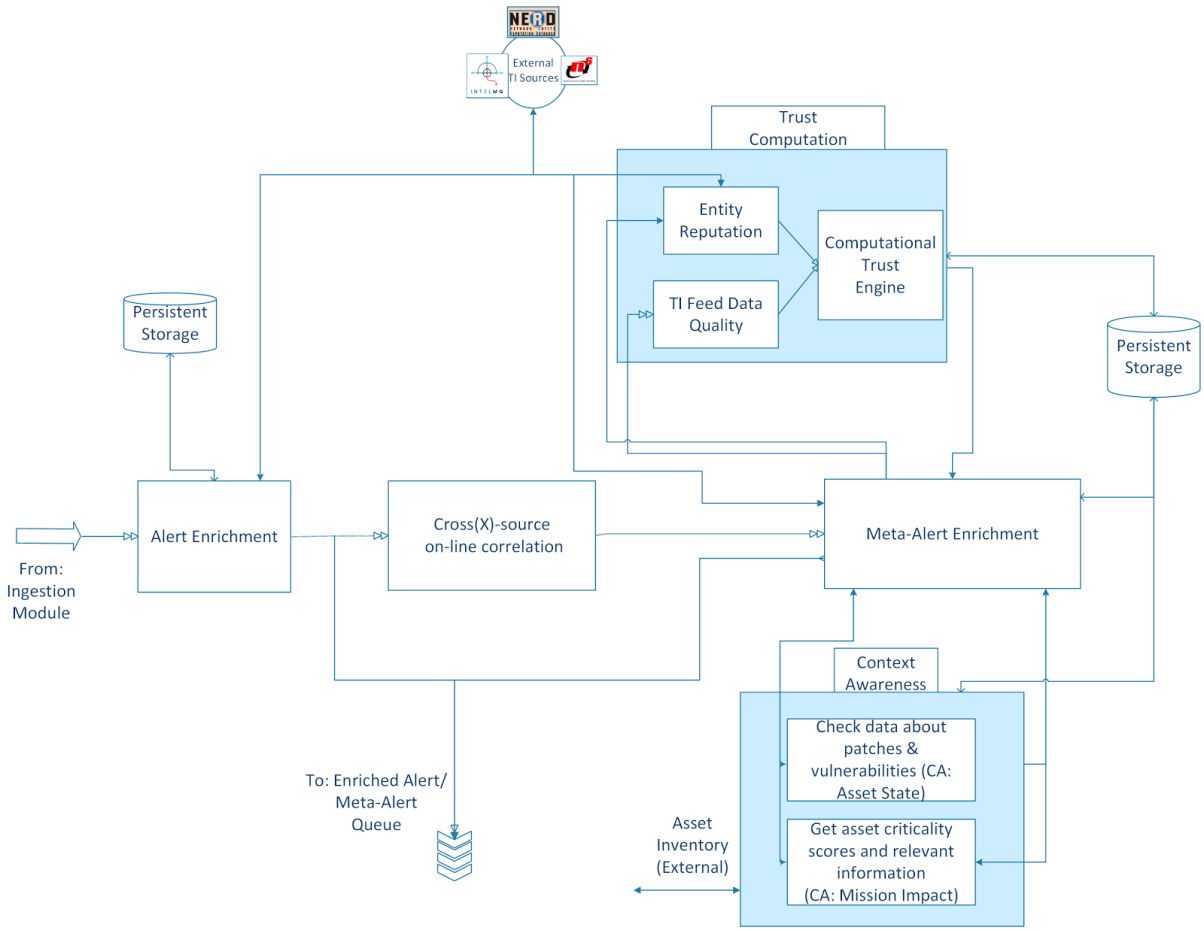


Figure 9 Enrichment phase of PROTECTIVE workflow - detailed view

Furthermore, external feeds will not be trusted by default but a trust/reputation score will be calculated by the TI Trust Computation (TC) module. The TC module aims to improve the management, sharing, and prioritisation of threat intelligence within the community of NRENs and SMEs by determining the quality (or reputation) of TI feeds, i.e. how "good" is the feed itself. Additionally, the Trust component aims to determine the quality of a particular malicious entity, i.e. how "bad" is an entity (e.g. IP address) associated with a feed. The first variant aims to improve the management and sharing of TI feeds with respect to their quality ("goodness") and the second variant aims to improve the prioritisation of TI feeds with respect to their level of maliciousness ("badness"). Therefore, we can say that the Trust Component deals with two contexts: one is to determine the reputation of threat intelligence feeds and another is to determine the reputation of malicious entity associated with feeds. This is realised by the TI Feed Data Quality and the Entity Reputation component in the figure above. The scores regarding Data Quality and Entity Reputation, provided by these two systems, are added to the alert which is then passed to the Prioritisation subsystem.

# 4 Correlation and Aggregation Methods

The portfolio of methods and approaches that can be used to facilitate either the correlation or aggregation process is extremely broad. Thus within this section, a brief overview of possibilities is provided. Both tasks can be applied to either low-level alerts or objects on higher abstraction level called meta-alerts (cf. Events, Alerts and Meta-alerts: General Description and Assumptions).

The alert aggregation process aims in the reduction of alerts volume by combining individual alerts into their summarized or condensed form (Yu Beng, et al., 2014), in perfect conditions without loss of valuable information from the perspective of the incident handler. Thus aggregation aims to reduce the volume of alerts and to increase the information content within a single aggregated alert.

The correlation process, by definition, derives new knowledge by identifying dependencies or associations among data. Correlation, in case of security systems, provides insights that cannot be inferred from single entities – single alert or single source of data (e.g. single location).

Salah, Macia-Fernandez and Diaz-Verdejo (Salah, et al., 2013), proposed a correlation process model composed of four modules: alert pre-processing, alert reduction, alert correlation and alert prioritization. That view is roughly consistent with the workflow proposed for PROTECTIVE. Alert pre-processing in their model corresponds to the Ingestion capabilities where the normalization task takes place. Alert reduction (including aggregation) and correlation within PROTECTIVE is part of Enrichment module and is further discussed in the following subsections. Alert prioritization is discussed within D3.2.

## 4.1 Reduction of Data Volume by filtering and aggregation

Alert data volume reduction methods overcome the problem caused by alert flooding or large volume of alert data generated by different IDS. The higher the capability of alert data volume reduction method, the lower manual effort will be required by the network security team in analysing and identifying the attackers in their environment. Some alert data volume reduction methods are discussed as follows:

### 4.1.1 Reducing data volume by false alert reduction

In this section, those methods are discussed which distinguish between false positive and true positive alerts. This distinction removes the delay caused by analysis of false positive alerts and improves the system efficiency. Various false alert reduction approaches are discussed as follows:

Alshammari et al. (Alshammari, et al., 2007) proposed neuro-fuzzy based approach to reduce number of false positive alarms. Hybrid neuro-fuzzy approach generate fuzzy rules for classifying true or false positives using historical alert data information. In an experiment, it is observed that trapezoid membership function with IP class background knowledge had the best ability in detecting false alarms.

