Proactive Risk Management through Improved Cyber Situational Awareness

Start Date of Project: 2016-09-01

Duration: 36 months

D6.1 Framework Specification

Deliverable Details	
Deliverable Number	**D6.1**
Revision Number	**E**
Author(s)	**PSNC, GMV, CESNET**
Due Date	**30/04/2017**
Delivered Date	**28/06/2017**
Reviewed by	**AIT, TUDA, UOXF**
Dissemination Level	**PU**
Contact Person EC	**Georgios Kaiafas / Alina-Maria Bercea**

Contributing Partners	
1.	**PSNC – Author**
2.	**GMV – Author**
3.	**CESNET – Contributor**
4.	**AIT – Reviewer**
5.	**TUDA – Reviewer**
6.	**UOXF – Reviewer**

Revision History

Revision	By	Date	Changes
A1	PSNC, GMV	26/04/2017	Document draft with Executive Summary, Information Exchange Formats, PROTECTIVE base candidates
A2	PSNC	27/04/2017	Drafted Architecture chapter
A3	PSNC, GMV	27/04/2017	Reorganized document structure (moved CESNET toolset into the architecture chapter), added Introduction, Implementation Framework Procedures, Tools and Technology scouting chapters drafts
A4	PSNC, GMV	02/05/2017	Adding content to Implementation Framework Procedures chapter, adding Analysis and Comparative part, extending architecture part with integration and framework base part, reorganizing Mentat/Warden descriptions
A5	PSNC, GMV	05/05/2017	Added some technology scouting and architecture content, reorganized the document
A6	PSNC, CESNET	12/05/2017	Finalized technology scouting chapter, updated CESNET toolset sections, updated architecture diagrams and functional models description, provided content on security monitoring of the developed solutions and implementation framework
A7	PSNC	13/05/2017	Finalized Implementation Framework Procedure chapter
A8	GMV	16/05/2017	Adding and reviewing content to Analysis and Comparative chapter with CESNET comments
A9	PSNC	17/05/2017	Finalized (except Runtime chapter) Implementation Framework Tools chapter
A10	PSNC, CESNET	24/05/2017	Polished from comments, remarks and prepared for the final shaping. Complemented abbreviation list, table and figure names, etc. Reorganized structure (new chapter 5, merged so-far chapters 6 and 7). Complemented some technology scouting (now moved to annex). Applied set of comments from the reviewers.
A11	PSNC	29/05/2017	Described Docker, OpenStack, relevance of MISP and Spot for PROTECTIVE. Extended Warden and Mentat selection criteria. Added lists of tables, figures, listings.
A12	PSNC	30/05/2017	Finalized rebuilding the frameworks and technologies selection chapter (now Chapter 4 + Annex C). Addressed and cleared out the majority of comments.
A13	PSNC, GMV, CESNET, AIT	31/05/2017	Extended PROTECTIVE suite roadmap. Addressed the majority of reviewers' comments (esp. to Analysis and Comparative chapter).
A14	PSNC, CESNET	01/06/2017	Improved Architecture description, Warden description and Chapter 4 structure. Provided some general final polishing, moved generic links to footnotes, removed typos.
A15	PSNC	02/06/2017	Finalized Introduction and Integration part, partially added technologies contextualization, updated Executive

Revision	By	Date	Changes
			Summary accordingly to the final structure, made some editorial changes.
A16	PSNC	03/06/2017	Mapped technical selection criteria to PROTECTIVE objectives, finalized technology contextualization, restructured finally (and complemented) technology scouting.
A17	PSNC, CESNET, TUDA	09/06/2017	Finalized Warden and Mentat selection justification. Addressed reviewers comments.
A18	PSNC, ITTI, TUDA, UOXF	20/06/2017	Addressed reviewers comments (including restructuring the document), added descriptions of supporting tools (Cybertool, CertainTrust), made some editorial improvements and unifications.
A19	AIT	22/06/2017	Rearranged the chapter structure place PROTECTIVE framework overview at beginning of document. Reworked Chap 4 to remove cut and paste text in parts. Streamlined the chapter.
A20	PSNC	25/06/2017	Extended IDEA justification section. Removed most of comments, made some improvements from contributors, made general final polishing.
A21	PSNC, CESNET, GMV, UOXF	26/06/2017	Added WSO2 DAS Description. Finalized IDEA justification section. Improved headers and footers. Applied UOXF final review.
E	PSNC	27/06/2017	Final version.

Abbreviation List

AGPL Affero General Public License
AMQP Asynchronous Message Queueing Protocol
AS Autonomous System
ASVS Application Security Verification Standard
CDSA Cyber Defence Situational Awareness
CEP Complex Event Processing
CERT Computer Emergency Response team
CI Continuus Integration
CO Component Owner
CP Candidate Point
CSA Cyber Situational Awareness
CSIRT Computer Security Incident Response Team
CT Cyber Tool
CTI Cyber Threat Intelligence
CVE Common Vulnerabilities and Exposures
C4ISR Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
DX.Y Deliverable number X.Y
DAS Data Analytic Server
DBMS DataBase Management System
DoS Denial of Service
DDoS Distributed Denial-of-Service
DNS Domain Name Service
DoA Description of Action
DoS Denial-of-Service
DSN Delivery Status Notification
DXL Data eXchange Layer
EB Executive Board
ENISA European Network and Information Security Agency
FACT Federated Analysis of Cyber Threats
HTTP HyperText Transfer Protocol
ICMP Internet Control Message Protocol
ICT Information and Communication Technology
IDEA Intrusion Detection Extensible Alert
IDEA Intrusion Detection Extensible Alert
IDMEF Intrusion Detection Message Exchange Format
IHAP Incident Handling Automation Project
IOC Indicator of Compromise
IODEF Incident Object Description Exchange Format
JSON JavaScript Object Notation
LIDS Local Intrusion Detection System
MCDA Multiple Criteria Decision Aid
MCDM Multiple Criteria Decision Making
MIME Multipurpose Internet Mail Extensions
MISP Malware Information Sharing Protocol

ML	Machine Learning
MSSP	Managed Security Systems Provider
MOA	Massive Online Analysis
MTA	Mail Transfer Agent
NIDS	Network Intrusion Detection System
NREN	National Research and Education Network
ODM	Open Data Models
OWASP	Open Web Application Security Project
OWL	Web Ontology Language
PO	Pilot Objective
PO	Product Owner
RBL	Realtime Blackhole List
REST	Representational State Transfer
SCADA	Supervisory Control And Data Acquisition
SDLC	Security Development Life Cycle
SDO	STIX Domain Object
SIEM	Security Information Management System
SIEM	Security Information Management System
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service-Oriented Architecture
SOC	Security Operations Center
SRO	STIX Relationship Object
SM	Scrum Master
STIX	Structured Threat Information Expression
TAXII	Thrusted Automated eXchange of Indicator Information
TI	Threat Intelligence
TO	Technical Objective
TTP	Tactics, Techniques, and Procedures (or: Tools, Techniques and Procedures)
WP	Work Package
WSO2	Web Services Oxygen(O2)
X-ARF	Extended Abuse Reporting Format
XML	eXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol

Executive Summary

This report presents the design of the PROTECTIVE framework and describes the tools and technologies selected to form the base of this framework. It also specifies internal information exchange formats and any special frameworks services that will be needed. We identify and analyze existing state-of-the-art technologies, solutions, tools, libraries, software frameworks in order to identify those that will serve as the basis of the PROTECTIVE security information processing pipeline. This will include security related toolsets and technologies as well as more general software tools and technologies. It will also evaluate and identify software management and development practices and tools that can be utilized to carry out the developmental and software quality of goals the project.

The described task is defined within the PROTECTIVE Description of Action (DoA) as T6.1 and takes time M1 through M8 of the project activity period. One of its results is the following document. In the following chapters, we present results of the efforts undertaken over the past 8 months. The document is divided into a set of chapters which allow the reader to get generally acquainted with concepts crucial to the project (more detailed descriptions of these may be found in relevant deliverables provided by WP2, 3, 4 and 5), and thoroughly describe proposed solution overview, procedures for developers to follow in order to create that solution and finally a report of which state-of-the-art solution were evaluated, their relevance to the project and possible reusability assessment. The structure of the document is as follows:

- **Chapter 1 (Introduction)** briefly presents the purpose of the deliverable as well as its general structure and interrelations with other PROTECTIVE documents (deliverables and DoA).
- **Chapter 2 (PROTECTIVE Framework Design)** sets the context for the document by introducing the information processing workflow concept and relating it to the PROTECTIVE tool.
- **Chapter 3 (Information Exchange Formats)** mentions the most influential data formats in CSIRT domain and involves the analysis of different internal information data exchange formats, mainly IDEA and STIX, which will be used throughout the PROTECTIVE framework.
- **Chapter 4 (Solutions Analyzed as Potential PROTECTIVE Main Components)** contains the comprehensive report on the evaluated available software and technologies that have been investigated.
- **Chapter 5 (Technological Baseline Selection for the PROTECTIVE System)** summarizes the selection process of main technological solutions. Selection criteria as well as comparative analysis are provided that led to make the selection. A further roadmap has also been sketched on how the supplementation and extension of the base technologies are envisaged by the PROTECTIVE consortium. Finally, the most relevant scouted technologies and solutions have been described; either necessary backend elements (e.g. databases) or tools required in particular stages of the PROTECTIVE pipeline (e.g. scientific tools for prioritization supported with the MCDA approach).
- **Chapter 6 (Implementation Framework Procedures and Tools)** presents the proposed development cycle and tools that will support bespoke software development. An important aspect of the workflow is assuring the required security level of the provided software.

To summarize – this document specifies the design of the PROTECTIVE Framework and describes the architecture, data exchange formats, tools and technologies analyzed and finally selected to form the base of the PROTECTIVE framework.

The Contents

List of Figures

List of Tables

List of Listings

1 Introduction

As stated in the DoA: "*PROTECTIVE will develop a comprehensive solution to raise organizational cyber situational awareness (CSA) through:*

- *enhancement of security alert correlation and prioritization,*
- *linking of the relevance/criticality of an organization's assets to its business/mission,*
- *establishment of a threat intelligence sharing community,*

These three elements will be tightly woven to provide an integrated CSA platform that will be developed firstly for CSIRTs in the NREN community and later applied for further validation within the SME end-user community."

This report specifies the design of the PROTECTIVE information processing pipeline framework and describes the tools and technologies selected to form the base of the framework as the result of task T6.1. It also specifies chosen information exchange formats and any special frameworks services that are deemed necessary.

This task is closely aligned with the architecture designed in WP2 and related WP3, WP4 & WP5. The report is to a significant extent a result of a bidirectional cooperation of project teams from WP6 and the aforementioned architectural and analytic work packages. D6.1 is intended to emphasize on the investigation of and reason for selected technologies and solutions in PROTECTIVE. This is why architecture, scenarios or PROTECTIVE processing pipeline, are only outlined at a high level. More in-depth descriptions may be found in the dedicated PROTECTIVE deliverables, referred to in text.

Data exchange format design that will be used throughout the framework is a shared responsibility with WP5 and here we only briefly report results of the WP5 work documented in D5.1. For instance, this deliverable reviews some of the existing formats and their adaptation for PROTECTIVE.

Identification and selection of suitable backend engines that will support successful implementation of core platform is a crucial task for WP6. In T6.1 backend storages, search engines, visualization tools, and workflow and event processing engines available on the market were carefully examined to identify the most promising ones. Selection of existing security tools (e.g. IntelMQ[1], Warden[2], etc.) as a base for the framework was also part of this task. This report results in selecting appropriate main solutions, covering multiple elements of the planned PROTECTIVE pipeline (especially Warden and Mentat[3]) as well as additional solutions leading to complement the base systems in the optimal recognized manner. For instance, events correlation and prioritization (ranking) tools for the multi-criteria decision aiding processor, required by WP3 are also mentioned in this report.

Finally, agile development and testing support strategies/approaches and tools are also described as part of the Integration and Testing of WP6. Methodology and tools that guarantee the maximum security level of the PROTECTIVE suite have also been identified and described.

2 PROTECTIVE Framework Design

In order to appropriately address the challenge of building the framework, we refer to the Technological Objective No. 5 of PROTECTIVE:

TO5 – To develop a software framework to support information flow processing. …There are ***many existing toolsets that can be taken as a starting point*** *for such a pipeline. However* (Kijewski, 2012)

[1] https://github.com/certtools/intelmq
[2] https://warden.cesnet.cz/
[3] https://mentat.cesnet.cz/

reviewed a number of tools used by CSIRTs and concluded that none of them provided the necessary support and that "the early warning of network security incidents is still a long way away".

PROTECTIVE will therefore construct a processing pipeline framework from a combination of these existing toolsets and bespoke components to be developed within the project. This will use an existing software framework as a base in order to not reinvent the wheel. On the other hand, the identified gaps will be filled with additional software components which will also improve existing analytic opportunities. Another facet is to have the necessary awareness about the most recognized existing security awareness frameworks (or single solutions supporting such frameworks) in order to be compatible with them at least in terms of information exchange capabilities.

In this chapter, we introduce the ENISA pipeline concept[4] then show how PROTECTIVE relates to it. This chapter defines the context for the discussion in the remainder of the document. Information Processing Pipeline.

2.1 Security Information Processing Pipeline

PROTECTIVE is first and foremost a security information processing pipeline designed according the principles of the processing pipeline model described by ENISA (ENISA, 2014) and described in D2.1 Chapter 4. The processing pipeline described in (ENISA, 2014) follows a pipe and filter pattern. Information enters at the left-hand side and is gradually transformed in a number of processing steps as it flows through the pipeline. Alerts are combined with other information to give an aggregated or more abstract view of TI. Certain information will be discarded (e.g. false positives or alerts resulting from legitimate security assessment) and some others will be persisted for longer or shorter periods. The configuration of the pipeline is configurable and information flows may be split/joined and routed for separate processing (e.g. according to information type) as shown in Figure 1 .

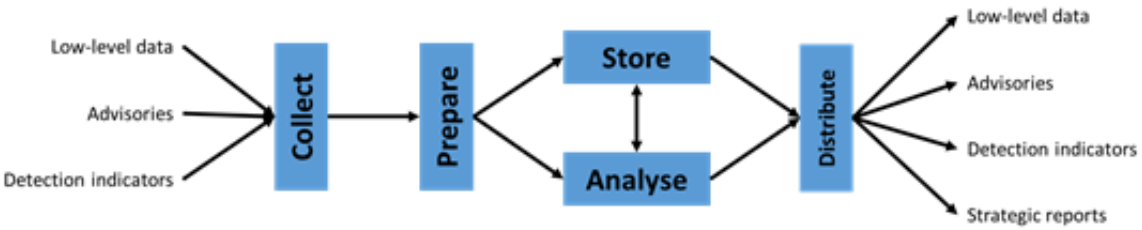


Figure 1. ENISA security Information Processing Pipeline

The stages in the ENISA generic pipeline are:

- **Collect** – in this stage different types of information from both internal and external sources is ingested. Information may be of different granularity, from multiple sources and intended for different purposes and audiences.
- **Prepare** – once the data has been collected, it is transformed to make it more useful (i.e. more actionable) from the recipient's point of view. This requires the development of parsing and normalisation processes for each distinct input format i.e. define how each is mapped into an internal data structure. Information may also be aggregated by combining multiple events into a single event that represents some activity as a whole. Events can also be enriched by adding additional context to existing information, thus increasing completeness. From the technical perspective, enrichment 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.

[4] The text in this section is derived from D2.1 Chapter 4.1

- **Storage** – the requirements for storage vary according to the information type, the volume of information ingested and the range of application types that process the information.
- **Analysis** – the analysis step is a process that takes collected and prepared information as the input and produces new conclusions. In contrast to enrichment, analysis is about deriving new information beyond that context that is explicitly linked to the original data. Analysis is a very wide-ranging activity and depends on the particular problem to be solved and it is almost impossible to list all of the methods and tools that are used for analysis.
- **Distribute** – the final step in the pipeline corresponds to the application and dissemination of actionable information that has passed through the previous stages. The importance of distribution stems from the simple fact that in order to ensure that appropriate mitigation actions are taken, a CERT needs to notify its constituents, who can then act appropriately based on the information in notifications. Disseminating the correct information in a timely fashion is frequently a non-trivial task requiring an investment of time in understanding those constituents' needs.

The information types that form the input and outputs of this pipeline, as defined in (ENISA, 2014), are outlined below:

- **Low-level data** – collected from multiple monitoring systems collecting data related to various activities occurring within an organisation. These activities include network traffic, actions performed by users, behaviour of applications, and many others.
- **Detection indicator** – is a pattern that can be matched against low-level data in order to detect threats. A crucial part of an indicator is the contextual information included with the indicator. The quality of the contextual information is critical – ideally, it should allow an analyst to clearly understand the threat the indicator is meant to detect. Indicators are machine processable.
- **Advisories** – includes several sorts of information that cannot be directly translated into a process for preventing or detecting threats, but which still provides information for analysts that might trigger a defensive action or help shape the nature of those actions. Advisories are usually intended for direct human consumption.
- **Strategic report** – information can also come in the form of highly summarised reports that aim to provide an overview of particular situations.

2.2 PROTECTIVE Ecosystem

The PROTECTIVE security alert and TI ecosystem for NREN's[5] is shown below in Figure 2. This depicts collaboration between a number of NRENS and between each NREN and its constituency . The placement of the PROTECTIVE tool within this overall ecosystem is also shown

[5] The tool is intended for use within any CSIRT but is described in an NREN context as this in the primary use within the project.

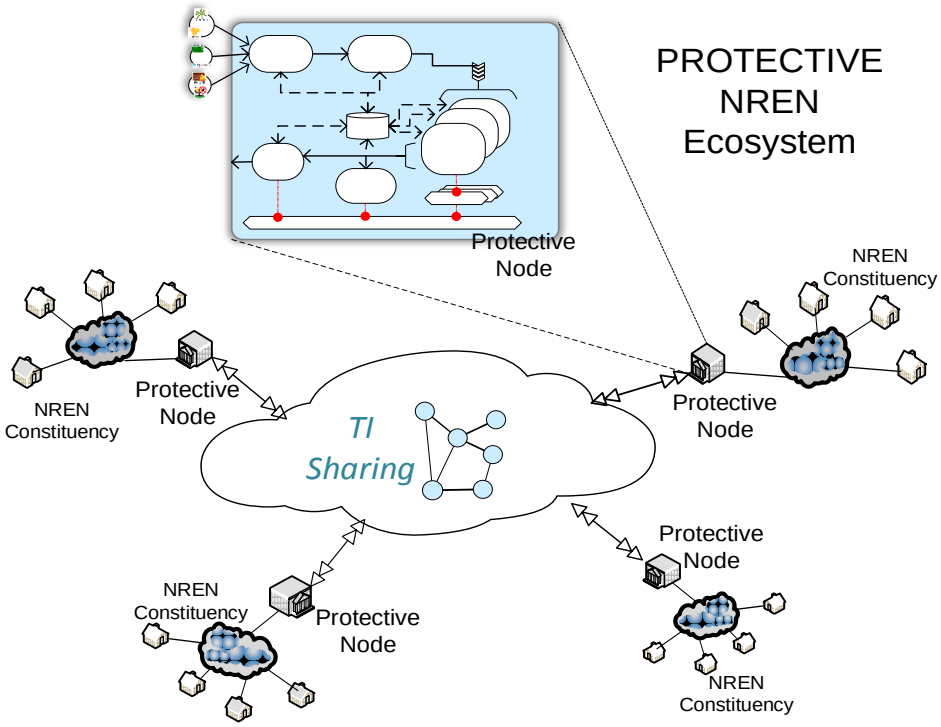


Figure 2. PROTECTIVE system in NREN environment

While this diagram shows the broader picture, it is necessary to decompose the PROTECTIVE too structure in order to get a better understanding of its information processing pipeline characteristics. We do this by giving a brief description of its components parts over the rest of this chapter.

The PROTECTIVE node internal structure is shown in Figure 3 below. The information processing pipeline relationship can be immediately seen. Security events from within the center at the top left side and progress through a series of processing steps. Alerts may be stored within the system and/or some may be shared with other NRENs. Each of the main functional blocks/subsystems in the node description may be further decomposed as described in D2.1. We do not fully reproduce that decomposition here but, in the next section give overview of the main such PROTECTIVE functional modules to serve as a base for the technology analysis which follows in subsequent chapters.

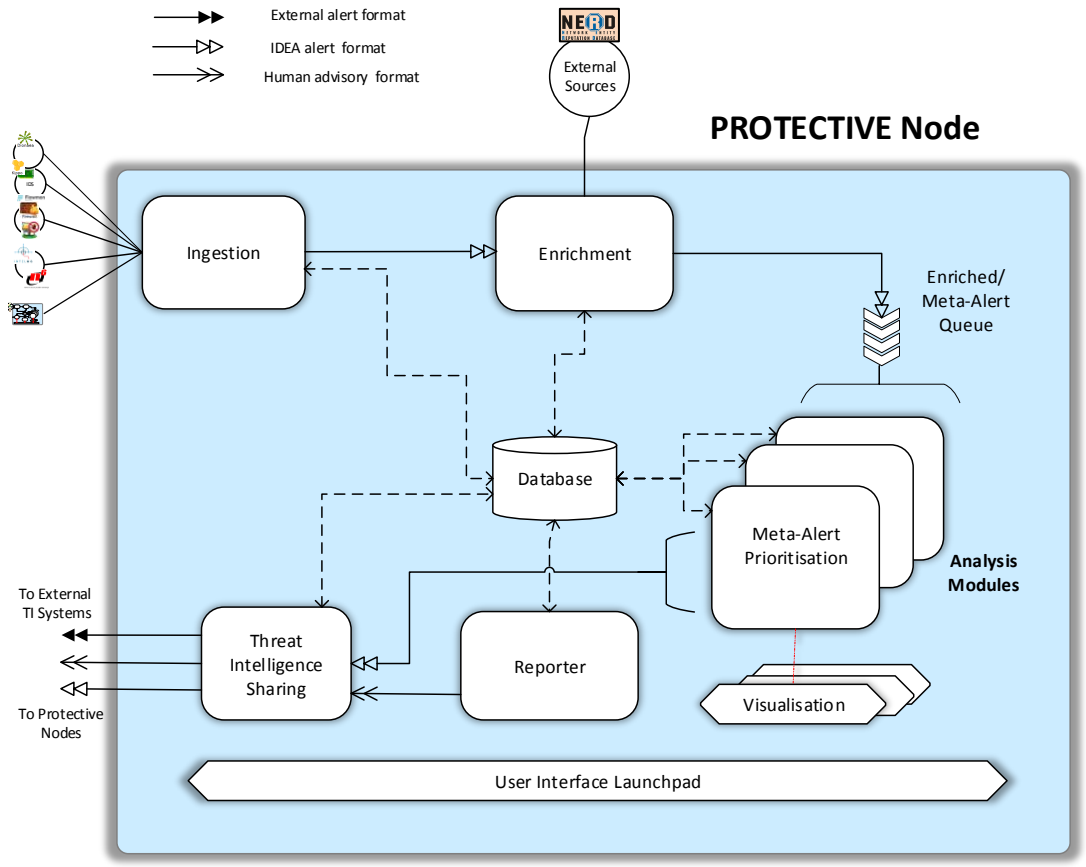


Figure 3. PROTECTIVE Node Internal Structure

2.3 PROTECTIVE Functional Modules

As mentioned, the detailed functionalities of the final system are discussed within deliverable D2.1 Within this section, the focus is put on general functional modules included in the tool. These include:

- Ingestion mechanisms including: data receivers, data normalization, queuing system, aggregation mechanisms, real-time (on-line) correlation
- Data storage facilities: to store raw and process data including TI, as well as configuration of the system
- Enrichment mechanisms including: acquisition of additional (context related) data from external sources, evaluation of acquired data quality (acquired either from internal and external entities), assessment of sharing entity reputation (c.f. D5.1), data related to asset state (e.g. patch levels, c.f. D4.1) and asset criticality (depicting impact of possible failure of compromise on business processes, see D4.1 for details)
- Analysis modules including: advanced correlation engines and data mining, configuration of message flow (message routing) and processing pipelines
- TI sharing mechanism (c.f. D5.1) including: data distribution layer, community building, access control mechanisms, compliance checking (e.g. GDPR compliance)
- Prioritisation and ranking building (namely meta-alert prioritization, to be defined in details within D3.2)
- Reporting capabilities

Data ingestion or acquisition mechanisms are tightly coupled with enrichment capabilities and include processes, probes, and collectors needed to collect a wide range of TI data, namely, alerts or

events, and a versatile contextual information, e.g. WHOIS[6] data, Realtime Blackhole Lists (RBLs), vulnerability-related data, business value of the resources, trust scores related to data sources. Data acquisition is present in various steps of the overall processing workflow within the PROTECTIVE system. It is crucial for gathering basic alerts on premise as well as for enriching local data feeds using external data sources. Enrichment adds more context to the locally observed actions of malicious adversaries, thus it is an immanent step in building and rising situational awareness. Definition of internal processing flows of the data, e.g. message routing through internal modules is part of the system configuration capabilities.

A community building mechanism is needed to facilitate efficient and secure sharing information within trusted societies[7]. This functional module is responsible for the provisioning of mechanisms allowing implementation of various sharing scenarios.

TI sharing mechanism or generally data sharing mechanisms are closely related to community building, although the emphasis is put here onto actual protocols and logistics of data distribution. Anonymization engines, filtering processes (including for example legal compliance checking), reporting and alerting are parts of this module. Moreover, access control mechanisms are needed to protect stored and shared data. It is also needed to make the system compliant with the data protection legislations.

Correlation engines enable reduction of data volume necessary by combining related pieces of information into single objects and also to identify particular campaigns or attacks. Advanced analytics is necessary to discover less obvious patterns within data acquired and stored in the system and also to identify new TI. Data storage facilities are crucial for effective operational capabilities and analytics possibilities of the whole system.

Prioritisation and ranking building aims in improving decision-making capabilities of the operator reacting on the spotted and reported threats. The alerts related to the most harmful threats, most harmful from a business perspective with the inclusion of preferences of the decision maker, should be addressed as the first ones. The more detailed view of the functional modules taking into account usage scenarios found in D2.1 is depicted in Figure 4.

[6] https://tools.ietf.org/html/rfc3912
[7] Described in-depth in D5.1

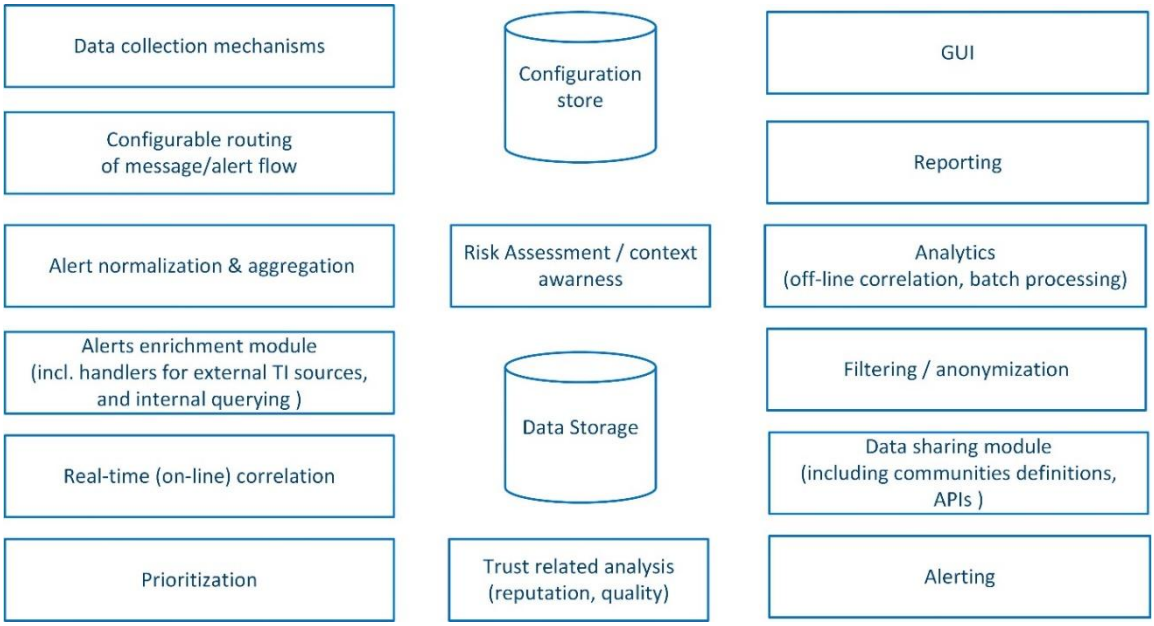


Figure 4. Functional modules of PROTECTIVE system

2.4 Modules Integration

It is assumed that during the implementation of the PROTECTIVE suite a set of existing open source components will be used and integrated together in order to avoid duplicating existing tools but concentrate on obtaining the optimal results from them and filling identified gaps with new functionalities.

The PROTECTIVE development teams responsible for particular modules are distributed across Europe. Combining this fact with the most probable situation that the components themselves will be, possibly, distributed among independent computing systems and communicating over IP networks, it is necessary to provide an elastic and robust way to deploy them and integrate either in distributed and single host environments.

As such, we will be striving to design particular components in a generally accepted, coherent way. They will be aimed to realize the assumed functionality by processing data fed with, and forwarding the results to the next components in the pipeline for further processing. Having sketched the architecture, the processing domains of PROTECTIVE will be defined – the components building them, their dependencies and communication APIs.

Thus, instead of proceeding with a monolithic architecture, it is intended that PROTECTIVE will be designed and developed using the post-SOA paradigms – namely microservices architecture[8]. This will allow to maximize the finely-grained modules decoupling, to keep the implementation independent and to only require for the consortium to agree on efficient communications APIs – be them blocking Asynchronous Message Queueing Protocol (AMQP) or reactive streams. The details will be further discussed in the upcoming chapters where we describe the technologies, frameworks and databases supporting PROTECTIVE processing development.

[8] http://microservices.io/

3 Information Exchange Formats

This chapter examines a number of information exchange formats that could be used to transfer TI within the PROTECTIVE ecosystem. The chapter has to consider formats that may be used for information transfer internally within the PROTECTIVE tool and between PROTECTIVE nodes in the overall TI sharing ecosystem.

The format for security event exchange is nothing new, and efforts exist that define languages or formats to allow for exchange, however none of them have become universally accepted. In the last few years, a number of projects for automated incident report exchange appeared, namely Warden[9], AbuseHelper[10], IntelMQ[11], n6[12], Megatron[13], CIF[14] and Prelude[15].

For the exchange of cybersecurity information (FIRST, 1995 - 2017) to occur as messages between any two entities, **it must be structured and described in some consistent manner that is understood by both parties**. This chapter describes specifications that enable this exchange. The goal is to make it easier to share cybersecurity information that often includes "common enumerations", that is, ordered lists of well-established information values for the same data type. Common enumeration allows distributed databases and other capabilities to be linked 0t0ogether, and to facilitate the cybersecurity related comparisons.

The following sections describe two main analyzed information exchange formats:

- Structured Threat Information Expression (STIX)
- Intrusion Detection Extensible Alert (IDEA)

Other, less intensively considered formats that have been investigated, also have a short mention. Finally, **the proposed information exchange format for PROTECTIVE platform is IDEA**.

3.1 STIX

3.1.1 Description

STIX is a language and serialization format used to exchange Cyber Threat Intelligence (TI) (OASIS-CTI-TC, 2017). STIX enables organizations to share TI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to observe and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve capabilities such as collaborative threat analysis, automated threat exchange, automated detection and response, and more (OASIS-CTI-TC, 2017).

A set of specifications currently provides the normative description of STIX 2.0. While STIX 2.0 is defined to be independent of any specific serialization, JSON is the mandatory-to-implement serialization and informative JSON schemas are available. The Listing 1 shows a JSON-based example of a STIX 2.0 Campaign object:

[9] https://warden.cesnet.cz/
[10] http://abusehelper.be/
[11] https://github.com/certtools/intelmq
[12] http://n6.cert.pl/
[13] https://gitorious.org/megatron
[14] https://code.google.com/p/collective-intelligence-framework/
[15] https://www.prelude-ids.org/

```
{
    "type": "campaign",
    "id": "campaign--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "created": "2016-04-06T20:03:00.000Z",
    "name": "Green Group Attacks Against Finance",
    "description": "Campaign against targets in the financial services sector."
}
```

Listing 1. STIX 2.0 Campaign object

In addition to describing a set of STIX Domain Objects (SDOs), STIX 2.0 enables relationships to be defined between objects. As the example shows, a Campaign object may be attributed to a Threat Actor and may target a Vulnerability while an Indicator indicates the Campaign.

A companion TI specification, TAXII (OASIS-CTI-TC, TAXII, 2017), is designed specifically to transport STIX Objects. However, the structures and serializations of STIX do not rely on any specific transport mechanism, and STIX provides a bundle, a container for STIX Objects to allow for transportation of bulk STIX data, especially over non-TAXII communication mechanisms.

Figure 5 shows an example of how STIX objects can relate to each other. In the example we see how several indicators link to both threat actors and a grouping of adversarial, malicious behaviors over a period of time against a specific set of targets (campaign) that targets vulnerabilities attributed to aforementioned threat actors.

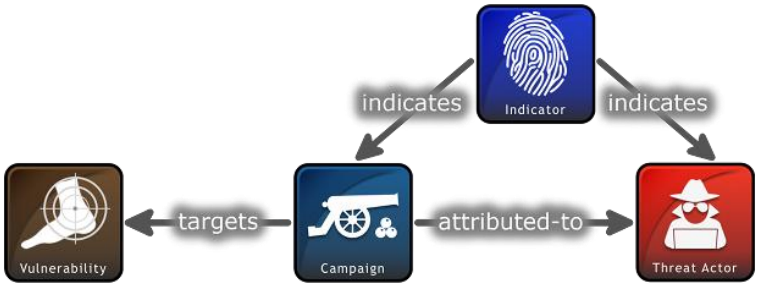


Figure 5. STIX 2.0 Relationship Example

3.1.2 Architecture

The STIX architecture (see Figure 6 below) identifies the core cyber threat concepts as independent and reusable constructs and characterizes all of their interrelationships based on the inherent meaning and content of each. Connecting arrows between construct icons indicate relationships in the form of content elements within the construct at the root of the connecting arrow, that is of the conceptual type of the construct at the head of the connecting arrow and is suggested but not required to utilize the specific STIX implementation of that construct. The bracketed asterisk on each of the arrow labels implies that each relationship may exist zero to many times. The structured content of each construct is fleshed out in detail within the language implementation (currently in the form of an XML Schema). The eight core constructs – Observable, Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign and ThreatActor – along with a cross-cutting Data Markings construct are briefly characterized below.

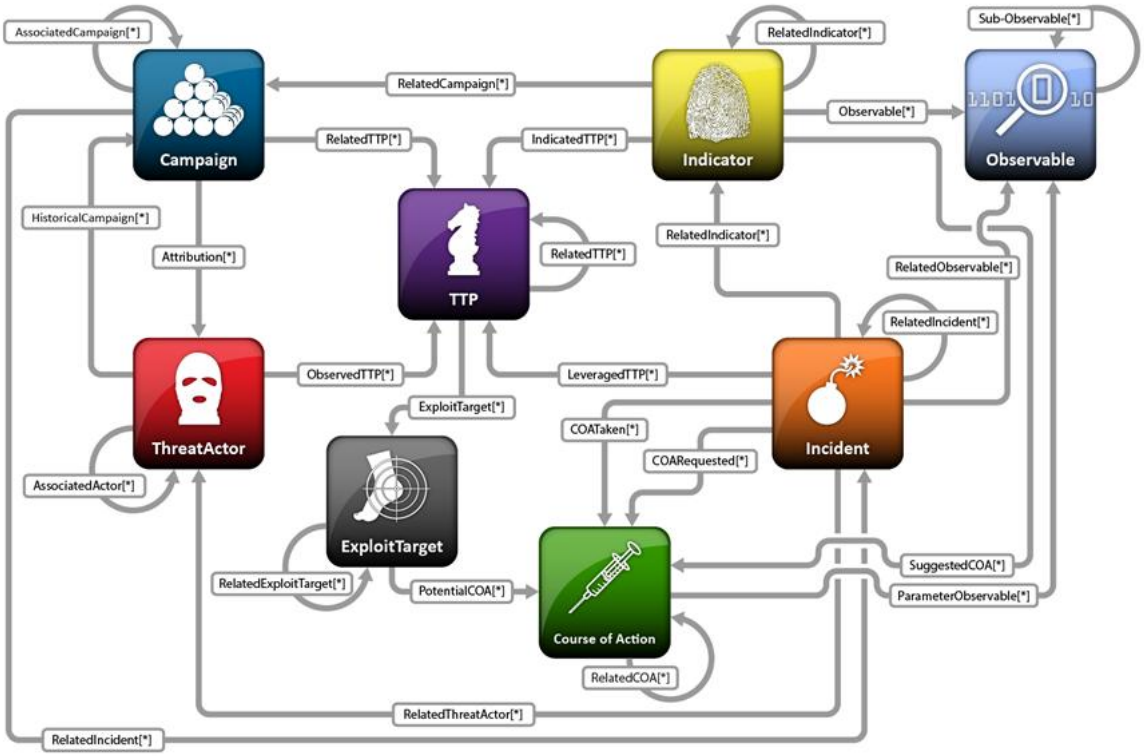
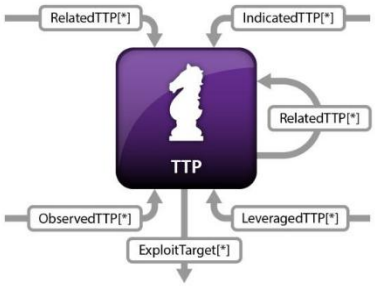


Figure 6. STIX 2.0 Architecture[16]

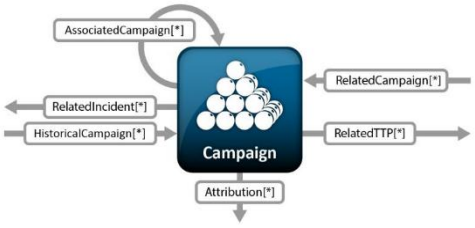
The nine SDOs included in STIX 2.0 (listed below) represent a minimally viable product that fulfils basic consumer and producer requirements for CTI sharing. Objects and properties not included in STIX 2.0, but deemed necessary by the community, will be included in future releases of STIX. The description below is provided, following the STIX website[17].