Grill, Pevný and Rehak (Grill, et al., 2017), used Local Adaptive Multivariate Smoothing (LAMS) method for reducing large number of unstructured false positives caused by stochasticity of the network traffic. Experiment results shows reduction of structured false positives without impacting the

remaining cases. The proposed mechanism first formulates structured and unstructured false positives and argues why unstructured false positive are more difficult to remove then proposes sound method to decrease the rate of unstructured false positives.

Hubballi, N. and Suryanarayanan (Hubballi & Suryanarayanan, 2014), surveyed false alarm minimization techniques in signature-based NIDS. Various false alarm minimization techniques can be classified as: signature enhancement, stateful signature, vulnerability signatures, alarm mining, alarm correlation, alarm verification, flow analysis, alarm prioritization and hybrid methods. These techniques are briefly explained as follows:

- Signature enhancement based scheme uses few additional information for verification along with attack signature.
- Stateful signature-based approaches take state of network into account for improving the performance.
- Vulnerability signature-based scheme uses application semantics and shows improvement of performance in terms of accuracy.
- In alarm mining technique, data mining is used for reducing false alarms such as alarm classification, alarm clustering, neural network, frequent pattern mining etc.
- In alarm correlation techniques, alarms are aggregated to predict the attack scenario. Examples of alarm correlation techniques are: multi-step correlation, knowledge based, complementary, casual relation based, fusion based, attack graphs based, rule based etc.
- Alarm verification is based on the outcome of the attack and verification of its impact on the system.
- Technique based on flow analysis uses a set of alarms generated under normal and abnormal scenarios for minimizing false positive.
- Technique based on alarm prioritization rates the alarms based on post assessment or some evaluation.
- Finally, hybrid approach combines two or more approaches together.

### 4.1.2 Other methods

According to (McConnell & Siegel, 2004), alert volume can be reduced using following methods:

- **Roll-up Method:** In this method, hierarchical collector structures are used to reduce alert volumes. These hierarchical structures reduce volume by roll-up from one level to the next. Figure shows three collectors using virtual transactions against the same server. Let's suppose server is congested. Now, all collectors will forward a server slow alert to the aggregator which simply passes a single server slow alert to event manager or another level in the instrumentation hierarchy.

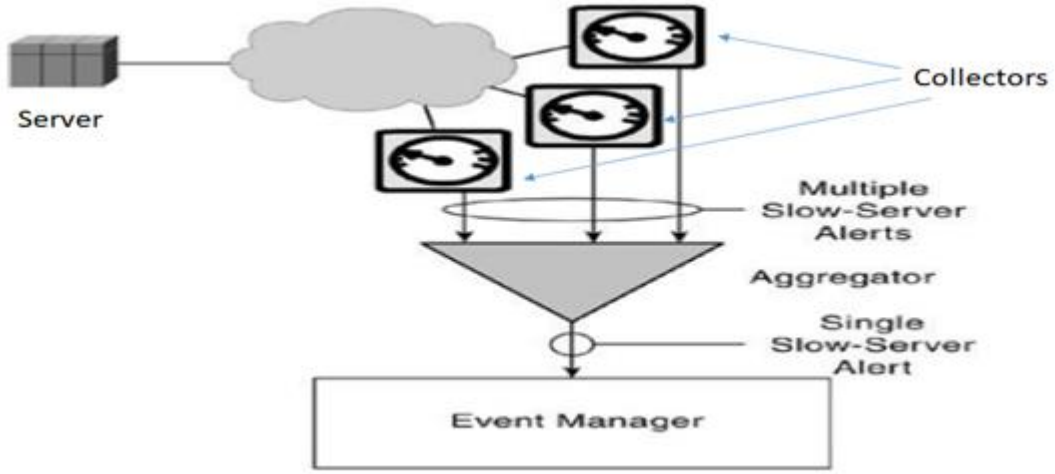


Figure 10 Roll-up to reduce alert volume

- **De-duplication:** A failure may generate alarms and events that can be consolidated as one by de-duplication. For example, failure of a device may indicate a large number of alarms about dropped connections but de-duplication consolidates a larger number of similar alarms to single event.
- **Intelligent Monitoring:** In this method, alert volumes are reduced by intelligently monitoring situations. For example, some service has dependencies i.e next process has to wait for output from previous process. If a process fails then monitoring those that follow do not yield any useful information until the failed process is repaired.
- **Artifact Reduction:** This is another technique in which raw alert volume is reduced by eliminating artifacts. Following approaches help to reduce the number of artifacts:
    - **Verification:** In this process, a quick verification of incoming alert is made to check whether this reporting an actual problem. This initial measurement is treated as an artefact if the verification does not reveal a problem.
    - **Filtering:** In this process, a set of rules are applied to a single alert source over some time interval. If measurements exceeds a specific response-time threshold then it is considered as artifact.
    - **Correlation:** In contrast to filtering, correlation deals with a number of independent alert sources simultaneously. It is more powerful than filtering because it analyses and identifies most likely cause of a flooding of alerts.

## 4.2 Correlation methods

According to (Salah, et al., 2013), correlation methods can be separated, depending on the strategy used to correlate the events, into similarity-based methods, sequential-based methods and case-based methods.

### 4.2.1 Similarity methods

Try to reduce the number of events by clustering and/or aggregating them based in their similarities. This can be done because it is assumed that similar alerts usually have the same cause and effects on the network and systems.

#### *4.2.1.1 Attribute based*

Correlate events using the similarities between some of their attributes. The attributes can be

IP, ports, protocols…

#### *4.2.1.2 Temporal based*

It is considered that alerts generated by the same cause are likely observed within a short time frame.

### 4.2.2 Sequential methods

Causal relationships between alerts are used to correlate them. In this methods, pre-conditions are defined as the necessary requirement for the attack to succeed, while consequences are the effects caused by a specific attack.

#### *4.2.2.1 Pre/Post conditions*

Tries to find the relationship between alerts based on pre and post conditions. The assumption is that previous alerts prepare for later ones.

#### *4.2.2.2 Graphs*

Collect the sequential information of alerts by mapping them into graphs. The nodes represent alerts and the edges connecting the nodes the relationship between those nodes.

#### *4.2.2.3 Codebook*

Encode the relationship between network faults and their symptoms by creating a matrix of problem codes that represent the dependency between the symptoms and the problems.