Tactics, Techniques, and Procedures (TTP)

TTPs are representations of the behavior or modus operandi of cyber adversaries. It is a term taken from the traditional military sphere and is used to characterize what an adversary does and how they do it in increasing levels of detail.

TTPs consist of the specific adversary behavior (attack patterns, malware, exploits) exhibited, resources leveraged (tools, infrastructure, personas), information on the victims targeted (who, what or where), relevant ExploitTargets being targeted, intended effects, relevant kill chain phases, handling guidance, source of the TTP information, etc.

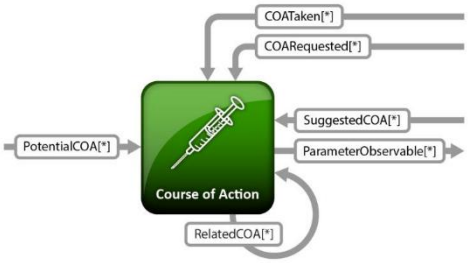


Campaign

Campaigns are instances of ThreatActors pursuing an intent, as observed through sets of Incidents and/or TTP, potentially across organizations. In a structured sense, Campaigns may consist of the suspected intended effect of the adversary, the related TTP leveraged within the Campaign, the related Incidents

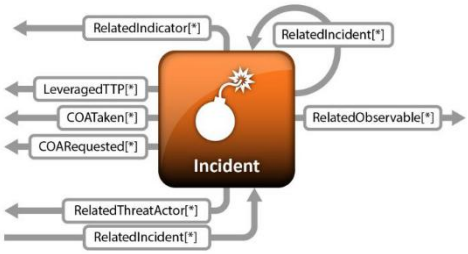
[16] http://stixproject.github.io/getting-started/whitepaper/#architecture
[17] https://oasis-open.github.io/cti-documentation/stix/about.html

believed to be part of the Campaign, attribution to the ThreatActors believed responsible for the Campaign, other Campaigns believed related to the Campaign, confidence in the assertion of aggregated intent and characterization of the Campaign, activity taken in response to the Campaign, source of the Campaign information, handling guidance, etc.



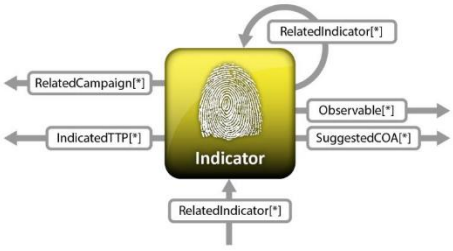
Course of Action

CoursesOfAction are specific measures to be taken to address threat whether they are corrective or preventative to address ExploitTargets, or responsive to counter or mitigate the potential impacts of Incidents. In a structured sense, COA consist of their relevant stage in cyber threat management (e.g., remedy of an ExploitTarget or response to an Incident), type of COA, description of COA, objective of the COA, structured representation of the COA (e.g., IPS rule or automated patch/remediation), the likely impact of the COA, the likely cost of the COA, the estimated efficacy of the COA, observable parameters for the COA, handling guidance, etc.



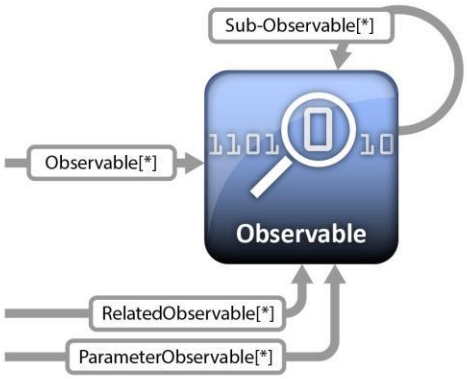
Incidents

Incidents are discrete instances of Indicators affecting an organization along with information discovered or decided during an incident response investigation. They consist of data such as time-related information, parties involved, assets affected, impact assessment, related Indicators, related Observables, leveraged TTP, attributed Threat Actors, intended effects, nature of compromise, response Course of Action requested, response Course of Action taken, confidence in characterization, handling guidance, source of the Incident information, log of actions taken, etc.



Indicator

Indicators convey specific Observable patterns combined with contextual information intended to represent artifacts and/or behaviors of interest within a cyber security context. They consist of one or more Observable patterns potentially mapped to a related TTP context and adorned with other relevant metadata on things like confidence in the indicator's assertion, handling restrictions, valid time windows, likely impact, sightings of the indicator, structured test mechanisms for detection, related campaigns, suggested courses of action, related indicators, the source of the Indicator, etc.



Observed Data

Observables are the "base" construct within the STIX architecture. Observables are stateful properties or measurable events pertinent to the operation of computers and networks. Information about a file (name, hash, size, etc.), a registry key value, a service being started, or an HTTP request being sent are all simple examples of observables. STIX leverages CybOX for its representation of Observables..

Data Markings

A consistent requirement across many of the core STIX constructs is the ability to represent markings of the data to specify things like handling restrictions given the potentially sensitive nature of many forms of cyber threat information. This construct is not conveyed as a separate entity in the STIX architecture diagram but exists as a cross-cutting structure within all of the top-level constructs described above.

Threat Actor

ThreatActors are characterizations of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behavior. In a structured sense, ThreatActors consist of a characterization of identity, suspected motivation, suspected intended effect, historically observed TTP used by the ThreatActor, historical Campaigns believed associated with the ThreatActor, other ThreatActors believed associated with the ThreatActor, handling guidance, confidence in the asserted characterization of the ThreatActor, source of the ThreatActor information, etc.

Exploit Targets

ExploitTargets are vulnerabilities or weaknesses in software, systems, networks or configurations that are targeted for exploitation by the TTP of a ThreatActor. In a structured sense, ExploitTargets consist of vulnerability identifications or characterizations, weakness identifications or characterizations, configuration identifications or characterizations, potential Courses of Action, source of the ExploitTarget information, handling guidance, etc.

A comparison of STIX 2.0 to 1.x can also be seen in the "Annex A: STIX 2.0 Versus 1.x".

3.2 IDEA

3.2.1 Introduction

Each organization will have different sets of requirements for types of cybersecurity-related data as well as how to processing that data. For the PROTECTIVE we have chosen to use the IDEA schema (CESNET, 2017). This schema aimed being a middle ground between having the complexity of IDMEF[18] and still be "free spirited" akin to AbuseHelper. IDEA was developed by CESNET before the PROTECTIVE project started, and the majority of this section content from the IDEA public documentation[19].

3.2.2 Description

A variety of models exist for the communication between honeypots, agents, detection probes, none of them are used because of their limitations for general usage. The IDEA format is an attempt to define current requirements in CSIRTs and propose foundations for viable solution for security event model, taking into consideration existing formats, their benefits and drawbacks.

Listing 2 shows an example of JSON serialized IDEA of a synthetic security event:

[18] https://www.ietf.org/rfc/rfc4765.txt
[19] https://idea.cesnet.cz/en/index

```
{
   "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"
      }
   ]
}
```

Listing 2. Security event described in IDEA format[20]

3.2.3 Requirements

The requirements analyzed during the (earlier) design of the IDEA format were based on the CESNET experience with other formats, which CESNET used in the first version of the WARDEN system, and had been formulated as enumerated in the list below. In PROTECTIVE, we have found that these

[20] https://idea.cesnet.cz/en/index

requirements as well as the IDEA format as a whole, sufficiently conform to the requirements formulated for the current project.

The original IDEA requirements are defined as follows[21]:

- **Interoperability** – communication between automated detection systems and information aggregators of various types needs to be gathered and exchanged. This type of data manifests wide variability, so representation should be extensible enough to be prepared for unexpected or new types of security events.
- **Simplicity and lightweight** – the format should be simple as complexity and intricacy may result in ambiguity. Additionally, it was attempted to be able to generate security messages at the closest place of detection, ideally at the detection probe, which might run on resource restricted platform. Imposing extended software or hardware requirements or need for applying complex tools or libraries was avoided.
- **Completeness** – a single message should be able to describe all detected facets of a particular security event in time of detection, not the whole timeframe or modus operandi of security case. For example, a single phishing email message may yield information about sender mail exchanger(s), reply-to address, URL of potentially malicious web page; these should be encompassed in a single detection message. However, if these pieces of information are gathered and aggregated from multiple sources or probes which are not directly related (e.g. email message and HTTP harvester), separate messages may be necessary for description.
- **Searchability** – the format should be searchable, therefore names of fields should be short and descriptive.
- **Non-recursiveness** – reasonably flat structure greatly simplifies processing, interpretation and storage in structured repositories (for example relational database systems).
- **Definiteness** – similar concepts should be represented in a unique, intuitive manner in a clearly visible, unambiguous parts of the message structure. The format users must clearly know where to insert particular pieces of information into the message and where to seek for information they want to obtain.
- **Programmability** – the selected format should have straightforward representation in common programming languages and data models. This would also allow for independence on serialization (JSON, XML, binary, etc.).
- **Semantic certainness** – data types and semantics of data fields should be unambiguous – one attribute should have only one strictly defined type and its type should not depend on the value of other attributes.
- **Compatibility** – the format should be able to reasonably cooperate with other existing formats. Reasonability is defined there as minimum necessary loss of information in translation other than the one based on fundamental design differences between IDEA and other formats.
- **Automation supporting structure** – the format should define structure only for information potentially analysable by machine means. Information which is meant only for human review does not need to be overly explicitly structured.
- **Explicit anonymization** – in security field, privacy plays great role in establishing and keeping trust. Explicit anonymization was assumed to be allowed.

[21] https://idea.cesnet.cz/en/requirements

- **Explicit incompleteness** – the format should also support explicit data incompleteness – information about attack sometimes can be incomplete or imprecise, but information about it should anyway be able to be distributed.

3.2.4 Security Event Types Classification

In order to remain consistent and keep information about security threats comprehensive enough, an appropriate classification (taxonomy) must be provided. When designing the IDEA format, a native classification has been approved that bases on a slightly extended taxonomy. This taxonomy, namely "mkII". This taxonomy was presented to the public during the 39th meeting of TF-CSIRT[22].

To classify security events, they must first be appropriately categorized and particular categories must be named. It may be that a security event matches multiple categories. In such cases all applicable category names must be used – for instance, phishing, detected from spam message, must be marked as both "Abusive.Spam" and "Fraud.Phishing". If it is not sure what is the precise nature of the incident, only the top level category name (omitting dot and subcategory) should be used. The section below provides the original description of the classification used by the IDEA creators[23].

The producer (sender) of an IDEA message should do the best to describe security events by existing category names, however completely new category/subcategory names may be used if none of the existing are applicable; this should however happen only in the case completely new security event type or modus operandi turns up in the wild. In case event is generated by machine and no human can assess new category, "Other" category can be used.

Category	Subcategory	Description
Abusive	Spam	Unsolicited Bulk Email, this means that the recipient has not granted verifiable permission for the message to be sent and that the message is sent as part of a larger collection of messages, all having a functionally comparable content.
	Harassment	Discreditation or discrimination of somebody (e.g. cyberstalking, racism and threats against one or more individuals)
	Child	Child pornography, Glorification of violence, etc.
	Sexual	
	Violence	
Malware	Virus	Slightly different types of software that is intentionally included or inserted in a system for harmful purpose. A user interaction is normally necessary to activate the code.
	Worm	
	Trojan	
	Spyware	
	Dialer	
	Rootkit	
Recon	Scanning	Attacks that send requests to a system to discover weak points. This includes also some kind of testing processes to gather information about hosts, services and accounts. Examples: fingerd, DNS querying, ICMP, SMTP (EXPN, RCPT …), port scanning and host sweeping.

[22] https://www.terena.org/activities/tf-csirt/meeting39/20130523-DV1.pdf. MkII is Don Stikvoort's update to eCsirt.net taxonomy, to the best knowledge of the PROTECTIVE consortium this is the most recent one.
[23] https://idea.cesnet.cz/en/classifications

Category	Subcategory	Description
	Sniffing	Observing and recording of network traffic (wiretapping).
	SocialEngineering	Gathering information from a human being in a non-technical way (e.g. lies, tricks, bribes or threats).
	Searching	Google hacking or suspicious searches against site.
Attempt	Exploit	An attempt to compromise a system or to disrupt any service by exploiting vulnerabilities with a standardized identifier such as CVE name (e.g. buffer overflow, backdoors, cross site scripting, etc.).
	Login	Multiple login attempts (guessing/cracking of passwords, brute force).
	NewSignature	An attempt using and unknown exploit.
Intrusion	AdminCompromise	A successful compromise of a system or application (service). This can have been caused remote by a known or new vulnerability, but also by an unauthorized local access. Also includes being part of a botnet.
	UserCompromise	
	AppCompromise	
	Botnet	
Availability	DoS	System bombarded with so many requests (packets, connections) that the operations are delayed or the system crashes. DoS examples are ICMP and SYN floods, Teardrop attacks and mail-bombing. DDoS often is based on DoS attacks originating from botnets, but also other scenarios exist like DNS Amplification attacks.
	DDoS	
	Sabotage	Outage, caused by local actions (destruction, disruption of power supply, etc.) - willfully or caused by deliberate gross neglect.
	Outage	Outage, caused by Act of God, spontaneous failures or human error, without malice or deliberate gross neglect being involved.
Information	UnauthorizedAccess	Besides a local abuse of data and systems the information security can be endangered by a successful account or application compromise. Furthermore, attacks are possible that intercept and access information during transmission (wiretapping, spoofing or hijacking). Human configuration/software error can also be the cause.
	UnauthorizedModification	Similar to the above but associated with modification of the accessed information.
Fraud	UnauthorizedUsage	Using resources for unauthorized purposes including profit-making ventures (E.g. the use of e-mail to participate in illegal profit chain letters or pyramid schemes).
	Copyright	Offering or installing copies of unlicensed commercial software or other copyright protected materials (Warez).
	Masquerade	Type of attacks in which one entity illegitimately assumes the identity of another in order to benefit from it.
	Phishing	Masquerading as another entity in order to persuade the user to reveal a private credential.
	Scam	Fraud on a person by falsely gaining confidence. Prominent example is Nigerian 419 scam.
Vulnerable	Open	Open for abuse, solvable preferably by patch, update, etc. - vulnerabilities apparent from Nessus and similar scans

Category	Subcategory	Description
	Config	Open for abuse, solvable preferably by configuration hardening/fixing - open resolvers, world readable printers, virus signatures not up-to-date, etc.
Anomaly	Traffic	Anomalies not yet identified as clear security problem, on different levels
	Connection	
	Protocol	
	System	
	Application	
	Behaviour	
Other	n.a.	Yet unknown/unidentified type of attack, or attack unrecognized automatically by machine.
Test	n.a.	Meant for testing.

Table 1. IDEA security events type classification[24]

3.2.5 IDEA selection as PROTECTIVE Information Exchange Format

For all the possible types of the cybersecurity related data it is not possible to either select or create the perfect solution (as each organization will have different sets of requirements, concerning processing information on different levels). For the PROTECTIVE project purposes **we have selected, as the first instance, IDEA format[25].**

As mentioned, IDEA provides the properties that has been assessed as the most accurate for the needs of the PROTECTIVE consortium. The main advantages of IDEA data format are as follows (see also the section 3.2.3 for the IDEA requirements):

- **Extensibility** – easy to include new, non colliding keys (in the case of occurring a new type of security alert)
- **Simplicity** – IDEA is modest on resources (the probes and sources of primary data can stay lightweight)
- **Searchability** – descriptive fields, flat structure, no recursion
- **Similarity** and **expectability**
- Straightforward **mapping** to common structures
- **JSON based**, which easily maps to common data structures, however serialization is up to implementors (JSON, YAML, XML, BSON etc.)
- **Basic data types** – (string, integer, real, Booleanm dics, list)
- **Unambiguity** – no interdependence of types
- **Conceptual data types** – timestamp, IPv4, IPv6, MAC, free enum, ID
- At most two level depth – more friendly to relational approach
- **Cooperation** – simple enough for sane mapping of other formats
- Parseability for machines, but descriptiveness for humans
- Support of anonymization, incompleteness an impreciseness

Thanks to the mentioned characteristics, IDEA covers most of PROTECTIVE needs in area of dealing with data (gathering, filtering, analysis, storing, correlation, sharing) and fits much better than STIX

[24] Based on: https://idea.cesnet.cz/en/classifications
[25] https://idea.cesnet.cz/en

to the sets of cybersecurity related data maintained by public organizations like NRENs which compose an important use case for the PROTECTIVE project. We foresee that IDEA will be easily extended to the SME use case especially when assuming enrichment functionalities utilizing publicly available sources of cybersecurity information.

IDEA represents the lower technical level than STIX, being thus a better fit in the context of gathering of primary events data from on site honeypots, probes and other detectors, whereas it is prepared (and easily extendable) for the planned correlation and enrichment facilities. Its structure is detailed, but simple enough (especially not recursive) and not ambiguous. It facilitates processing of large number of events. It must be noted that IDEA has been inspired by IDMEF (which is described in section 3.3.1) but is noticeably simplified and updated to nowadays security expectations in comparison to it. Interoperability between IDEA and IDMEF is briefly described in the aforementioned section as well.

Additionally, IDEA format has built-in anonymization facilities which is extremely important when dealing with TI sharing between different partners, possibly unwilling or even prohibited to disclose certain information about attacks affecting their infrastructure.

It must be clearly mentioned that IDEA is a result of work of a CSIRT team (CESNET) who found other well recognized formats as insufficient in the field of TI sharing – at least for the purposes of the team, that, however, significantly match PROTECTIVE TI sharing requirements. Therefore the PROTECTIVE consortium do not have to spend time, resulting in conclusions that had already been reached.

The section 3.3 provides information on other well recognized formats: IDMEF, X-ARF and IODEF. While describing them, an additional explanation is provided why their properties do not fit as well as IDEA's for the purposes of the PROTECTIVE suite. However, it is worth noting that for other operations (e.g. storing the events in the database and processing them) it is possible to select another format due to specific requirements of analytical mechanisms. For instance, in order to store information about connections in a network graph databases may be used that better conform to the connection model. Such databases may in turn operate better on other types of data than represented by a particular format.

Therefore we do not absolutely exclude all other formats. But as for now we have decided that **IDEA format will be used for TI sharing** (exchanging the information about security events) in the PROTECTIVE project.