#### *4.2.2.4 Markov models*

Stochastic production model composed of discrete states and a matrix of state transition probabilities. It is assumed that the Markov property is followed, which establishes that the next state depends only on the current state and not on the sequence of events that precede it.

#### *4.2.2.5 Bayesian networks*

Are used to represent knowledge about an uncertain domain. Each node in the graph represents a random variable, being the edges the probabilistic dependencies between the nodes. Based on the symptoms, received as alerts, it is possible to calculate the probability that a specific problem has occurred.

### 4.2.3 Case-based methods

This methods rely on the existence of knowledge to represent well-defined scenarios, trying to correlate alerts based on known scenarios.

#### *4.2.3.1 Expert based*

Imitates the knowledge of a human, this methods do not scale well.

##### 4.2.3.1.1 Expert rules

The knowledge is developed as conditional rules (if – then), which are matched against the events as they come in. The match can be exact or partial. This type of correlation is well suited for systems that rarely vary, as it has a high cost of implementation and adaptation.

##### 4.2.3.1.2 Pre-defined scenarios

The knowledge, as in the expert rules method, is manually added. In this case, however, a specific language is used to implement the well-defined scenarios. If several alerts contribute to the construction of a predefined scenario, they are correlated.

#### *4.2.3.2 Inferred knowledge*

Can be built using inference methods with machine learning algorithms.

## 4.3 Comparison of alert correlation techniques

Alert correlation can be categorized into following categories (Sadoddin & Ghorbani, 2006):

- **Normalization**: Alerts may come from different probes in different formats. Normalization is a process of translating each alert into a standardized format that is easy to understand by correlation components.
- **Aggregation**: It is a process of grouping similar alerts in one category. Alerts are considered to be similar if they match in all attributes except few differences, or if they share the same root causes.
- **Correlation**: It is a process of finding relationships among alerts in order to reconstruct attack scenario from the isolated alerts. Thus, it gives high level view of actual attacks. Various correlation methods can be categorised into four types: scenario-based, rule-based, statistical and temporal.
- **Strategy Analysis**: In this process, reasons about attacker's actual intentions are identified from low-level correlated alerts.
- **Prioritization**: In this process, alerts are classified based on their severity and taking into account various domain information like security policy, network topology, asset profiles etc.

Table shows comparative analysis of various alert correlation methods.

| Category | Strategy | References |
|---|---|---|
| **Aggregation** | Bayesian networks (sequential-based) | Iyer et al (Iyer, et al., 2004). |
| **Aggregation** | Pre-defined scenarios (case-based) | GhasemiGol and Ghaemi-Bafghi (GhasemiGol & Ghaemi-Bafghi, 2015) |
| **Aggregation** | Inferred knowledge (case-based) | GhasemiGol and Ghaemi-Bafghi (GhasemiGol & Ghaemi-Bafghi, 2015) |
| **Aggregation** | Pre/post condition based correlation (sequential-based) | Ning and Cui (Ning & Cui, 2002) |
| **Aggregation** | Ruele-based(Case-based) | Debar and Wespi (Debar & Wespi, 2001) |
| **Aggregation** | Similarity based clustering (similarity-based) | Cuppens (Cuppens, 2001)<br>Valdes and Skinner (Valdes & Skinner, 2001) |
| **Aggregation** | Temporal based (similarity-based) | Valeur et al. (Valeur, et al., 2004) |
| **Correlation** | Bayesian networks (sequential-based) | Ning et al. (Ning, et al., 2004)<br>Iyer et al (Iyer, et al., 2004).<br>Qin (Qin, 2005)<br>Qin and Lee (Qin & Lee, 2004) |

| Category | Strategy | References |
|---|---|---|
| **Correlation** | Pre-defined scenarios (case-based) | GhasemiGol and Ghaemi-Bafghi (GhasemiGol & Ghaemi-Bafghi, 2015) |
| **Correlation** | Inferred knowledge (case-based) | GhasemiGol and Ghaemi-Bafghi (GhasemiGol & Ghaemi-Bafghi, 2015) |
| **Correlation** | Pre/post condition based correlation (sequential-based) | Cuppens and Miège (Cuppens & Miège, 2002)<br>Ning and Cui (Ning & Cui, 2002)<br>Ning et al. (Ning, et al., 2004)<br>Qin (Qin, 2005)<br>Shittu et al. (Shittu, et al., 2015) |
| **Correlation** | Ruele-based(Case-based) | Debar and Wespi (Debar & Wespi, 2001) |
| **Correlation** | Similarity based clustering (similarity-based) | Valdes and Skinner (Valdes & Skinner, 2001) |
| **Correlation** | Temporal based (similarity-based) | Valeur et al. (Valeur, et al., 2004)<br>Sadoddin and Ghorbani (Sadoddin & Ghorbani, 2009) |
| **Normalization** | Ruele-based(Case-based) | Debar and Wespi (Debar & Wespi, 2001) |
| **Strategy Analysis** | Pre/post condition based correlation (sequential-based) | Cuppens and Miège (Cuppens & Miège, 2002) |
| **Prioritization** | Pre/post condition based correlation (sequential-based) | Shittu et al. (Shittu, et al., 2015) |

**Table 1 Comparative analysis of alert correlation methods**

## 4.4 Correlation of Alerts vs. Correlation of Meta-alerts

Alerts, collected from same or different probes, are correlated to generate meta-alerts. After generating meta-alerts, feature matching probability of two or more meta-alerts determines if these meta-alerts can be correlated or not. If meta-alerts correlation will not be enabled, then each alert might be considered as discrete activity and possibly dismissed as insignificant and treated as false positive. Important ways of correlating meta-alerts are as follows (Spadaro, 2013):

- **Temporal-based correlation of meta-alerts**: In this case, relationships among meta-alerts are identified using time series analysis. A set of ordered and finite values to a variable of interest are taken for analysis on a time axis.
- **Probability-based correlation of meta-alerts**: Meta-alerts can be correlated by creating a static matrix from prior probability distribution with expectation of feature similarity. Here, it is expected that feature values matches if two meta-alerts are causally linked.
- **Prerequisite and consequence-based correlation of meta-alerts:** Another way of meta-alert correlation is by matching the known prerequisites and consequences of attacks. A relationship is established by matching consequence of a meta-alert with the prerequisite of another meta-alert by taking into account the temporal characteristics like: attributes, identification etc. State descriptive language (like LAMBDA (Cuppens & Ortalo, 2000)) can be

used to specify events by: (i) pre-condition and post-condition of an attack, (ii) events which are performed by the attacker, (iii) events which allow the detection of the attack and (iv) conditions proving the attack succeeded.