3.3 Other Formats

This section describes the review of existing formats and tries to explain, why we are not using one of them. It works as summarization of benefits and drawbacks of each of them, together with our subjective remarks. An important note to make is that none of the following mentioned formats directly supports anonymization, which has been one of the requirements fulfilled by IDEA (see the section 3.2.3). The following properties as well as considerations on applicability of the other format compared to IDEA, have been prepared basing on the work of IDEA format creators)[26].

3.3.1 IDMEF

Intrusion Detection Message Exchange Format[27] is a format created for exchange of information about security events between detection probes. It is based on and very tightly coupled with XML and therefore its structure causes difficulties in efficient processing of large number of events.

[26] https://idea.cesnet.cz/en/other_formats
[27] https://www.ietf.org/rfc/rfc4765.txt

It has an inflexible structure but on the other hand certain fields are dynamic (it is for example able to represent more sources or destinations of the attack). The structure is also relatively deep and overly verbose (for instance, in order to address an attack source IP address it is necessary to use the "Alert.Source.Node.Address.address" locator. Moreover, certain fields are recursive, which causes the depth of message structure is actually arbitrary (not limited).

IDMEF has limited means facilitating extensibility, by:

- global key/value pairs, which are however tied to message, not to specific source, target or other key,
- own XML namespace in specific place of the structure (which introduces the need for a full-fledged XML parser and causes higher complexity of tools processing IDMEF).

IDMEF specification often mentions subclassing and aggregation, however these are reserved for specification authors and (possibly) for some next official versions of IDMEF. The format is often redundant – for instance, timestamps may be represented both in machine readable format and human friendly representation, which creates ambiguity in case of error.

It is worth noting that IDMEF is inconsistent in certain cases – it supports an extensive list of historic (or obsolete) network protocols in one field, however URL and SNMP are classes of their own. Incident type is represented by an arbitrary text description with only several extensions of specific classes (for buffer overflow, correlation, and tool). The format can be validated against the thoroughly defined schema, which is a positive feature, however the schema itself is not enough – certain cases must be validated specifically (for example timestamps and IP addresses).

IDMEF forms the basic idea behind IDEA, as the format able to describe the highest number of incident types. The basic problems, however, stem from its verbosity, depth and attempts to describe "everything". In our experience, several means to group and structure various types of information are never used in real life scenarios. Another IDMEF drawback is also the need for complex libraries – lightweight detection probes are usually not able to generate IDMEF messages directly and need some kind of intermediate format and translator.

3.3.2 X-ARF

X-ARF Network Abuse Reporting[28] is a format directed at exchange of incident reports by email between security teams. Its purpose is not for exchange between systems. We have analyzed it as security event messages are often exchanged between different groups of security specialists.

The event description bases on a set of keys and values, which is enclosed in a MIME/DSN encoded container along with human readable description as well as optional attachments. The structure is rigid and static – a single message is only able to describe exactly one source and one destination.

Different types of events have their specific schemes – X-ARF provides only a limited set of values which is common for all types. Particular types have their own sets of distinct descriptors. So far there are only four different schemes available, defined for login attack, malware attack, fraud and DNSBL informational message[29].

[28] http://www.x-arf.org/
[29] http://x-arf.org/schemata.html

3.3.3 IODEF

Incident Object Description Exchange Format[30] is meant for exchange of incident report description between CSIRTs. This is evident for instance as the format contains Action element which is expected to be used to communicate a recommended or required action.

IODEF is able to describe the whole case and its timeframe, not only a single event – each decisive or communication action can be documented. Like IDMEF, it has limited means for extending, but rather may be used in several explicitly defined scenarios.

The specification also provides information on limited IDMEF interoperability. Concerning interoperability with IDEA, the latter took a lot of inspiration from IDMEF, but also dropped a wide set of its features for the sake of simplicity and therefore processing efficiency. However, conversion from IDEA to IDMEF should be easily doable without loss of significant amount of crucial information. Conversion from IDMEF to IDEA (or generally IDMEF to anything) may be much harder (and not unambiguous), however most of basic IDMEF events should be mostly translatable. Problems may be recursive structure of IDMEF and parts of IDMEF completely unsupported by IDEA (organisational information etc.).

According to observations made by CESNET, there are only two use cases where IDMEF can be actually applied to directly support other formats: 1) IDMEF can be recomposed into IODEF; or 2) IDMEF can be inserted “as is” into an IncidentAlert wrapper node. The problems with the latter approach are evident – analysis tools must be able to decipher the complexity of both formats.

The representation of the Assessment field is fluent, types and semantics are dependent on attributes. While the sender of the IODEF message may choose a particular representation, which it will continue to use, the receiver of this message has to be informed and apply this choice (or, if this is not possible, may have to attempt interpretation of all possible representations).

[30] http://tools.ietf.org/rfc/rfc5070

4 Solutions Analyzed as Potential PROTECTIVE Main Components

Several tools were trialled while searching existing solutions that may have complementary features that might support PROTECTIVE. The goal of this chapter is not to provide an exhaustive list of all available solutions but rather to point out important representative examples of tools available. This chapter outlines these tools and their main features[31].

4.1 IntelMQ

4.1.1 Overview

IntelMQ is a solution for IT security teams for collecting and processing security feeds whose purpose is to create a highly-modular system for the collection, processing and distribution of security information (IntelMQ, 2017). Its main goal is to give to incident responders an easy way to collect and process TI thus improving the incident handling processes of CERTs.

Figure 7 shows general IntelMQ architecture.

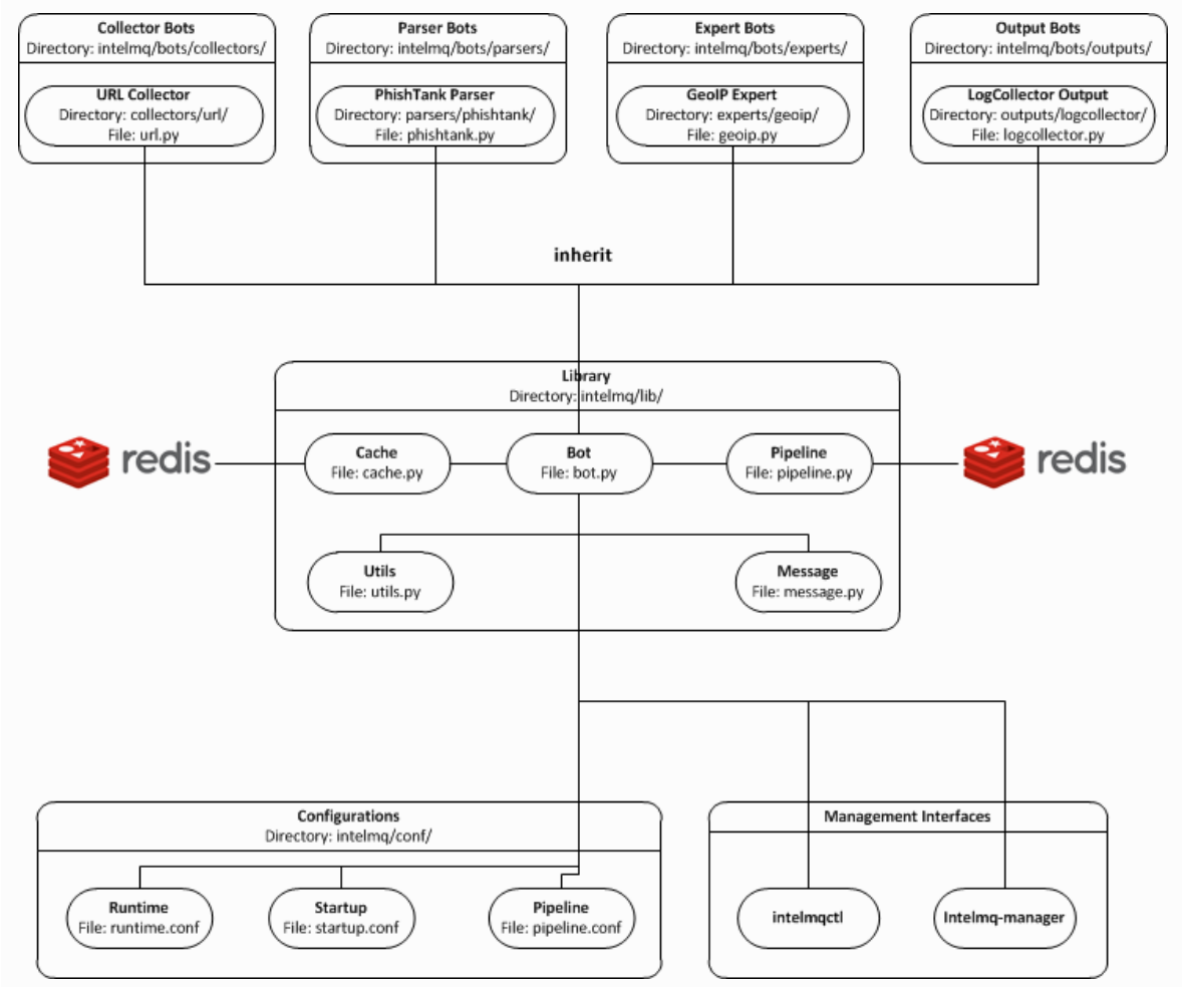


Figure 7. IntelMQ System Overview[32]

IntelMQ contains a graphical interface to manage configurations for the IntelMQ framework (IntelMQ Manager). An IntelMQ configuration is a set of configuration files which describe which

[31] Much of the descriptive text in this chapter is based on the descriptions from the relevant websites. While we have aimed to avoid excessive replication there is still an unavoidable degree of overlap do to the fact that these websites are in many cases the authoritative, and in some cases the only, sources of information.

[32] http://intelmq.readthedocs.io/en/latest/Developers-Guide/

processing steps should be run in which order. It is similar to designing flows in dataflow oriented languages.

IntelMQ's design was influenced by an earlier tool – AbuseHelper.

4.1.2 Relevance to PROTECTIVE

IntelMQ provides an integration backplane to chain processing modules together to form an information processing pipeline. It also provides a number of functional modules for information enrichment, storage etc. It is a strong candidate to provide the "glue" fabric to implement the PROTECTIVE pipeline.

4.2 TheHive

4.2.1 Overview

TheHive is a "case" (i.e. investigation) management system designed to support SOCs, CSIRTs, CERTs and information security practitioners dealing with security incidents that need to be investigated. Each such investigation is considered as a "case" (TheHive, 2016).

Its main features include[33]:

- **Collaboration** – multiple analysts can work on the same case simultaneously.
- **Elaboration** – Cases can be created from scratch and tasks added and dispatched to (or taken by) available analysts. Each task can have multiple work logs where contributing analysts may describe what they are up to, what was the outcome, attach pieces of evidence or noteworthy files, etc.
- **Analysis** – TheHive comes also with an analysis engine called Cortex. Cortex 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.

TheHive is written in Scala and uses ElasticSearch to store and access data on the back end. The front end uses AngularJS and Bootstrap. A number of REST API endpoints are also provided to allow for integrations and bulk actions. The architecture is shown in Figure 8:

[33] https://blog.thehive-project.org/2016/11/07/introducing-thehive/

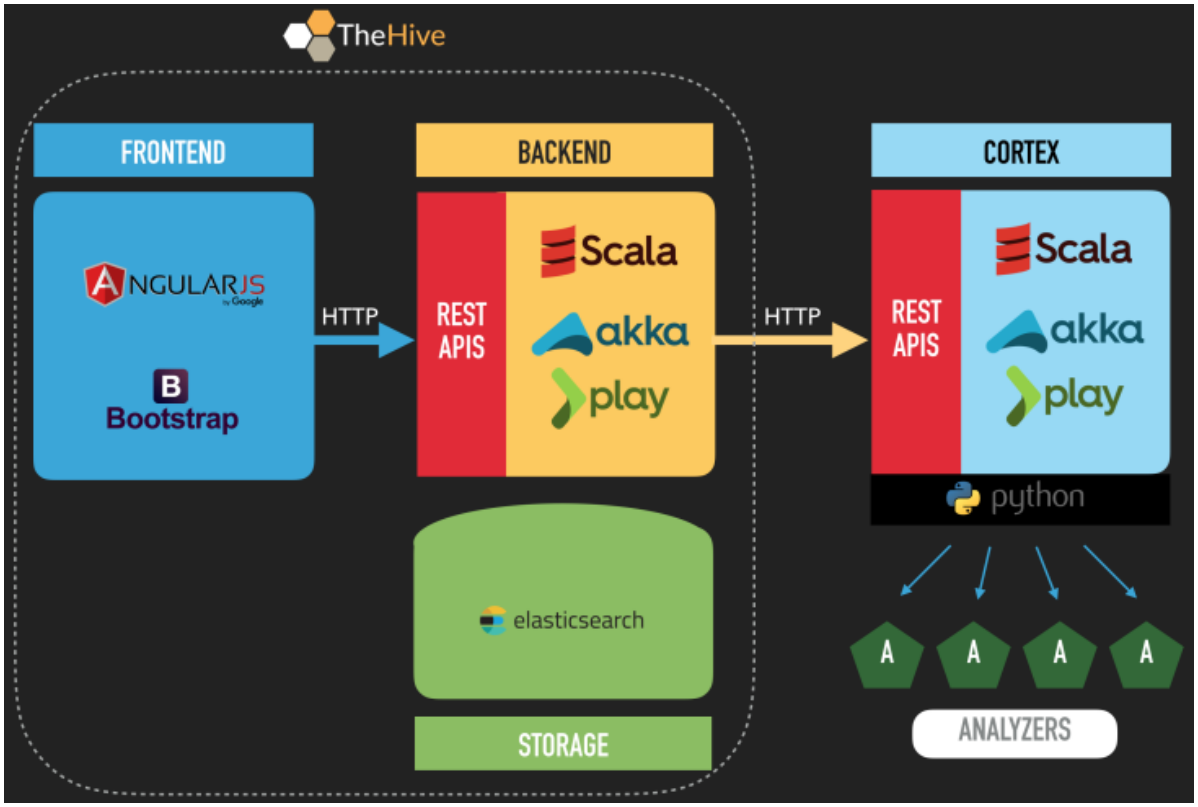


Figure 8. TheHive Architecture[34]

It should be noted that Cortex can also be used in a standalone mode and therefore its features may be used with more flexibility, without the need for installing other TheHive components.

TheHive can also be integrated with MISP (see next section) to provide an even more powerful incident handling toolset.

TheHive and Cortex are open source and free software released under the AGPL (Affero General Public License).

4.2.2 Relevance to PROTECTIVE

TheHIVE is complementary to PROTECTIVE in its purpose – its focus is primarily on incident i.e. "case" handling which, from a process point of view is downstream from PROTECTIVE. The Cortex component could be reusable as an information enrichment platform.

4.3 MISP

4.3.1 Introduction

MISP (Malware Information Sharing Platform and Threat Sharing) is an open source software solution for collecting, storing, distributing and sharing security indicators and threats about cyber security incidents analysis and malware analysis. (MISP, 2013). Its main goal is to facilitate sharing information related to targeted attacks and malware. Its primary features include a centralized searchable data repository, a flexible sharing mechanism based on defined trust groups and semi-anonymized discussion boards.

MISP is one of the most widely used in the CERT/CSIRT community

Following the MISP documentation[35], some of the main features of MISP are:

[34] https://blog.thehive-project.org/2017/02/01/meet-buckfast-cortex-the-dynamic-duo/

- IOC and indicators database allowing to store technical and non-technical information about malware samples, incidents, attackers and intelligence.
- Automatic correlation to find relationships between attributes and indicators from malware, attacks campaigns or analysis.
- Storing data in a structured format - allowing automated use of the database for various purpose).
- Export: generating IDS, OpenIOC, plain text, CSV, MISP XML or JSON output to integrate with other systems (network IDS, host IDS, custom tools).
- Import: bulk-import, batch-import, import from OpenIOC, GFI sandbox, ThreatConnect CSV
- Data-sharing: automatically exchange and synchronization with other parties and trust-groups using MISP.
- Adjustable taxonomy to classify and tag events following custom classification schemes.
- STIX support: export data in the STIX format (XML and JSON).

4.3.2 Relevance to PROTECTIVE

MISP and PROTECTIVE have many similarities. Unlike PROTECTIVE however, MISP focuses on the exchange of the most valuable indicators selected and annotated by analysts, and not on processing high-volume automated data feeds.

MISP could be envisaged as a complement to PROTECTIVE and may be possible to integrate with PROTECTIVE as it has with TheHive. Further many of the features of MISP e.g. such as it sharing community management could serve as functional basis for PROTECTIVE TI sharing community design.

4.4 Apache Spot – ONI (Open Network Insight) Project

4.4.1 Overview

Apache Spot is open source platform for packet and flow analytics based on Hadoop. It provides ingest and transform of binary data, scalable machine learning, and interactive visualization for identifying threats in network flows and DNS packets (ONI, 2015). It uses machine learning as a filter for separating malicious traffic from benign and to characterize the behaviour of network traffic. A process, of context enrichment, noise filtering, whitelisting and heuristics, is applied to network data to produce a shortlist of most likely security threats.

The architecture of Spot is shown in Figure 9.

[35] http://www.misp-project.org/features.html

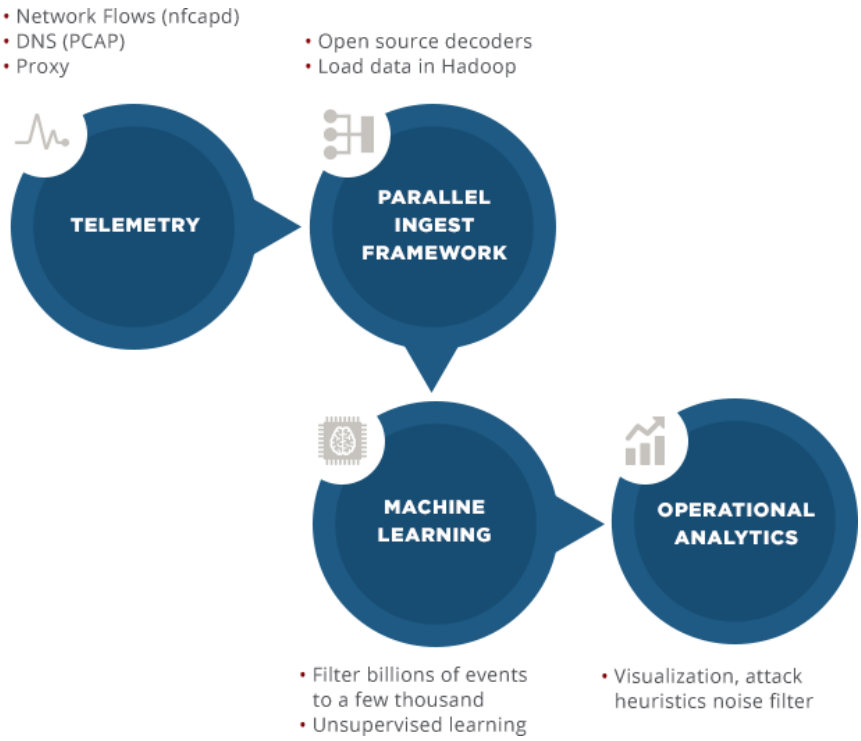


Figure 9. Apache Spot How It Works[36]

Following the Apache Spot website[37], the key features of Apache Spot may be defined as follows.