# 5 Additional Insights into Meta-alerts processing

## 5.1 Sources of Alerts in PROTECTIVE

In the following table, there are the details of some of the sources that send information into PROTECTIVE. Apart from the listed in the table, there are other sources which can be found in D6.6.

| Source | URL | Type | Detection | Description |
|---|---|---|---|---|
| **Flowmon ADS** | https://www.flowmon.com/cs/products/flowmon/anomaly-detection-system | NetFlow analyzer | spam, scan, dns, rdp/ssh bruteforce, dos, malware, sniffer, anomalies | Flowmon ADS evaluates statistics on network traffic and detects specific behavioural patterns and anomalies |
| **FTAS** | https://www.cesnet.cz/sluzby/sledovani-provozu-site/ | NetFlow analyzer | large scans, DDoS attacks, brute forcing | FTAS evaluates network traffic statistics and detects attacks and network anomalies. |
| **Nemea** | https://www.liberouter.org/technologies/nemea/ | IPFIX Data Analyzer | DNS, SIP, DDoS, scan, low-threshold password searches | Nemea evaluates the network traffic statistics expanded by the application layer data and detects attacks and anomalies in the network |
| **Kippo/Cowrie** | http://www.micheloosterhof.com/cowrie/ | SSH bruteforce honeypot | bruteforce attempts, successful logins, saves downloaded files, and record entire session | MySQL, hpfeeds, text log |
| **Dionaea** | http://dionaea.carnivore.it/ | Honeypot (smb, http(s), ftp, tftp, mssql, sip) | Successful attempts to save downloaded files | sqlite, xmpp, python module, text log, hpfeeds |
| **Beekeeper** | - | Aggregator and data classifier from VoIP honeypots | - | Beekeeper evaluates data sent from Beekeepersender (also known as Bkprnode), which is obtained from VoIP honeypots or by parsing tcpdump above Asterisk. Bkprnode / Bkprsender sends this data to the Beekeeper in CSV format on the REST interface. In the future, this will be ensured by the MQTT protocol |
| **LaBrea** | http://labrea.sourceforge.net/ | Tarpit (captures tcp connection and prevents it from closing) | Information about connection attempts and captured connections | LaBrea takes over unused IP addresses, and creates virtual servers that are attractive to worms, hackers, and other denizens of the Internet. The program answers connection attempts in such a way that the machine at the other end gets "stuck", sometimes for a very long time. |
| **Honeyd** | http://www.honeyd.org/ | Honeypot (series of scripts for different protocols) | - | Scan a specific service (recognizes that an attacker is talking through a specific protocol) |

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.

| Source | URL | Type | Detection | Description |
|---|---|---|---|---|
| **HP TippingPoint** | http://www8.hp.com/us/en/software-solutions/ips-intrusion-prevention-system/index.html | Active IPS / firewall (security in the box) | Scan, spam, vuln | IPS/IDS system |
| **Snort** | https://www.snort.org/ | Traffic Analyzer based on patterns | Scan, dos, login, compromise, trojan, trojan, various anomalies | IPS/IDS system |
| **Fail2Ban** | http://www.fail2ban.org/ | Regexp engine to create ip sheets and run actions based on sample logon identification | A lot of information can be vaporized by a suitable regex, such as bruteforce, spam mta | - |
| **IntelMQ** | https://github.com/certtools/intelmq | Complex engine for taking and processing of different types of data from predominantly free and publicly available sources | Practically anything, often information from blacklists or public lists | - |
| **Fortigate connector** | - | PROTECTIVE connector | Logs from firewall | Fortigate connector creates a bridge between the events stored in the UTM (Fortigate) equipment and the Protective message exchange system, Warden. Thus, we create a flow of security events caught up by Fortigate UTM and send them to other exchange partners by transforming them into a common format. The connector also takes care of duplicate events |

## 5.2 Log Formats and Storage

All the data generated in the PROTECTIVE project has to be stored on disk. The possible ways to do so, however, are different and depend on the uses for the data and the processes it will be subject to.

### 5.2.1 Text-based log files

Is the format that is most common, as there are a lot of applications that generate this type of logs and it has multiple benefits.

- The generation of logs is inexpensive in terms of CPU and I/O.
- Human readable format, can be processed by common tools.

#### *5.2.1.1 Flat text files*

Flat schema-less file that can follow a common pattern or be free form. It is one of the more common formats.

#### *5.2.1.2 Indexed flat text files*

One of the limitations of the previous format is the ability to query/sort/retrieve elements quickly. Indexed flat text files provide a way to organize the data, maintaining the human readable format and providing further capabilities such as give results to queries more quickly.

### 5.2.2 Binary files

Machine readable log file that require special tools to read and process, for example Microsoft Windows Event Logs. This format makes an efficient use of disk space, although it does not compress well.

### 5.2.3 Compressed files

To preserve space, log files can either be compressed directly or after a certain amount of time, thus increasing the space available for the storage of logs.

### 5.2.4 Database Storage of logs

The formats specified previously provide fast access to logs, but are not so good neither for generating reports nor for filtering or correlating the data.

The use of databases allows the generation of queries to search and retrieve data quickly. In addition, there are many tools, both applications and programming languages, to interact with databases therefore cutting down costs for the development of custom made tools.

## 5.3 Search Engines

To query and extract meaningful information from all the data stored in the PROTECTIVE system, some search capabilities have to be added to the project.

The search capabilities can be provided by a full text search engine, either a tool developed for this specific purpose or by the built-in capabilities of a database.

Full text search engines are capable of quickly searching high volumes of unstructured text, provide rich text search capabilities and sophisticated relevance ranking tools.

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.

This type of engines are able to quickly categorize data based on specific values of concrete fields. Moreover, the relevancy ranking capabilities, used for determining the best match for a query, include using the frequency of query terms in the document, the proximity of query terms near each other in the document and special weightings for particular terms, among others.

Full text search systems depend, generally, on some type of index in order to perform the queries. In the cases where an index is needed, either an index exists for each one of the relevant fields or all fields are contained in a single index. Moreover, this type of systems offer the capability to store and retrieve the data in its original form.

To sum up, full text search engines are useful when dealing with:

- High volume of free form text data
- Support of interactive text-based queries.
- The need to offer relevant search results.
- Flexibility in generating the queries.

Some of the specific built tools for this purpose are Lucene, Solr, Sphinx and ElasticSearch.

# 6 Development of Correlation Mechanisms

During the first months we studied several implementations of a Complex Event Processor (CEP), which have been extensively used in the industry to detect correlated events (Ficco, 2013). In the following table, we can see the score obtained by the different tools in regard to the aspects that have been evaluated:

- Licensing.
- Date of last release.
- Quality of the documentation.
- Integration with other tools.
- Ease of use.
- Future uses.

| Criteria | Nodebrain | Simple Event Correlator | Drools | Esper | WSO2 DAS |
|---|---|---|---|---|---|
| Licensing | MIT License | GNU General Public License | Apache Software License 2.0 | GPL v2 | Apache Software License 2.0 |
| Date of last release | sep-15 | feb-17 | ago-17 | jun-17 | 2017 |
| Quality of documentation | 3 | 3 | 4 | - | 4 |
| Ease of use | 2 | 1 | 3 | - | 3 |
| Integration with other tools | 1 | 1 | 2 | - | 3 |
| Future uses | 0 | 0 | 2 | - | 4 |

Figure 11 Comparison of CEP engines

## 6.1 Nodebrain

Rule engine, written in C, for construction of state and event monitoring applications, its last release was in September 2015. Can be downloaded from http://nodebrain.org/index.html

### Basics

- Declarative rule language: order of actions and evaluation does not matter.
- Agents: NodeBrain script to monitor elements or streams.
- Servants: Program/Script (any language), that runs as a child of NodeBrain, are able to communicate with the parent using stdin, stdout and stderr and can generate dynamically, update rules or implement external actions.
- Modules: Plugin which provides added capabilities using a C API.
- C API

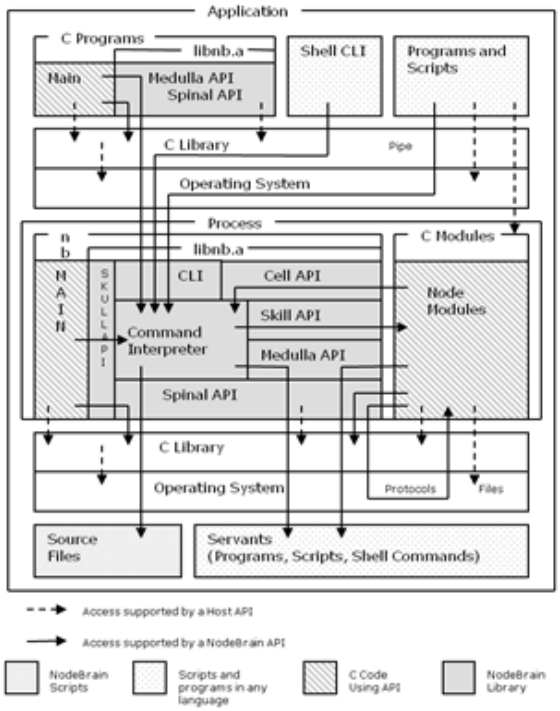


Figure 12 Nodebrain schema

NodeBrain can be run in different ways:

- Command line.
- Scripts.
- Daemon.
- System startup.

In addition, nodebrain has its own rule language and modules to extend its functionality, making it able to send mails, snmp traps and syslog messages and communicate through pipes. However, it is complex to function.

## 6.2 Simple Event Correlator

Lightweight event correlation tool, written in perl, for advanced event processing, its last release was in June 2016.Its homepage is https://simple-evcorr.github.io/

#### Basics

- Single threaded, multiple processes can run at the same time.
- Performance depends on the number of rules and their order.
  - Rules arranged into sequences (rulesets), each one in a separate text file.
- Well documented, with many examples.
- Feature full rule language:
  - Join rules to create event correlation schemes.
  - Use of Perl functions inside rules for pattern matching, parsing and as additional filters.

    - Use of named match variables and match caching.
- Can be run in different ways:
    - Used interactively with shell pipelines.
    - Executed as daemon(s)
    - Connected to other applications over FIFOs, pipes, network sockets …
- Origin of input events:
    - Regular files.
    - FIFOs.
    - Standard input.

## 6.3 Drools

Drools is a Business Rules Management System (BRMS) solution, developed in Java. It has a community release from Jboss.org without support and can be obtained from http://www.drools.org/

#### Basics

- The component that provides complex rule processing, which is "Drools Fusion", is well documented and has a rich rule language.
- Drools Fusion:
    - Supports streams of events.
    - Allows detection, correlation, aggregation and composition of events.
    - Supports temporal constraints in order to model the temporal relationships between events.
    - Sliding time window - allows to write rules that only match events occurring in the last X time units.
    - Supports adapters for event input into the engine.

## 6.4 Esper

Esper is licensed under the GPL v2 license. As a consequence, any development done atop of it is exposed. Redistribution of the software requires commercial licensing. Therefore, this option has not been evaluated further. The homepage is http://www.espertech.com/products/esper.php

## 6.5 WSO2 Data Analytics Server

WSO2 Data Analytics Server (DAS) is a robust open source analytics platform, its homepage is https://docs.wso2.com/display/DAS310/WSO2+Data+Analytics+Server+Documentation

It is used for realtime streaming data analysis, offering also CEP facilities as well as machine learning methodologies. It is able to analyze a constant stream of events that represents the transactions and activities from different sources, process them in realtime and report on its interfaces. WSO2 DAS combines realtime analytics with batch jobs, as well as interactive and predictive (thanks to machine learning facilities) analysis of data into an integrated solution.

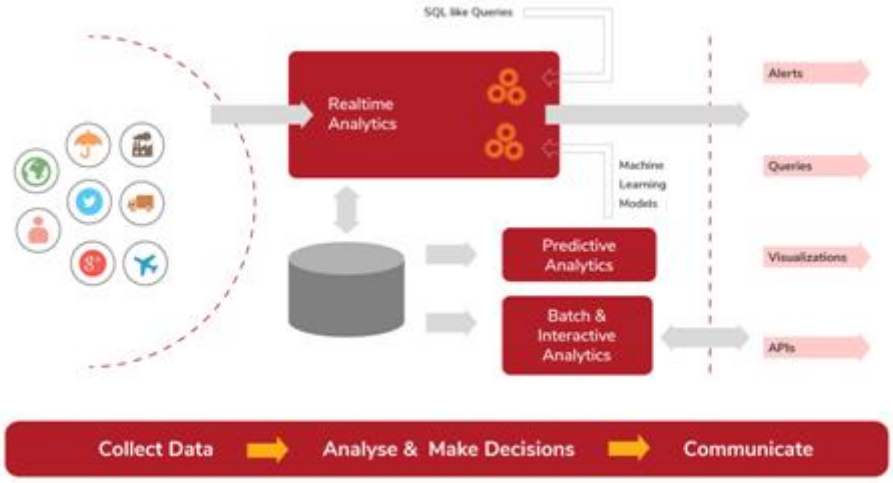


Figure 13 WSO2 DAS Architecture[1]

WSO2 DAS workflow consists of three main phases briefly described below. Each of the phases conforms to the particular stage of the PROTECTIVE pipeline and will be supporting it:

- **Collecting Data** – WSO2 DAS provides a single API for multiple sources (including other sources in the DAS instance through Event Receivers) and is able to analyze streaming and persistent data.
- **Analysing Data**
  - In real time, using Siddhi Query Language for the definition of relevant stream properties
  - As batch or interactive jobs. The batch analytics is based on Apache Spark.
  - As predictive analysis through integration with WSO2 ML module.
- **Communicating Results** – through several different presentation interfaces, e.g. customizable Analytics Dashboard.