- Suspicious DNS packets detection.
- Threat Incident and Response.
- Suspicious Connects. Apache Spot uses advanced machine learning to build a model of the machines on the network and their communication patterns. The connections between the machines that are the lowest probability are then visualized, filtered for noise, and searched for known patterns. The result is the most likely threat patterns in the data, a few hundred flows picked from billions.
- Open Data Models (ODM). Spot provides common open data model for network, endpoint, and user – Open Data Models. These Open Data Models provide a standard format of enriched event data that makes it easier to integrate cross application data to gain complete enterprise visibility and develop net new analytic functionality.

The primary use case initially supported by Spot includes Network Traffic Analysis for network flows (Netflow, sflow, etc.), DNS and Proxy. The Spot open data model strategy aims to extend Spot capabilities to support a broader set of cybersecurity use cases.

4.4.2 Relevance to PROTECTIVE

Apache Spot addresses a different aspect of the cyber-threat space to PROTECTIVE i.e. it attempts to detect suspicious network data and traffic flows and to raise an alert or alarm when it does so. As such its outputs could be used as input to PROTECTIVE. The ONI-ingest module could be a candidate component for the corresponding PROTECTIVE part. However it was noted that

- The project roadmap was empty without any clear plans for further development
- The project is still in Apache Incubation

[36] http://spot.incubator.apache.org/
[37] http://spot.incubator.apache.org/

4.5 Warden

4.5.1 Overview

The Warden project is a platform for **automated sharing of detected security events** among security teams[38]. It enables CERTS/CSIRT teams (and security teams in general) to share and make use of information on detected anomalies in network and services operation generated by different systems – IDS, honeypots, network probes, traffic logs, etc. – easily and efficiently. The detailed information on Warden's origin may be found at its Web page[39]. The architecture of the Warden system is that of the client-server type. Warden consists of a server, receiving clients and sending clients. The server, on request of receiving clients, distributes new (previously undistributed) events fed to the server by sending clients.

Each entity/network that wishes to feed data into the Warden system should have a so-called sending client. Each entity/network that wishes to receive data from the Warden system should have a so-called receiving client. The server (the centre) ensures the data reception and storage as well as the interface for the access to data stored. Data which the clients send into the centre will be referred to as events. Events are sent by the clients after authentication; the access to the centre is also authenticated. In particular, X.509 is used for the authentication.

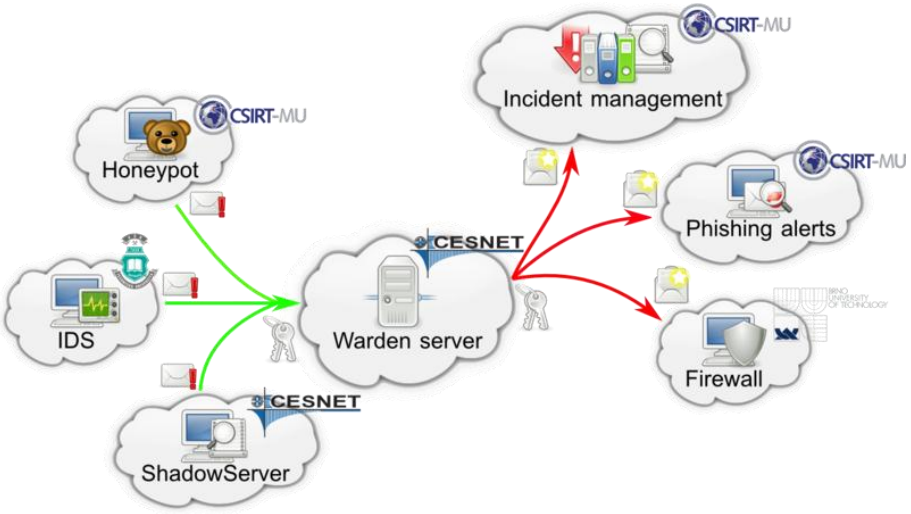


Figure 10. Warden System Architecture[40]

The two main features interesting from the PROTECTIVE toolset point of view are events sending and events receiving. Following the project documentation[41], they have been described below.

1. **Sending events**. Entities/networks involved feed the centre events from various data sources. These may be within the entity/network, i.e. data from detection system operated within the entity/network and monitoring network and service traffic in the network concerned (IDS, honeypots etc.) or the third party sources (like Shadowserver[42], Honeynet[43], various blacklists), or even the aggregated or correlated data. There are of course the ways to distinct these cases. Entities/networks involved feed the centre events from various data sources. Warden can accept data from sources **with the IDEA format** (see section 3.2) or

[38] Warden is developed by PROTECTIVE consortium member CESNET, as is the MENTAT tool.
[39] https://warden.cesnet.cz/en/about_project
[40] https://warden.cesnet.cz/en/architecture
[41] https://warden.cesnet.cz/en/architecture
[42] https://www.shadowserver.org/wiki/
[43] https://www.honeynet.org/

also accepts data from sources which are able to send the data in the IDEA data format (that is sources for which there is a "connector" available that is able to translate data from the internal format of the security tool into a "security event" in the IDEA format).

2. **Receiving events**. Each entity participating in the system may receive data from the Warden system (through a receiving client). The server (centre) provides participating entities relevant events that it had been sent earlier – in an unmodified form. Moreover, the server sends the client only these events which have not been sent to them so far, or a notification that there is no new (unsent) event on the server. However, the latter happens only if the client explicitly asks the server for the most recent events. Participating entities may use data obtained from Warden as necessary to ensure the security of their own network and services provided. Through Warden it is possible to receive the information relating to a network (organization), download the data and process them in a particular manner (manually, but also automatically within tools like Mentat, in SIEM etc.).

4.5.2 Relevance to PROTECTIVE

Warden is used to ingest and share TI. It is primarily an information broker and routes information to predetermined, configured, destinations. It could be used as a base for the TI sharing functions within PROTECTIVE.

4.6 Mentat

4.6.1 Overview

Mentat is a distributed modular Security Information Management System (SIEM) designed to monitor networks of all sizes. Its architecture enables reception, storage, analysis, processing and response to a great volume of security incidents originating from various sources, such as honeypots, network probes, log analysers, third party detection services, etc. Although the source code has not yet been made publicly available, the Mentat system has been developed as an open-source project. (Mentat, 2017).

The Mentat system is a platform enabling to unify the collation and subsequent processing and managing of various detected security events coming from a wide range of different detection systems.

Figure 11 below provides an overview of the existing architecture of the Mentat system.

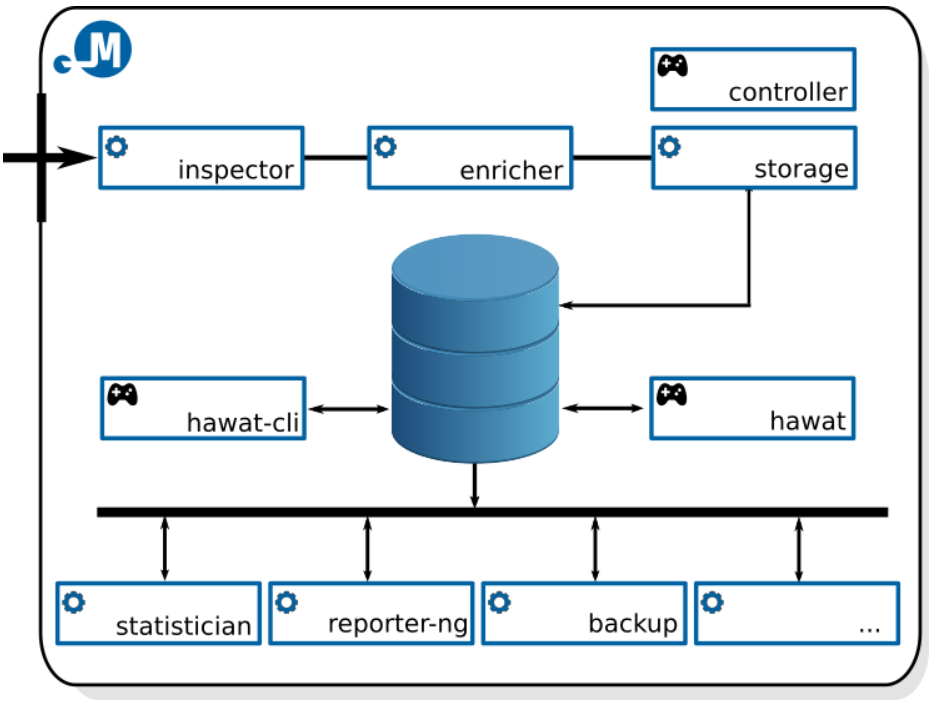


Figure 11. Mentat System Architecture[44]

The Mentat system consists of tools allowing processing events both in real time and retrospectively over a particular period of time. At present, the following modules for real time processing are available (described following the earlier provided system documentation[45]):

- **mentat-inspector** – enables the processing of IDEA messages based on the result of given filtering expression. There is a number of actions that can be performed on the message in case the filtering expression evaluates as true.
- **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.
- **mentat-storage –** enables to store incoming IDEA notifications in a database (MongoDB).
- **mentat-statistician** – this feature enables statistical processing of events over a given self-defined period. At present, the feature is present to five-minute intervals. For each of these intervals, it determines the frequency of events according to detector type, event type, IP address etc. These statistical reports are stored in a separate database and can later support an overview of system's operation, provide underlying data for other statistical reports or for the creation of dictionaries for a web interface.
- **mentat-reporter-ng** – enables to distribute periodical event reports directly to end abuse contacts of responsible network administrators.
- **mentat-briefer** – this feature is similar to the above described reporter. It provides periodical summary reports on system's statues and reports sent.
- **mentat-backup** – a configurable feature enabling periodical database backups. At present, a full backup of system collections (users, groups, etc.) is created once a day while event collections are backed up incrementally.

[44] https://mentat.cesnet.cz/en/architecture
[45] https://mentat.cesnet.cz/en/architecture

- **mentat-cleanup** – a configurable feature enabling periodical database clean-ups.
- **mentat-precache** – a configurable feature enabling data caching, in particular of various dictionaries for Web interface.
- **hawat-negistry** – enables data synchronisation between Negistry and Mentat's system database. It synchronises abuse groups and address blocks assigned to them.

The last important components of the system are the **administrative interfaces**:

- **hawat** – a web interface for the Mentat system. The interface enables in particular to search through the event database and sent reports, system statistics and overviews and to configure the entire system and the reporting algorithm in particular.
- **hawat-cli** – CLI interface for system administrators enabling the automation of certain acts relating to the administration of the Mentat system.
- **mentat-controler** – a script enabling to control particular components on a given engine.

4.6.2 Relevance to PROTECTIVE

Mentat is conceptually similar to IntelMQ in that's it provides an integration backplane to build information processing pipelines, albeit with a significantly different implementation. Mentat is a strong candidate, alongside IntelMQ, to provide the integration "glue" for PROTECTIVE.

5 Technological Baseline Selection for the PROTECTIVE System

In this chapter we conduct a detailed evaluation and comparison of the tools described in the previous chapter in order determine their applicability to the PROTECTIVE framework. It is also worth noting that the most relevant technologies are described in section 5.3. In the interest of clarity, less relevant technologies have been moved to the Annex C: Additional Technology Scouting. The described research and selection of suitable security engines and technologies have been crucial tasks for the successful implementation of the core PROTECTIVE platform.

5.1 Cross Check of the Functionalities and Capabilities of the Systems Available on the Market

In this section, we list different functionalities of PROTECTIVE Framework as the primary capabilities PROTECTIVE should provide and how the analysed solutions match with the PROTECTIVE framework's various defined functionalities. These functionalities have been drawn as boxes in the standard flow that data information will be treated to, from the source (when it is collected) to the end (when it is processed and stored) and also classified into three main technical work packages: WP3, WP4 and WP5. The purpose of Figure 12-Figure 15 is to demonstrate how we have cross-checked the functionalities and capabilities of systems available on the market with what is necessary in a PROTECTIVE environment.

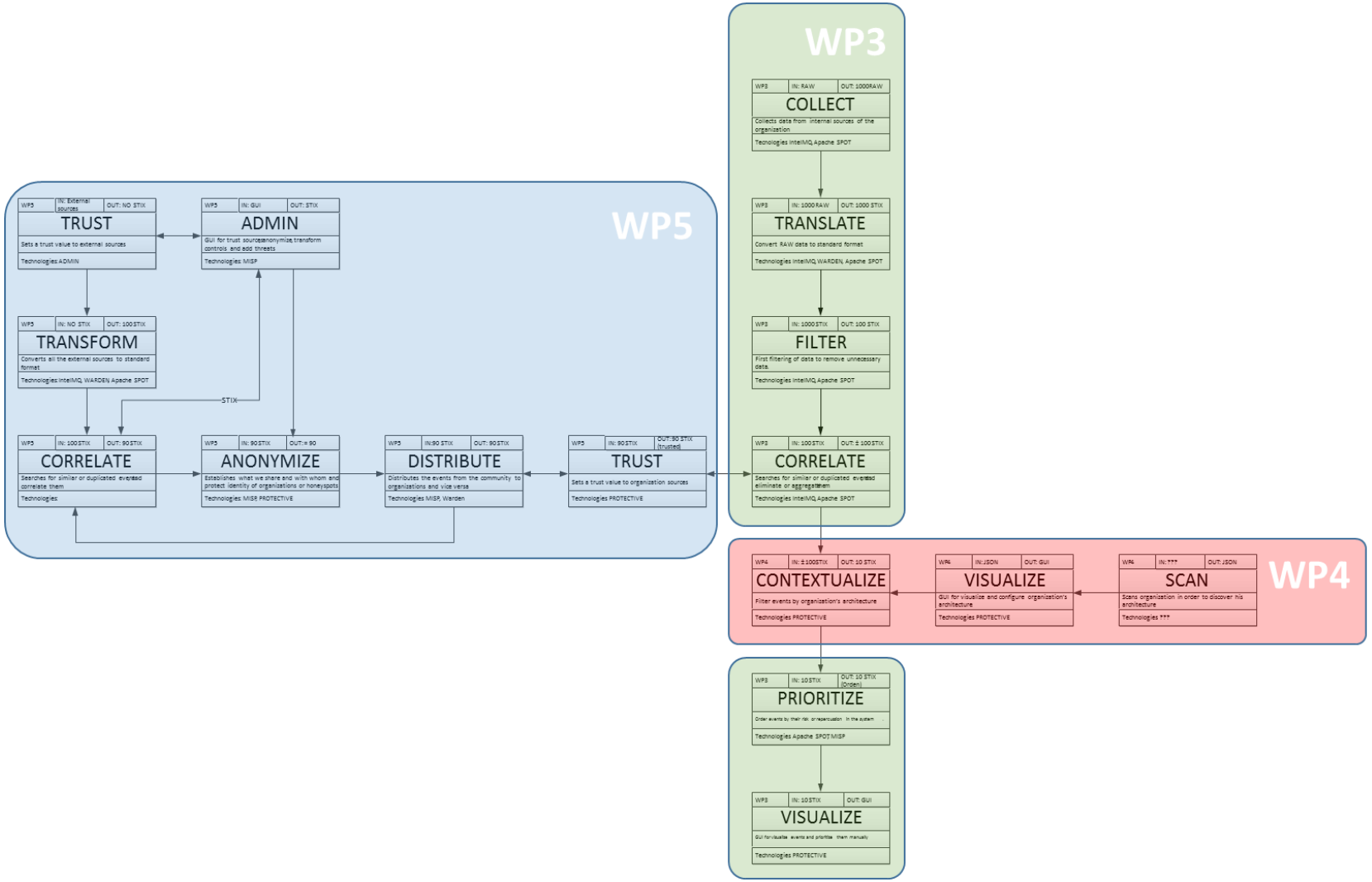


Figure 12. PROTECTIVE Capabilities

For each work package we detailed the main PROTECTIVE capabilities and how these capabilities match with the functionalities presented in the section 2.3 “PROTECTIVE Functional Modules”.

The WP3 includes these main functionalities:

- Correlation engines and advanced analytics
- Prioritisation and ranking building
- Data storage facilities

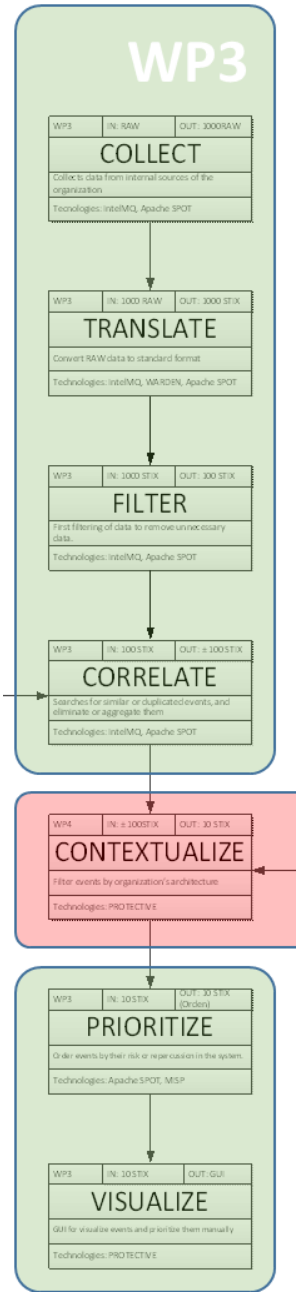


Figure 13. PROTECTIVE WP3 Capabilities

The WP4 includes the following main functionalities:

- Context Awareness
- Data storage facilities

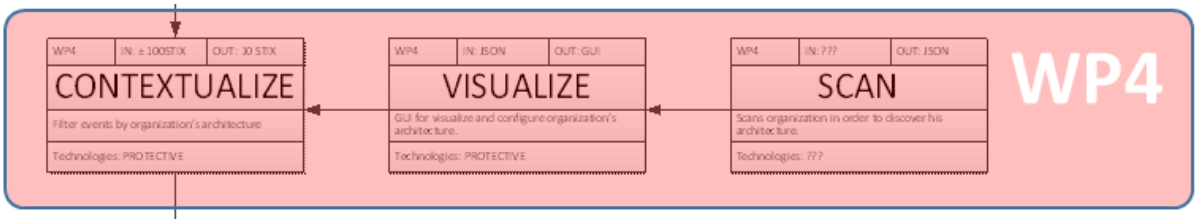


Figure 14. PROTECTIVE WP4 Capabilities

The WP5 includes these main functionalities:

- Ingestion mechanisms including: data receivers
- Data sharing mechanisms
- Access control mechanisms
- Data storage facilities

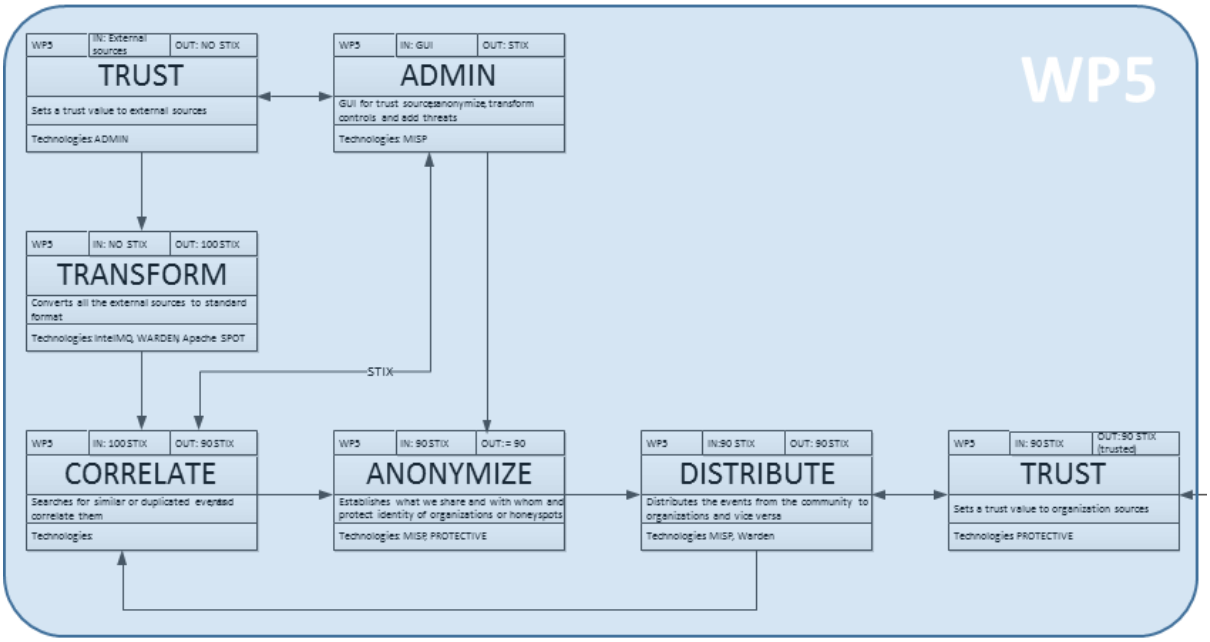


Figure 15. PROTECTIVE WP5 Capabilities

The following “primary capabilities” are used as a basis for comparison of the various tools:

- Collection
- Correlation
- Prioritization
- Contextualization1
- Decision Making
- Trust Calculation
- TI Distribution
- TI Admin
- TI Analytics

5.1.1 Warden vs. MISP vs. Apache Spot

The details of each product in the title are provided above in this document. In this chapter, the main positive aspects concerning the list of the key capabilities have been explained.

The main features of Warden**,** meaningful for PROTECTIVE, are as follows:

- It is operated and used by CESNET in e-infrastructure CESNET, also has been comprehensively examined, during the analysis, by other PROTECTIVE members (GMV, PSNC and RoEduNet), and proved to be highly functional.
- Contains both sending and receiving clients.
- Natively uses the selected IDEA data format.
- Collects data from various security sources (e.g. honeypots, IDS, network probes etc.).
- Sent data are available to other Warden partners.

The coverage chart shown below, as well as other coverage and comparison charts within this chapter, have been prepared by the PROTECTIVE consortium members. To calculate and standardize the shown values in % in the next figures we use a percentage scale between 0% (no coverage) and 100% (full coverage) (with only the following values: 0 “no coverage”, 20 “very poor coverage”, 40 “poor coverage”, 60 “medium coverage”, 80 “high coverage” and 100 “full coverage”, which are an estimation after the usage of the solution). These percentages indicate the approximate level of compliance with the expected PROTECTIVE main requirements for each of the capabilities analysed. Particular values have been chosen according to the experience of the PROTECTIVE consortium engineers installing and pre-evaluating particular software solutions at the selection stage.

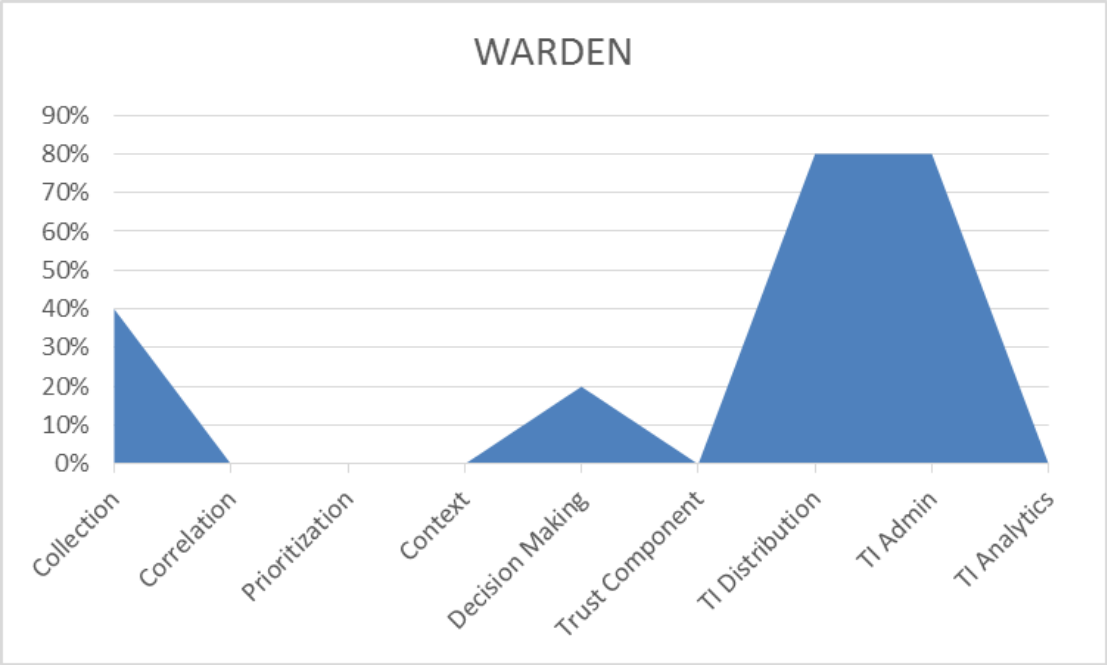


Figure 16. Warden Coverage

The main features of **MISP**, relevant for PROTECTIVE, are mentioned below:

- Was installed and tried out by PROTECTIVE members (GMV).
- Each organization manually sets their events.
- Each organization defines what to share, when and with whom.
- Each MISP server can pull data from other MISP servers.
- Used by +2500 organizations.
- Well documented.

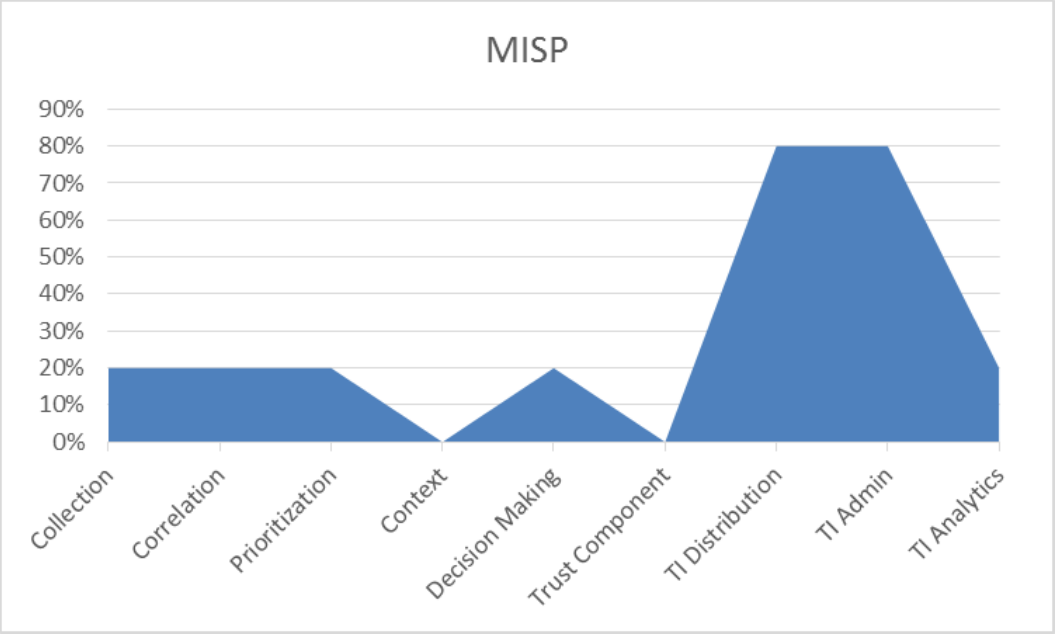


Figure 17. MISP Coverage

Finally, the most significant Apache Spot features from the PROTECTIVE point of view, are enumerated below:

- Was installed and tried out by PROTECTIVE members (GMV).
- Ingest module reads periodically DNS, Proxy and flow logs from a folder.
- Utilizes ML (Machine Learning) techniques to infer probabilistic model to prioritize the suspicions events for its analysis by the CSIRTs.
- It has an OA (Operational Analytics) module with functionalities for visualization and processing of the events investigated.
- It uses the capabilities of existing tool as Hadoop to improve data transformations, parallel programming and analytics.

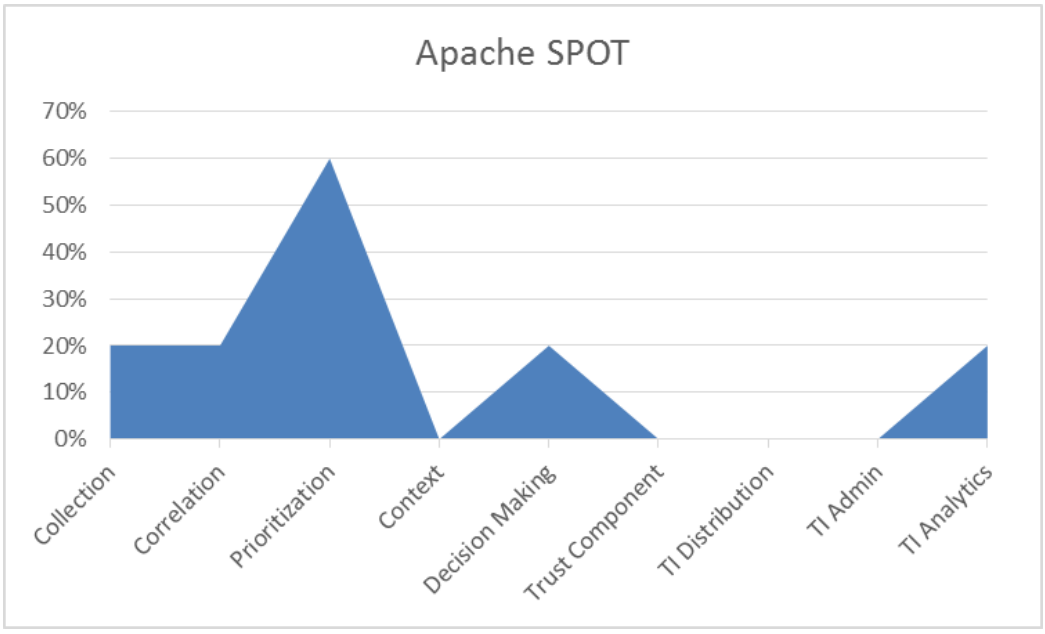


Figure 18. Apache SPOT Coverage

Out of the box, Warden has more coverage into “Collect” and “Normalize” functionalities than MISP, which in turn has more coverage into “Events sharing and lists” and “Correlation” functionalities than Warden. Both of them have high coverage into “Distribute” capability. Also Warden can be used to send events to a MISP system and MISP can be a source for Warden. A crucial difference between the two is the focus of MISP on low volume manual event creation vs the high colume automated event distribution of Warden.

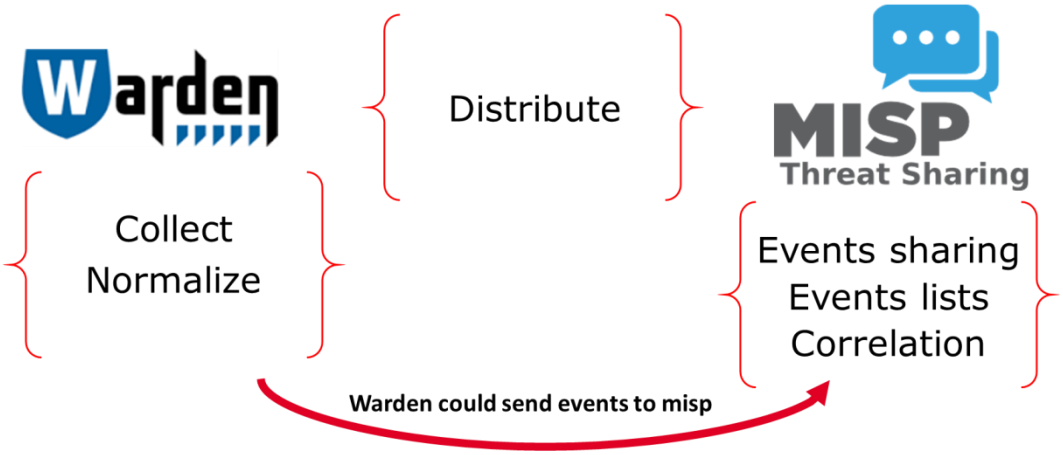


Figure 19. Warden – MISP (Comparative)

In the case of comparing MISP versus Apache SPOT we can observe the following differences:

MISP	Apache SPOT
• Adds events manually • Correlates similar events • Prioritizes event’s risks (High, Medium, Low) • Sharing configuration (Who & What) • Easy pull/push from other servers	• Collects events from folder • Automatic suspicious events detection • Machine Learning to improve detections • GUI for events visualization and storyboard generation

Table 2. MISP and Apache SPOT features comparison

An interesting point is that the Storyboard on Apache SPOT could be ingested by MISP. The picture below shows a possible relationship between MISP and Apache SPOT (the right side logo).

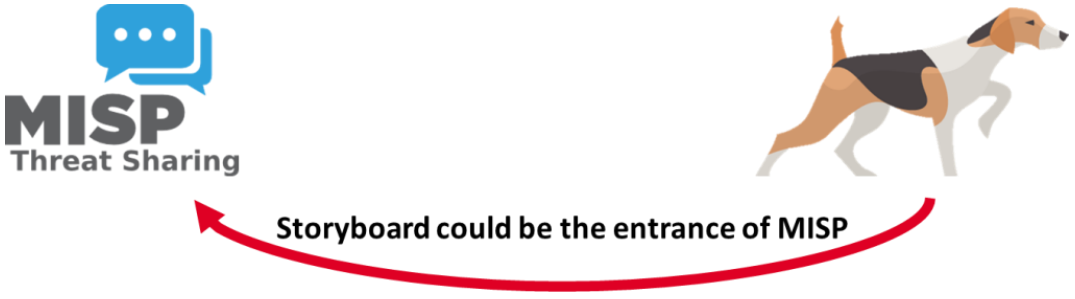


Figure 20. MISP – Apache SPOT (Comparative)

Table 3 shows the comparison of selected features for Warden with IDEA format, MISP and Apache SPOT. The triple dash (---) means either no coverage at all or poor coverage.

	Warden + IDEA	MISP	Apache SPOT
Features	Collects from public URLs and raw data	---	Collects from DNS, PROXY and FLOW
	---	Event list	Event list
	---	Manual prioritization	Prioritization based on Learning
	---	Correlation	---
	Sharing sender to all configured receivers	Sharing through rules point 2 point	---
	---	GUI Admin sharing	---

Table 3. Warden + IDEA, MISP and Apache Spot features comparison

Figure 21 below shows the comparison of the most significant features for the three mentioned main tools.

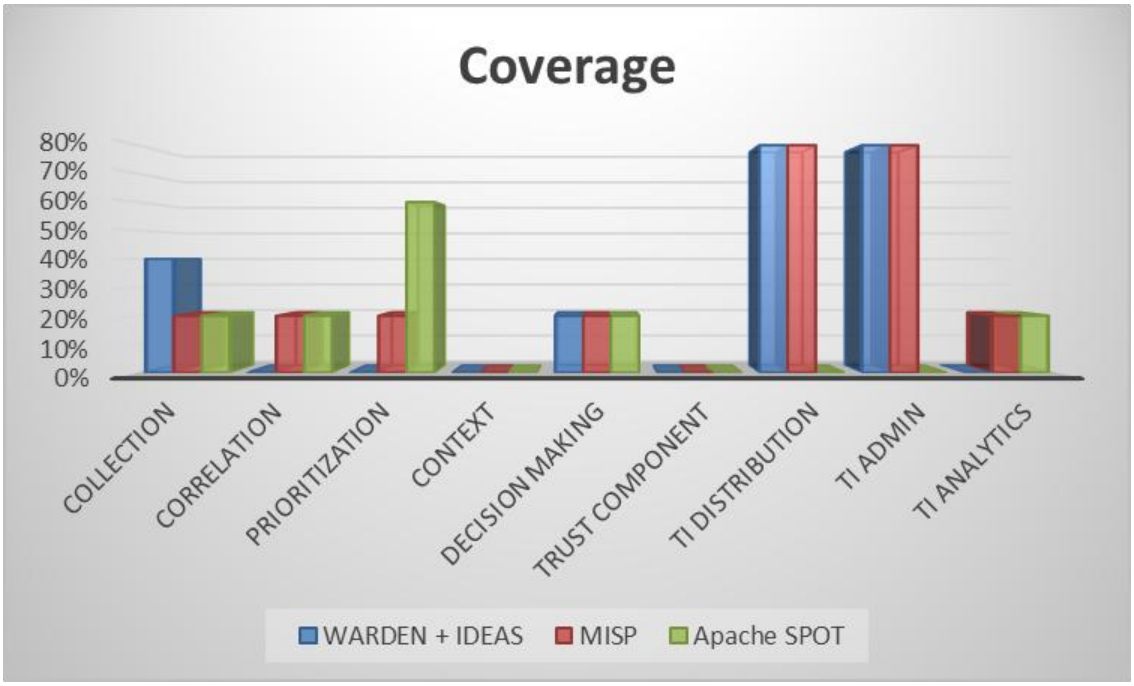


Figure 21. MISP – Warden – Apache SPOT (Comparative coverage)

5.1.2 IntelMQ vs. Mentat

The full details of each mentioned product are provided above in this document. In this section the main positive aspects concerning the list of the “big capabilities” have been explained.