GMV, the PROTECTIVE consortium partner, has got significant experience in using WSO2 DAS. PSNC has also used Siddhi and WSO2-based engine in one of their national projects SECOR[2].

[1] https://docs.wso2.com/display/DAS310/Introducing+DAS

[2] Frankowski, G., Jerzak, M., Miłostan, M., Nowak, T., & Pawłowski, M. (2015). Application of the Complex Event Processing system for anomaly detection and network monitoring. Computer Science Journal , 16 (4), pp. 351-372

# 7 Summary

As seen through the document, extensive research has been performed by the partners in order to decide some key components in the PROTECTIVE project. Both the alert schema and the pipeline have been defined. We have also inspected the methods to enrich alerts.

Moreover, we have explained and compared different correlation methods, which has allowed us to gain useful insights to decide what we want to implement in the project, in addition to investigating the different formats for log storage and the advantages and uses of a search engine.

To finish with, we have tried several correlation engines. During this process, we have taken into account several points:

- Licensing.
- Date of last release.
- Quality of the documentation.
- Integration with other tools.
- Ease of use.
- Future uses.

Based on the results of our experimentation, we have arrived at the conclusion that the most suited, at the moment, for the tasks we want to achieve is WSO2 DAS.

# 8 References

Alshammari, R., Sonamthiang, S. & Teimouri, M., 2007. Using Neuro-Fuzzy Approach to Reduce False Positive Alerts. In: *Communication Networks and Services Research.* s.l.:IEEE, pp. 345-349.

Aziz, N., 2007. *Intrusion Alert Correlation.* [Online]
Available at: http://www.slideshare.net/amiable_indian/intrusion-alert-correlation

CNSS, 2015. *Committee on National Security Systems (CNSS) Glossary.* [Online]
Available at: https://www.cnss.gov/CNSS/openDoc.cfm?0SK1qPsaRpQtdsXBZsIxLQ==

Cuppens, F., 2001. Managing alerts in a multi-intrusion detection environment. In: *Proceedings of the 17th ACM annual computer security applications conference.* New York: ACM.

Cuppens, F. & Miège, A., 2002. Alert correlation in a cooperative intrusion detection framework. In: *Proceedings of IEEE symposium on security and privacy.* New York: IEEE, pp. 202-215.

Cuppens, F. & Ortalo, R., 2000. LAMBDA: A Language to Model a Database for Detection of Attacks. In: *RAID 2000: Recent Advances in Intrusion Detection.* s.l.:s.n., pp. 197-216.

Debar, H. & Wespi, A., 2001. Aggregation and correlation of intrusion-detection alerts. In: *Proceedings of the international symposium of the recent advances in intrusion detection (RAID'01).* Berlin: Springer, pp. 85-103.

Ficco, M., 2013. Security event correlation approach for cloud computing. *International Journal of High Performance Computing and Networking,* 7(3), pp. 173-185.

GhasemiGol, M. & Ghaemi-Bafghi, A., 2015. E-correlator: an entropy-based alert correlation system. *Security and Communication Networks,* 8(5), pp. 822-836.

Grill, M., Pevný, T. & Rehak, M., 2017. Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. *Journal of Computer and System Sciences,* 83(1), pp. 43-57.

Hewlett-Packard, n.d. *What is Event Correlation?.* [Online]
Available at: http://www8.hp.com/us/en/software-solutions/what-is/event-correlation.html

Howard, J. D. & Longstaff, T. A., 1998. *A Common Language for Computer Security Incidents,* Albuquerque: Sandia National Laboratories.

Hubballi, N. & Suryanarayanan, V., 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications,* Volume 49, pp. 1-17.

IEEE, 1996. *The IEEE Standard Dictionary of Electrical and Electronics Terms,* New York: Institute of Electrical and Electronics Engineers, Inc..

Iyer, P., Reeves, D. & al, e., 2004. Reasoning about complementary intrusion evidence. In: *Proceedings of the 20th ACM annual computer security applications conference.* New York: ACM, pp. 39-48.

Kruegel, C., Valeur, F. & Vigna, G., 2005. Intrusion Detection and Correlation: Challenges and Solutions. In: s.l.:Springer, p. 31.

Kruegel, C., Valeur, F. & Vigna, G., 2005. Intrusion Detection and Correlation: Challenges and Solutions. In: s.l.:Springer, p. 44.

McConnell, J. & Siegel, E., 2004. *Practical Service Level Management: Delivering High Quality Web-based Services.* s.l.:Cisco Press.

Ning, P. & Cui, Y., 2002. Constructing attack scenarios through correlation of intrusion alerts. In: *Proceedings of the 9th ACM conference on computer and communications security.* s.l.:s.n., pp. 245-254.

Ning, P., Xu, D., Healey, C. & Amant, R., 2004. Building attack scenarios through integration of complementary alert correlation methods. In: *Proceedings of the 11th annual network and distributed system security symposium (NDSS'04).* s.l.:s.n., pp. 97-111.

NIST, 2013. *Glossary of Key Information Security Terms,* s.l.: National Institute of Standards and Technology.

NIST, 2016. *Guide to Cyber Threat Information Sharing.* [Online]
Available at: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf

Pouget, F. & Dacier, M., 2003. *Alert correlation: Review of the state of the art.* s.l.:s.n.

Qin, X., 2005. *A probabilistic-based framework for INFOSEC alert correlation,* s.l.: Ph.D. dissertation.

Qin, X. & Lee, W., 2004. Attack plan recognition and prediction using causal networks. In: *Proceedings of the 20th ACM annual computer security applications conference.* New York: ACM, pp. 370-379.

Sadoddin, R. & Ghorbani, A., 2006. Alert correlation survey: framework and techniques. In: *Proceedings of the ACM international conference on privacy, security and trust.* New York: ACM, pp. 1-10.

Sadoddin, R. & Ghorbani, A., 2009. An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security,* 28(3-4), pp. 153-173.

Salah, S., Maciá-Fernández, G. & Díaz-Verdejo, J. E., 2013. A model-based survey of alert correlation techniques. *Computer Networks,* 57(5), pp. 1289-1317.

Shittu, R. et al., 2015. Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security,* Volume 50, pp. 1-15.