IntelMQ

- It has been installed and tried by PROTECTIVE members (GMV)
- Reads from different sources, harmonizes them and redirects the results to other fonts
- Focuses on intensive data processing
- It has an already implemented MISP integration

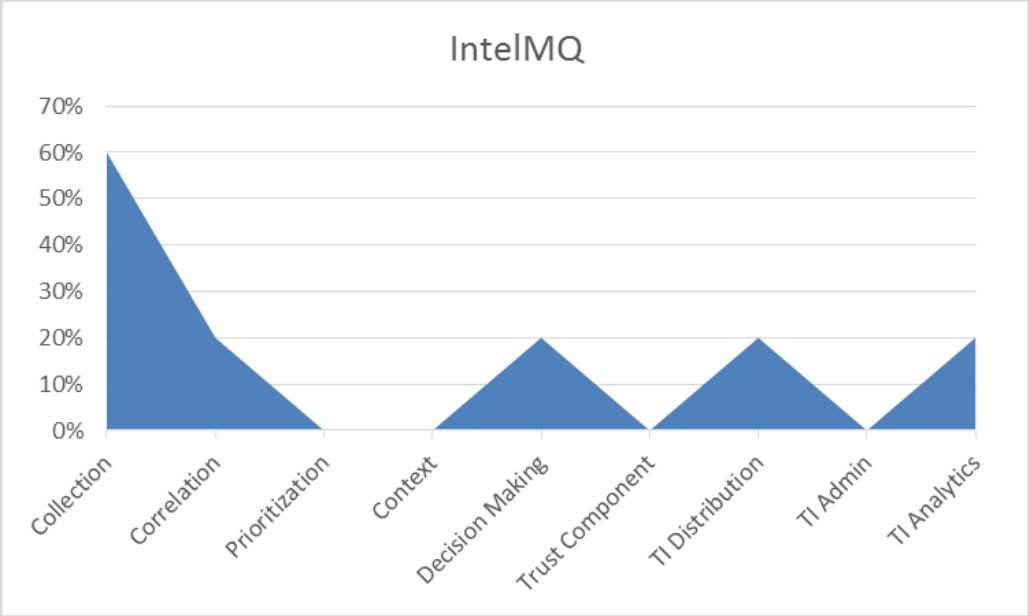


Figure 22. IntelMQ Coverage

Mentat

The description below briefly summarizes Mentat features significant for the comparative analysis described in this section. A more thorough review of Mentat functionality and features may be found in section 4.6.

- It is operated and used by CESNET in e-infrastructure CESNET, also has been installed and analyzed by other PROTECTIVE consortium members (GMV, PSNC and RoEduNet).
- Natively uses the selected IDEA data format.
- Mentat gathers (receives), storage, analysis, processing data and reports and visualizes them to users.
- Reports security reports directly to the responsible administrators.
- Receives data collected by Warden.
- Has a Web interface for users which allows to see the relevant data.

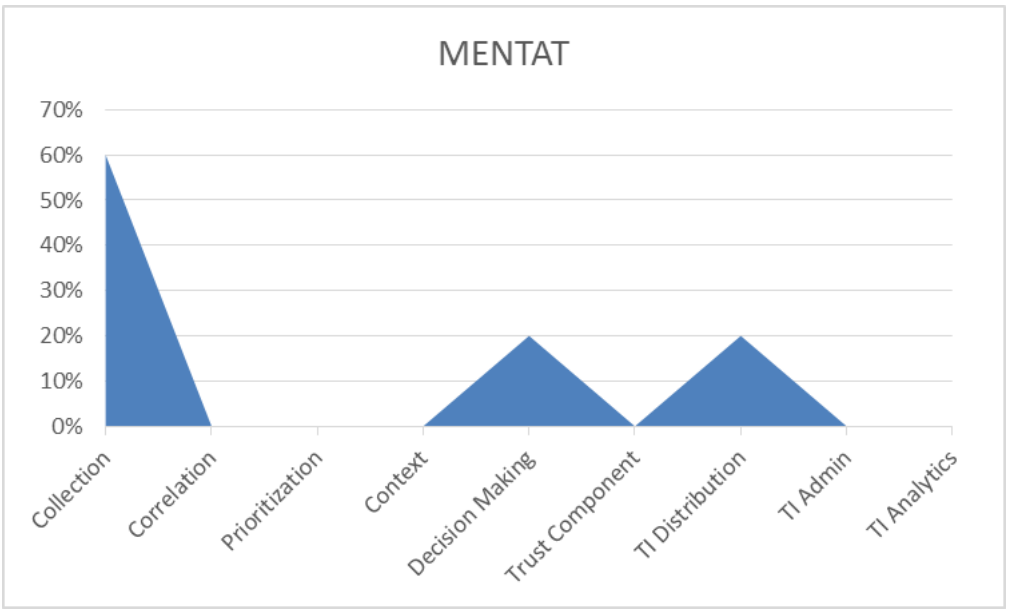


Figure 23. Mentat Coverage

In the case of comparing IntelMQ versus Mentat we can observe the following aspects (triple dash denotes a lack of or only poor coverage):

	IntelMQ	Mentat
Features	Loads from public URLs	Loads from public URLs through Warden or IntelMQ
	Collects and harmonizes feeds	Loads data from Warden
	Different result outputs (MISP, REST, MongoDB,…)	Different result outputs (MongoDB, Warden, Mentat e-mail reports).
	Customizable/Expandable	Customizable/Expandable
	---	Reads from Warden
	---	Sends reports via e-mail
	Web interface	Custom Web interface (allows to work with data/information and with reports)

Table 4. IntelMQ and Mentat features comparison

The Figure 24 shows comparison of IntelMQ and Mentat with respect to the selected criteria.

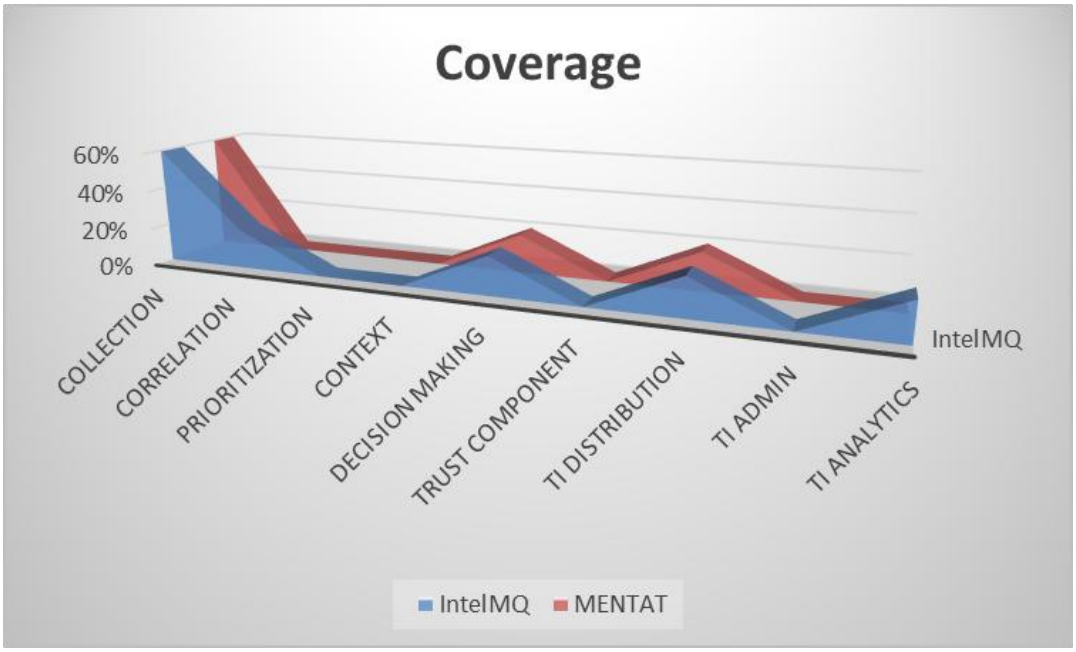


Figure 24. IntelMQ – Mentat (Comparative coverage)

5.1.3 Scenarios by Tools

The PROTECTIVE consortium performed a comprehensive analysis of the described solutions as the time constraints allowed. Particular tools have been installed by at least one consortium members in order to directly compare functionalities referring to the defined main PROTECTIVE capabilities.

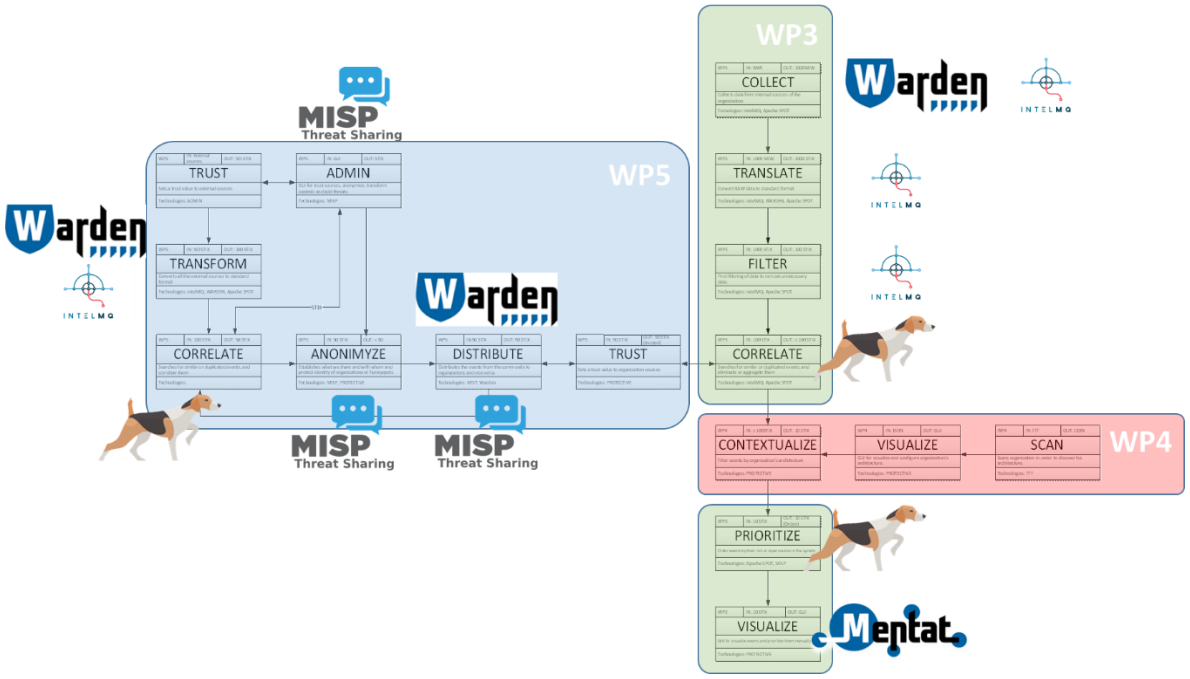


Figure 25. PROTECTIVE Capabilities by Analyzed Tools

With this big picture of how some tools can be fit into the main "primary capabilities" of PROTECTIVE, we can define different scenarios that combine the existing tools that could help to achieve the base of the PROTECTIVE Framework.

The main scenario covers Warden (with IDEA format) and Mentat. Other ones could be MISP with IntelMQ. It is true that a combination of these tools also can create other possible scenarios. Currently it seems that IntelMQ can give us support for configuring a processing pipeline but Mentat can be also applicable for this purpose.

It is true that Warden and Mentat (in their current versions) do not have today some MISP functionality that may be necessary in PROTECTIVE (for instance, creating/editing event reports or support for a more comprehensive concept of community). It will also be necessary to add extra analytics function, but to decide about the inclusion of SPOT there is a need to analyse if inclusion of Big Data is required. The definitive requirements about prioritization and correlation, and the quantity of CERTs and SMEs that will use the PROTECTIVE platform, will be the key points to decide if the PROTECTIVE Framework should include SPOT (Hadoop based solution or similar) or whether this could be an over-dimension of the solution.

5.2 The Choice of Promising Technological Components

5.2.1 Technical Selection Criteria for Base Components

In order to select existing solutions as a PROTECTIVE base, we have prepared a set of technical criteria that were taken into account for the selection. The most significant criteria are listed below and their meaning is described.

Criterion	Priority	Meaning
Applicability	Highest	The selected solution must be already applied at least for 1 year within a large production environment (at least of the size of a single NREN network or an enterprise network of a comparative size and structure) for the purposes of cyber defense.

Criterion	Priority	Meaning
Maturity	Highest	The selected solution must have proven its applicability, i.e. must be fulfilling at least the majority of the assumed requirements with acceptable quality and efficiency.
Availability	Highest	The PROTECTIVE consortium must have full access to the source code of the selected solution (e.g. it is Open Source).
License flexibility	Medium	It must be flexible to the maximum extent to modify the source code and to apply it both in non-commercial and commercial scenarios. Additionally, the selected solution must not require applying third party components that need to be purchased or have inflexible licenses.
Modularity and Extensibility	Highest	It must be easy and flexible to extend the functionality by adding consecutive modules (e.g. as microservices). Especially it must be easy to extend the selected solution with the new sources of alerts or meta-alerts. The existing modules must have standardized or at least simple and documented interfaces that the new modules might be connected to.
Access Control	High	The solution should provide fine-grained access control to the processed information, per user and per organization – or at least such access control mechanisms must be easy to be implemented.
Compatibility	Highest	The selected solution must be compatible with the selected information exchange format. The selected solution data exchange format must support transmission at least of alerts[46], meta-alerts[47], static information.
Scalability	Highest	The selected solution must be easily scalable in order to be applied in networks of different sizes. Namely the selected solution must allow for multiple (parallel, serial, hierarchical) integration of individual components and their interconnection into one globally communicating (sharing) system.
Automation level	Highest	The selected solution must be oriented towards automating the information processing. It must allow integration into relating processes within e.g. CSIRT team (such as incident handling process).
Processing capabilities	Highest	The selected solution must facilitate both online (streaming) data processing as well as offline processing of the data (either raw or pre-processed/filtered) stored persistently.

Table 5. Technical selection criteria for PROTECTIVE base

5.2.2 Mapping Criteria onto PROTECTIVE Objectives

We have mapped the aforementioned criteria onto the relevant PROTECTIVE objectives (Technical Objectives and Project Objectives). Following the PROTECTIVE DoA, the project objectives are as follows:

[46] According to the "WP3 terminology dictionary" interim document (which will be also a part of the D3.1), alert is a notification that a specific attack has been directed at an organization's information systems, with a known structure, composed of a list of attribute-value pairs.

[47] Following the same document, meta-alert is a result of the merging of two or more related alerts (or other meta-alerts), merged as a part of the alert correlation process.

Objective	Abbreviation	Meaning
Main	n.a.	PROTECTIVE will develop a comprehensive solution to raise organisational cyber situational awareness (CSA) through: • enhancement of security alert correlation and prioritisation • establishment of the relevance/criticality of an organization's assets to its business/mission • establishment of a threat intelligence sharing community
Technical Objectives	TO1	To enhance security monitoring through improved incident correlation and prioritisation
	TO2	To establish the criticality of an organisations assets to its mission
	TO3	To develop tools and techniques to establish community threat intelligence sharing
	TO4	To improve trust in the data quality of shared intelligence
	TO5	To develop software framework to support information flow processing
Project Objectives	PO1	To demonstrate the operational efficiency of the developed solution in an NREN environment
	PO2	To evaluate the developed solutions to support SME security management

Table 6. PROTECTIVE objectives

The selected criteria are focused on maximizing functionality and extensibility of the software composing base for the PROTECTIVE framework. The selection process aims to choose the proper starting components that could be further extended, without a hassle, also with bespoke PROTECTIVE software.

Without a stable software base with operationally proven usability, it will be impossible to provide the whole PROTECTIVE suite, especially in the context of reaching the following project objectives: TO1, TO3 and TO5, as well as both Pilot Objectives (generally speaking, to be able to provide and successfully evaluate NREN and SME pilot installations).

In specified cases the criteria might be additionally mapped to particular objectives:

- **Access Control** - facilitates the TO3 as the communities will reject sharing information through a system not providing sufficient security measures and could e.g. cause data leakage. To some extent it also supports TO4, preventing from unauthorized provision of TI feeds. Such a malicious provision of false alerts might lead e.g. to hide actual attacks or just to cause DoS against the operator.
- **Processing capabilities and Scalability** supports TO1 where increased capabilities of tools are required in order to be able to process all the relevant information, either from own security systems of the PROTECTIVE suite users or acquired through TI sharing modules. **Maturity** criterion also indirectly supports TO1.
- **Automation level** also supports the objective TO1 due to optimizing the work of CERT/CSIRT operators.
- **License flexibility and Modularity** as well as **Extensibility** directly facilitate TO5 with special emphasis put on "framework" term which requires ability to extend the PROTECTIVE base

with additional modules (also after the project finishes, e.g. as a reaction on users' demand or modified attack trends). Moreover, PO2 concerning the project SME installation and possible further SME deployments may require particularly flexible licence, e.g. allowing to close certain specified parts of the source code.

- On the other hand, ability to provide specific modules as open source (**Licence flexibility** criterion) supports the TO3 and TO4 as communities may analyze the source code of the provided solutions to see how they work, that there are no backdoors (increasing trust to the provided solutions), or propose further optimizations.
- **Compatibility** supports also particular TOs and POs concerning enhancements in security monitoring (TO1) and TI sharing (TO3, TO4) as it aims at extending capabilities of TI sharing functionality due to integrating information from more sources(sensors). Compatibility with a broad set of input sources will also increase the trust level of for particular alerts (or meta-alerts), as specified events may be confirmed by multiple, independent sensors.

5.2.3 Selection Summary

After the investigation of solutions available on the market, we decided to choose **Warden** and **Mentat**, developed by CESNET, as the core set of tools for the implementation of the first proof of concept of the system. The tools are described in more details in sections 4.5 and 4.6, respectively.

Warden as a sharing platform facilitates information sharing and covers to large extent functionalities such as data collection, data sharing, normalization through the usage of IDEA format and storing events in the database.

Mentat adds functionalities such as filtering, data enrichment through collecting WARDEN were chosen especially wrt to data from external sources, reporting engine, limited analytical features (e.g. statistics), document store and graphical user interface.

What is important to note is that Mentat is already a modularized system and new components may be added to the processing workflow relatively easy. Certain functionalities, fulfilling particular PROTECTIVE objectives, will be implemented as PROTECTIVE bespoke software and may be easily integrated into the processing pipeline containing Warden and Mentat.

From an architectural point of view, the best possible flexbility will be achieved through the deployment of multiple instances of Warden and Mentat. Such an approach will allow for flexible sharing of data in peer to peer mode as well as through centralized shared community servers.

5.2.4 Selection Justification

This section provides a summary of reasons why Warden and Mentat were selected as a base for PROTECTIVE framework as well as motivation to reject other solutions.

Apache SPOT was rejected for two main reasons:

- It has been designed mainly to processing of network flow and packet data to detect anomalies rather than processing and sharing the threat information,
- It is not mature enough yet and the PROTECTION consortium has not been aware of its production-grade usage in any organization (thus it does not meet applicability and maturity criteria).

The Hive has also been rejected as it has a different goal than is required in PROTECTIVE – It is designed mostly to help security specialists with investigating security incidents, not for automated TI processing. Also, its license is a problem, as Affero GPL requires to publish source code of any Web application based on the licensed work, which prevents its incorporation into commercial MSSP systems.

Therefore the remaining options were:

- Warden+Mentat,
- IntelMQ,
- MISP.