Spadaro, A., 2013. *Event correlation for detecting advanced multi-stage cyber-attacks.* [Online]
Available at: https://repository.tudelft.nl/islandora/object/uuid:d7d43988-f1b0-4549-8b44-3b5c9f1f401b?collection=education

Subrahmanian, V., 2013. *Handbook of computational approaches to counterterrorism.* s.l.:Springer.

Technopedia, n.d. *Data Filtering.* [Online]
Available at: https://www.techopedia.com/definition/26202/data-filtering

The Free Dictionary, n.d. *The Free Dictionary.* [Online]
Available at: www.thefreedictionary.com/preference

Valdes, A. & Skinner, K., 2001. Probabilistic alert correlation. In: *Proceedings of the international symposium of the recent advances in intrusion detection (RAID'01).* Berlin: Springer, pp. 55-68.

Valeur, F., Vigna, G., Kruegel, C. & Kemmerer, R., 2004. A comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing,* 1(3), pp. 46-169.

Yu Beng, L., Ramadass, S., Manickman, S. & Soo Fun, T., 2014. A Survey of Intrusion Alert Correlation and Its Design Considerations. *IETE Technical Review,* 31(3), pp. 233-240.

# Annexes

## Annex A: Examples of Alerts

The examples are intended to show the diversity of alerts from different sources. Attributes have been marked with **bold font**. IP addresses involved have been at least partially anonymized in certain cases.

### Example 1. A high level alert

This alert has been obtained from EGI as an email notification. It is very easy to be understood by a human operator but difficult to be automatically processed.

```
FROM: <you>
TO: <site-security-contacts@mailman.egi.eu/abuse@egi.eu>
SUBJECT: Security incident suspected at <site> [EGI-<Date in Format:YYYYMMDD>] TLP:AMBER
** AMBER Information - Limited Distribution                    **
** This may be shared with trusted security teams on a need-to-know basis **
** see https://wiki.egi.eu/wiki/EGI_CSIRT:TLP for distribution restrictions **

Dear security contacts,

A security incident has been detected at <StieName>.

- Short summary of the incident
<Provide a high-level overview of the incident>

- Host(s) affected
<List of compromised hosts and/or hosts running suspicious user code.
ex: grid-worker-node-124.mysite.org (123.123.123.123)>

- Host(s) used as a local entry point to the site (ex: UI or WMS IP address)
<The host that the attacker is likely to have used to access the site.
ex: grid-ui-101.mysite.org (123.123.123.124)>

- Remote IP address(es) of the attacker
<The remote host from where the attacker is likely to have connected from.
ex: 123.adsl.somecorp.com (012.012.012.012)>

- Evidence of the compromise, including timestamps (ex: suspicious files
or log entry) <Ex: the attacker logged in has root from 123.adsl.somecorp.com.
Times are UTC:
Mar 24 12:00:09 grid-ui-101 sshd[13896]: Accepted password for root
from 012.012.012.012>

- What was lost, details of the attack
<Provide available details on the extent of the compromise. Ex:
System logs revealed the attacker guessed the root password of
grid-ui-101 on Mar 24 12:00:09
(UTC) after hundreds of attempts. Then, the attacker [...] etc.>

- If available and relevant, the list of other sites possibly affected
<Ex: firewall logs reveal suspicious SSH connections from the compromised node to grid-
ui.friendlysite.org on Mar 24 13:01:03 (UTC). friendlysite.org has been contacted.>

- Possible vulnerabilities exploited by the attacker
<Ex: the attacker exploited a weak root password and gained further access by exploiting
CVE-2009-
1234 against [...] etc.>

- Actions taken to resolve the incident
<Ex: Disk images have been saved, hosts have been reinstalled from scratch with new, strong
root
passwords, and SSH has been configured to prevent "root" logins with password.>

- Recommendations for other sites, actions suggested
<Ex: Sites should check and report any successful SSH connection from grid-ui-101 between
Mar 24
12:00:09 (UTC) and Mar 24 17:00:00 (UTC).
```

```
It is also recommended to avoid direct SSH access, and to configure sshd with
"PermitRootLogin
without-password".>

- Timeline of the incident
<Ex:
2009-03-24 09:12:43 UTC Multiple SSH connection attempts from 12.012.012.012
2009-03-24 12:00:09 UTC Attacker connects as root on grid-ui-101.mysite.org from
012.012.012.012
2009-03-24 13:01:03 UTC SSH scan from grid-ui-101 against grid-ui.friendlysite.org
[...]
2009-03-24 15:00:00 UTC Site security team investigating
2009-03-24 15:34:00 UTC EGI security contacts informed [...]>
```

## Example 2. A low level alert

This alert has been obtained from the n6 incident exchange platform. This is a very low level alert, easy to be processed but hard to be understood by a human.

```
"time","id","source","category","name","md5","ip","url","fqdn","asn","cc","details"
"2015-11-19T13:50:47Z","0ea737a2b1688881c13604cdf3ec06d7","hidden.7","bots","b106-
multi","","150.254.XXX.YYY","","","9112","PL","from port 49170 to 204.95.99.109:7799"
```

## Example 3. IDEA alert

This alert is described in the IDEA (Intrusion Detection Extensible Alert) format, developed by CESNET and selected to be used in the PROTECTIVE system for threat information sharing. More information about the properties of the IDEA format in the context of the PROTECTIVE requirements may be found in the PROTECTIVE deliverable D6.1. The PROTECTIVE consortium believe that this format combines easiness of automated processing with relatively good opportunities of understanding by a human operator, when relevant. The alert itself is significantly larger than the n6-based one but also provides much more meaningful information.

```
{
  "Format": "IDEA0",
  "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
  "CreateTime": "2012-11-03T10:00:02Z",
  "DetectTime": "2012-11-03T10:00:07Z",
  "WinStartTime": "2012-11-03T05:00:00Z",
  "WinEndTime": "2012-11-03T10:00:00Z",
  "EventTime": "2012-11-03T07:36:00Z",
  "CeaseTime": "2012-11-03T09:55:22Z",
  "Category": ["Fraud.Phishing"],
  "Ref": ["cve:CVE-1234-5678"],
  "Confidence": 1,
  "Note": "Synthetic example",
  "ConnCount": 20,
  "Source": [
      {
      "Type": ["Phishing"],
      "IP4": ["192.168.0.2-192.168.0.5", "192.168.0.10/25"],
      "IP6": ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
      "Hostname": ["example.com"],
      "URL": ["http://example.com/cgi-bin/killemall"],
      "Proto": ["tcp", "http"],
      "AttachHand": ["att1"],
      "Netname": ["ripe:IANA-CBLK-RESERVED1"]
      }
  ],
  "Target": [
      {
      "Type": ["Backscatter", "OriginSpam"],
      "Email": ["innocent@example.com"],
      "Spoofed": true
```

```
        },
        {
        "IP4": ["10.2.2.0/24"],
        "Anonymised": true
        }
  ],
  "Attach": [
        {
        "Handle": "att1",
        "FileName": ["killemall"],
        "Type": ["Malware"],
        "ContentType": "application/octet-stream",
        "Hash": ["sha1:0c4a38c3569f0cc632e74f4c"],
        "Size": 46,
        "Ref": ["Trojan-Spy:W32/FinSpy.A"],
        "ContentEncoding": "base64",
        "Content": "TVpqdXN0a2lkZGluZwo="
        }
  ],
  "Node": [
        {
        "Name": "cz.cesnet.kippo-honey",
        "Type": ["Protocol", "Honeypot"],
        "SW": ["Kippo"],
        "AggrWin": "00:05:00"
        }
  ]
}
```

# Annex B: Definitions of Main Concepts Related with Meta-alerts

The table below provides a set of definitions related with the concept of meta-alerts (besides alert and meta-alert themselves that have been described thoroughly above). A more extended set of definitions may be found in an intermediary WP3 document entitled "WP3 terminology dictionary".

| No. | Term | Explanation and notes | Source |
|---|---|---|---|
| **1** | attribute | A property of an alert, element that helps to describe that alert. | Own |
| **2** | correlation | **Event correlation** refers to the processes involved in sensing and analyzing relationships between events. Event correlation plays a vital role in automatically reducing the noise and allowing IT to focus on those issues that really matter to the business service and IT objectives.<br>NOTE: the term will rather be used referring to **alert correlation**. Alerts may be understood as being the consequences of events (security systems, as a reaction on certain events, will raise alerts). | (Hewlett-Packard, n.d.) |
| **3** | event | Any observable occurrence in a system and/or network. Events sometimes provide an indication that an incident is occurring (NIST, 2013) .<br>Another definition (Howard & Longstaff, 1998) states that event is an action directed at a target which is intended to result in a change of state (status) of the target.<br>NOTE: the latter definition would have to be considered if it matches all types of security attacks. For instance, does the unauthorized read access to the target change its status? | (NIST, 2013), (CNSS, 2015), (Howard & Longstaff, 1998), (IEEE, 1996) |
| **4** | filtering | Data filtering in IT can refer to a wide range of strategies or solutions for refining data sets. This means the data sets are refined into simply what a user (or set of users) needs, without including other data that can be repetitive, irrelevant or even sensitive. Different types of data filters can be used to amend reports, query results, or other kinds of information results. | (Technopedia) |
| **5** | incident | An assessed occurrence that actually or potentially jeopardizes the confidentiality, integrity, or availability of an information system; or the information the system processes, stores, or transmits; or that<br>constitutes a violation or imminent threat of violation of security policies, security procedures, or acceptable use policies. | (NIST, 2013), (CNSS, 2015) |
| **6** | observable | An event (benign or malicious) on a network or system. | (NIST, 2016) |
| **7** | operator | A person (usually) or a system that is involved in the process of handling alerts or meta-alerts concerning targets in the infrastructure that is under the responsibility of that person, performing relevant actions. | Own |
| **8** | preference | The selecting of someone or something over another or others.<br>NOTE: The term will be used in PROTECTIVE concerning alerts and/or meta-alerts; the system operator will prefer to handle one alert/meta-alert more promptly than the other one. | (The Free Dictionary) |

| | | | |
|---|---|---|---|
| **9** | prioritization | The process of ordering items in order of their relative importance due to the specified criteria.<br>NOTE: Prioritization will be performed referring to the list of meta-alerts and possibly alerts. The system operator will be able to indicate a preference on the two meta-alerts $M_1$, $M_2$. An indication of preference allows pointing following four opportunities: $M_1$ has precedence over $M_2$ (should be handled before $M_1$), $M_2$ has precedence over $M_1$, $M_1$& $M_2$ are equally important, precedence is not possibly to point (meta-alerts are incomparable). The operator will not need necessarily to generate the full ranking of all meta-alerts $M_1$, … $M_n$. The prioritization process will rely on a set of criteria like e.g. criticality of the endangered asset, risk level, trust score, organization security policy, operator preferences, etc. | Own |
| **10** | property | Equivalent to attribute. | Own |
| **11** | source | (**attack source**) A computer or logical network entity (account, process, or data) or physical entity (component, computer, network or internetwork) that is performing an attack against the specified target(s). | Own, based on (Howard & Longstaff, 1998) |
| **13** | value | State of a particular attribute (property) of an alert or meta-alert that may be used to describe or quantify the alert or meta-alert. | Own |

# Annex C: Example of alerts correlation

*Table 2 Example of alert correlation,* shows an example for alert correlation. In this example, monitored network has four heterogeneous IDS sensors, i.e., network based IDS 1 and 2, (N1, N2), host based and application based IDS (H, A). Following are the actions performed by an attacker (Subrahmanian, 2013):

1. An attacker (31.3.3.7) launches a port scan against 10.0.0.1 and discovers the vulnerability of Apache server.
2. After scanning, the attacker performs a successful Apache buffer overflow exploit on the target and obtains user privilege on the server.
3. Finally, the attacker launches privilege escalation by using a local exploit linux.conf.

In an alert correlation process, Alerts #2 and #3 are grouped as malicious scanning, Alerts #4 and #5 as vulnerability attempts, Alerts #6 and #7 as privilege escalation, and the noisy Alert #1 will be marked as irrelevant. Once alerts are aggregated, alert groups (malicious scanning, vulnerability attempts and privilege escalation) are reported to security analysts. Here, it is easy to notice that once correlated, it will be easy and effective for an analyst to process the high-level attack descriptions instead of individual alerts.

| Alert ID | Description | Sensor | Start/end time | Source | Target |
|---|---|---|---|---|---|
| 1 | IIS exploit | N1 | 12.0/12.0 | 80.0.0.1 | 10.0.0.1, port: 80 |
| 2 | Scanning | N2 | 10.1/14.8 | 31.3.3.7 | 10.0.0.1 |
| 3 | Port scan | N1 | 10.0/15.0 | 31.3.3.7 | 10.0.0.1 |
| 4 | Apache exploit | N1 | 22.0/22.0 | 31.3.3.7 | 10.0.0.1, port: 80 |
| 5 | Bad request | A | 22.0/22.1 | | localhost, Apache |
| 6 | Local exploit | H | 24.6/24.6 | | linuxconf |
| 7 | Local exploit | H | 24.7/24.7 | | linuxconf |

Table 2 Example of alert correlation (Valeur, et al., 2004)