In comparison to IntelMQ, Warden has an advantage because IntelMQ is designed to fetch external data feeds only, while Warden is used to transfer (share) data, from internal detectors (security tools such as IDS, honeypots, IPS, network probes, logs, firewalls etc.) as well as external feeds, acting as a sharing server. In fact, there exists an IntelMQ to Warden converter, which uses IntelMQ to fetch external data feeds and converts them to IDEA format and sends them to Warden. Similarly, it is possible to implement conversion of MISP events to IDEA messages and send them automatically to Warden. The opposite direction of conversion may be problematic because:

- MISP uses a simpler data format without particular structure (actually a list of indicators), therefore potentially important information would be lost during mapping the formats,
- Warden is designed to handle a large number (at least millions) of simple events per day (although it can process more high-level data as well), while MISP is designed rather for sharing results of malware analysis which are usually in the order of tens per day only.

Therefore Warden can ingest data from both IntelMQ and MISP, which makes it a better choice for the PROTECTIVE base. It is true that MISP has significantly more advanced access control mechanisms built-in, but similar functionality can be achieved by combination of several pairs of Warden and Mentat, or, if necessary, similar access control mechanism can be implemented into Warden relatively easily.

Mentat is a natural choice to complement Warden with processing capabilities, since they both are designed to work together and use the same data format. Moreover, in comparison with the other tools, Mentat has currently higher processing capabilities. Although IntelMQ and MISP have certain capabilities in correlation and enrichment of data, they focus mostly on data gathering and sharing, respectively, not to data processing or reporting. Also, the modular architecture of Mentat allows to easily implement completely new types of functionality, which is very important for PROTECTIVE since none of the compared solutions currently provides all the functions required.

Mentat and IntelMQ both provide a basis for implementing an information processing workflow backplane. Either could function as such a backplane – however Menat has a number of advantages because of its close integration with Warden and the fact that CESNET consortium member.

The other advantage of Warden and Mentat systems is using IDEA data format. Using that format makes sharing platform based on Warden and Mentat extremely robust and flexible. IDEA data format together with Mentat and Warden functionalities allows to recognize and work with several types of the data – alerts (events produced by security tools, called also *primary data*), meta-alerts (results of intelligent analysis and correlation), static information such as “entity X is in the blacklist Y” (received usually from other sharing systems or data collectors). This is very important aspect as PROTECTIVE is expected to work with many heterogeneous data and information in order to create TI information.

Warden and Mentat allow to deploy multiple instances, in a parallel or hierarchical structure, in peer to peer mode as well as through centralized shared community servers, which is very flexible from the PROTECTIVE point of view and allows to create a real sharing-infrastructure across Europe.

The last, but also important, reason to select Warden and Mentat is the fact that one of the consortium members (CESNET) uses both systems for several years and have a good experience with

them. Current Warden and Mentat instance operated by CESNET in CESNET e-infrastructure has the following capabilities:

- Receives and processes 2 millions events per day,
- Events are sent from security tools placed in CESNET e-infrastructure,
- Serves approximately 330 institutions connected to CESNET e-infrastructure.

Also, there is a number of data sources already contributing to CESNET's Warden which can be immediately used in PROTECTIVE.

Both Warden and Mentat system have been developed with respect to security issues, therefore access to data is not public, but restricted only to recognized and validated partners. Access control is based on Public Key Infrastructure. Additional required functionalities such as risk assessment, assets database including assets state, trust related functionalities will be implemented as additional services within the PROTECTIVE project. Due to the modularity of Warden and Mentat, their integration will be easy.

5.2.5 Reference of Warden and Mentat to Selection Criteria

Table 7 provides more direct references to the selection criteria in the context of Warden and Mentat.

Criterion	Reference to Warden and Mentat
Applicability, Maturity	CESNET started to operate Warden system in 2012 and Mentat system in 2014. Both Warden and Mentat systems serve ca. 300 organizations connected to CESNET infrastructure, which provides network connectivity and other services to ~500000 users.
Availability, Licence Flexibility	Both Warden and Mentat systems are being developed as OpenSource projects. In case of Warden system there is a direct access (read only) to its repository granted for all PROTECTIVE partners and tarball with all source code for full installation of Warden server and client library. The .tgz files are available at Warden web page: https://warden.cesnet.cz/en/downloads In case of Mentat system there is a direct access (read only) to its repository granted for all PROTECTIVE partners. Mentat will become a open project at the end of August 2017, after certain major changes in source code are finished.
Modularity and Extensibility	From the flexibility point of view, Mentat is more important to be flexible and modular, as Warden is only a transmission system. Mentat is designed as a distributed modular system with the emphasis on security, extensibility and scalability. The core of the system is implemented similarly to the Postfix MTA. It consists of many simple modules (working as daemons) and uses filesystem directory message queues. Each module is responsible for performing a particular 'simple' task. Mentat distinguished three types of modules: • real-time event processing modules such as „mentat-inspector", „mentat-enrichment" and „mentat-storage" • post processing modules (via database) such as „mentat-reporter", „mentat-statistician" • control modules and user interfaces.

Criterion	Reference to Warden and Mentat
	This approach to Mentat internal architecture enables smooth parallelization and extensibility. A new module can be implemented and hooked into processing workflow relatively easy, some of the modules can be parallelized, several instances of one module can be run simultaneously etc.
Access Control	The current Mentat system has an AAI layer to control access to data and other GUI functionalities. Warden system has implemented mechanisms of control over sending and receiving clients. Each sending as well as receiving client must be registered in Warden system and each client must have its own certificate which is used for authentication. Therefore Warden server (its operator) has full control over data sources and consumers and knows their identity. Both AA mechanisms in Warden and Mentat are easily expandable.
Compatibility	IDEA data format was selected in PROTECTIVE. Both Warden and Mentat system use this data format.
Scalability	Current Warden and Mentat instances operating in CESNET e-infrastructure collect and process 2 millions events per day and send reports to ca. 330 organizations connected to CESNET e-infrastructure. That e-infrastructure has got assigned the AS2852 number[48], which consists of almost 1 million of IP addresses. Certain values depicting the current scale of Warden status may be seen on the main Warden Web page[49]. The values are updated every hour. As for the time of writing this document, Warden stores over 50 millions of events[50]. Warden and Mentat systems are able to operate as a multi-instance system (both parallel and hierarchical). There may exist configurations with several Warden server sharing data peer-to-peer, as well as configuration of a “central” Warden carrying selected data, or combination of both. Both models are possible, because IDEA data format allows to mark “data source”, that allows for distinguishing who or what produced data and how they get into sharing infrastructure. Therefore it is possible to ensure that data are not cycling in the sharing system forever. Using the IDEA data format makes sharing platform based on Warden and Mentat extremely robust and flexible.
Automation level	The current Mentat system operating in CESNET e-infrastructure contains the “reporting” module, which sends gathered data related to one network to its abuse contact (network operators) via a human readable e-mails. There also exists a client for creating tickets in RT system. Both Mentat and Warden system are flexible enough to allow different approaches to processing and visualizing data – e-mail, tickets, Web GUI.

[48] https://www.tcpiputils.com/browse/as/2852
[49] https://warden.cesnet.cz/en/index
[50] Please note that the event in this case is considered as an information about a particular, recognized security issue – not as an event from the operating system point of view (like access to a filesystem object or successful authentication). The number of latter would be significantly larger, but PROTECTIVE does not refer events on that level.

Criterion	Reference to Warden and Mentat
Processing capabilities	Both Warden and Mentat system meet this criterion.

Table 7. Summary of filling selection criteria by Warden and Mentat

5.3 PROTECTIVE Roadmap

The first version of the PROTECTIVE framework will be therefore based on Warden and Mentat, including possible minor improvements and extensions – with the main goal to actually proving accurateness and applicability of the constructed pipeline. On the other hand, it will allow to detail the gaps necessary to be filled with bespoke software. Particular scouted technologies will be applied to fill these gaps in Phase 2 and Phase 3 of PROTECTIVE.

The already identified gaps include facilities needed for the enrichment phase including operational context (e.g. risk assessment, vulnerabilities and applied patches), prioritization and ranking, the inference of TI by means of analysis modules.

For the enrichment phase Mentat provides only basic capabilities that need to be extended. Namely, engines harvesting additional information from external sources relevant to particular alert type should be deployed. Moreover, facilities to handle asset criticality information, along with information about vulnerabilities are also needed and will be implemented within PROTECTIVE. Suitable candidates as the supportive tool could be in that case Cortex engine that is part of TheHive (see section 4.1.2), and IntelMQ collectors and parsers (see chapter 0). Both solutions can harvest data from a variety of already available sources.

The proposed base of technology does not provide advanced prioritization and ranking capabilities (besides simple sorting or filtering) that could handle multiple criteria, represent preferences of the decision maker (user of the system) and use them to order meta-alerts in the most efficient way. Those parts will be implemented within PROTECTIVE on the basis of available libraries for multicriteria decision aiding and preference learning (e.g. jRank – see section 5.3.5).

Warden provides the data exchange layer including authentication and authorization. However, it provides only a share to all type of communication channel. Thus it is necessary to provide additional distribution layer on top of Warden instances to enable deployment of more sophisticated sharing scenarios. This top layer could be implemented as new Mentat module.

The TI inference process produces a concise form of information as a result of wide range of analysis. Thus, the analytical pipelines must be configurable to allow rearrangement of the processing task when such a need arises. For example, an analyst might need to conduct a full-text search using particular sequences of bytes or keywords. Thus it would be justified to apply Elastic Stack (formerly known as ELK Stack) in parallel to base processing flow or instead of some Mentat modules. Other analysts might need to use machine learning methods to identify reoccurring patterns in sequences of alerts, so it would be useful to them to be able to include Apache Spot (see section 4.4) into the processing pipeline or export data to it.

Summarizing the aforementioned PROTECTIVE roadmap, we have identified that the framework technology base has to be complemented with additional elements addressing single particular PROTECTIVE pipeline stages. These elements will be:

- Existing solutions, e.g. a database engine most feasible to store the relevant types of information;

- Bespoke software developed within the project, e.g. additional connectors facilitating collecting information from security systems (e.g. SIEM or NGFW), translating it to the IDEA format and sending to Warden.

The following sections provide the descriptions of the most relevant additional technologies significant for PROTECTIVE. Descriptions of additional technologies, considered as less crucial, are provided in Annex C: Additional Technology Scouting.

5.3.1 Existing Supplementary Tools

This section presents a set of tools, either third party or developed by the PROTECTIVE consortium members, that will be used to support particular stages of the PROTECTIVE pipeline.

5.3.1.1 Cyber Tool

Cyber Tool (CT) has been developed by ITTI, PROTECTIVE consortium member. It addresses the crucial role of information and knowledge superiority in the continuous process of preparedness and cyber situational awareness. Cyber security issues are a powerful and evolving asymmetric threat. Each day about 30 new vulnerabilities are discovered which summarizes to ca. 10,000 new vulnerabilities a year[51]. There are currently multiple tools for improving network security. Still it is common to overlook the fact that similar network configurations differ significantly. Moreover, tools that do not include expert knowledge can easily omit information crucial to the security of particular network. Cyber Tool moves toward solution to this problem, summarizing the vulnerabilities of the system found (e.g. by audit tools) and attempts to provide information on what problems to tackle first. In the prototype version Cyber Tool also retains and abstracts the expert knowledge among multiple networks and domains, and thus has a great potential for future use in the military domain.

Cyber Tool was developed as prototype software designed to perform security analysis of C4ISR systems as well as SCADA systems. Currently CT allows users to incorporate expert knowledge, for example, in the course of the audit. By utilizing the experts' knowledge Cyber Tool provides the means to retain essential knowledge.

As an input Cyber Tool takes data about the network and its assets. The typical result of applying Cyber Tool computations is a security report with ranking. The input information required by Cyber Tool is:

- **Database of vulnerabilities** – threats or attacks that exploit these vulnerabilities, and safeguard that reduces some threats. Currently CT reads the ontology from an OWL (Web Ontology Language) file,
- **Topology** – describes interconnections between asset instances and is required to perform successful vulnerability identification,
- **Expert rules** – provide the tool with additional expert knowledge.

In the prototype version Cyber Tool provided a proof-of-concept. Web-based GUI dedicated to support evaluating the security level of civil/military IT systems (especially providing functionalities of identifying the network topology, threats and vulnerabilities as well as creating a security report). In case of PROTECTIVE however it is most probable that CT will not be providing a dedicated user interface, instead working as a service. In such a case, only data exchange occurs and the GUI does not need to be used.

[51] http://www.cvedetails.com/browse-by-date.php; in 2016 there were 6435 vulnerabilities with CVE number assigned; however between January and June 17, 2017 the relevant number already exceeded that value (and reached 6722)

A more thorough information about CT may be found in PROTECTIVE D4.1. Although CT is being developed as a proprietary, closed-source solution, a free-of-charge licence will be granted to the PROTECTIVE partners.

5.3.1.2 CertainTrust

The trust mechanism in PROTECTIVE aims to improve the management, sharing, and prioritisation of TI within the community of NRENs and SMEs by determining the quality (or reputation) of TI feeds.

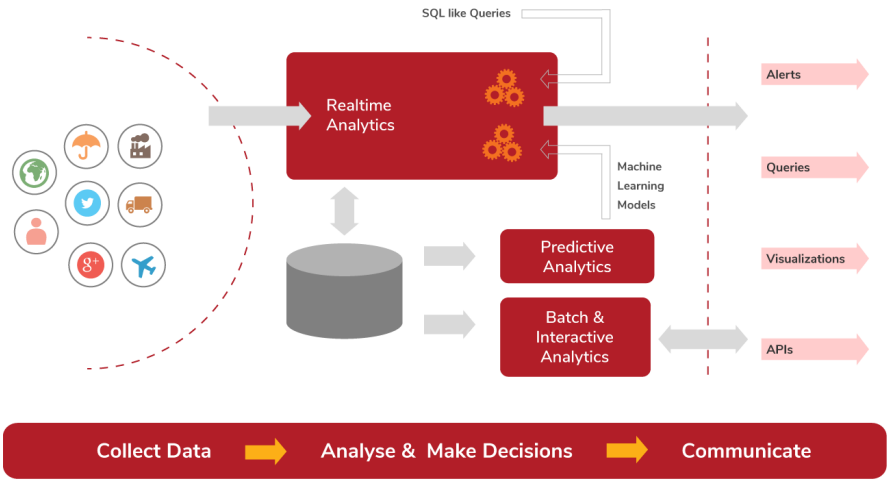
In PROTECTIVE's system, computational trust methods (CertainTrust and CertainLogic)(Habib, Volk, Hauke, & Mühlhäuser, 2015) will be used, designed by one of the project consortium partners (TUDA), to assess the TI feeds quality. CertainTrust provides a mechanism to quantify the level of confidence (certainty) along with the quality of TI feeds. The quality assessment of the TI feeds are based on multiple aspects or dimensions such as accuracy and timeliness of the feeds. CertainLogic has the ability to compute and visualise quality of the TI feeds and associated confidence level based on multiple aspects. In PROTECTIVE, we aim to develop a usable feedback mechanism leveraging the visualisation module of CertainTrust.

CertainTrust Software Development Kit (SDK), consisting of CertainTrust and CertainLogic methods, will be instantiated and extended to deal with different types of properties. The SDK provides various functions to aggregate multiple properties to generate an overall reputation score. These functions will be leveraged to generate quality (reputation) of the TI feeds as well as reputation scores of the malicious entities. The SDK also provides visualisation function to communicate multi-dimensional reputation score to security analysts in NRENs and SMEs. Security analysts can use the visualisation to provide feedback on the data quality or reputation scores.

A more detailed information about CertainTrust has been put into PROTECTIVE D5.1.

5.3.1.3 WSO2 Data Analytic Server (DAS)

WSO2 Data Analytics Server (DAS)[52] is a robust open source analytics platform for realtime streaming data analysis, offering also CEP (Complex Event Processing) 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.



[52] http://wso2.com/smart-analytics

Figure 26. WSO2 DAS Architecture[53]

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 realtime, 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 (Frankowski, et al., 2015).

5.3.2 PROTECTIVE Bespoke Extensions to Existing Tools

In this part we present the most significant tools and frameworks which define the PROTECTIVE modules design and development from the very decision to utilize them.

5.3.2.1 µServices - Spring Cloud & Netflix OSS Based

As PROTECTIVE's bespoke modules will be designed and developed to feature easy distribution, resilience and horizontal scalability as microservices, the targeted runtime environment also has to deal with such features.

Netflix OSS

As advanced and easy to use deployment and management tools for microservices architecture are not part of the project, we need to acquire such tools that will allow us to use them in plug and play approach. The Netflix Open Source Software (OSS) group provides a set of open source projects that deal with these runtime requirements. Amongst others, the following projects may be relevant for PROTECTIVE:

- The Hystrix[54] middleware abstracts tight service dependencies within a distributed environment by isolating the (remote) access points. The external service calls are wrapped in a Hystrix command object where service failures can be detected and managed easily by avoiding cascading failures and fallback logic options;
- Eureka[55] is a RESTful service for managing and locating services within a distributed environment. Eureka provides advanced load balancing (basic round-robin strategy) and failover features for middle tier servers;
- Feign[56] eases the creation and handling of text-based HTTP API calls by introducing templates based on code annotations. The JAVA library seamlessly integrates the Netflix Denominator[57] library for managing DNS cloud environments and additionally reduces the complexity within a distributed environment. In PROTECTIVE this library could reduce the effort of creating Machine-to-machine communication over REST by just declaring the API.

[53] https://docs.wso2.com/display/DAS310/Introducing+DAS
[54] https://github.com/Netflix/Hystrix
[55] https://github.com/Netflix/eureka
[56] https://github.com/OpenFeign/feign
[57] https://github.com/Netflix/Denominator

- Ribbon[58] abstracts the creation of REST calls by adding template builder functionalities with built-in client-side fallback hooks and response validation schemes. Ribbon also supports multiple endpoints and protocols such as HTTP, TCP or UDP. Ribbon could be used in PROTECTIVE to facilitate scaling and resilient communication behaviour without any other dedicated load balancing service;
- Zuul[59] provides a gateway service with dynamic routing and monitoring capabilities. symbIoTe could use Zuul as central gate keeper for all requests where several cross cutting concerns such as monitoring, logging or security aspects (e.g. anomaly detection, authentication) could be handled.

Spring Cloud

By agreeing that PROTECTIVE modules will adhere to the microservices architecture, the systems need support for management, configuration and DevOps analysis tools e.g. provided as Netflix OSS projects. Spring Cloud offers seamless integration of its own solutions from the Spring stack (e.g. Framework & Boot) with those tools. It is a complete solution, which provides features of modern software on all layers:

- starting with the lowest level of e.g. inversion-of-control principle for objects lifecycle management;
- building web services upon that core with Spring Framework;
- simplifying development, configuration and deployment of one's code with Spring Boot;
- and finally by adapting those solutions to self-route in distributed cloud environments.

As such, most of the components released by Netflix under OSS project are already integrated with Spring Boot to provide a scalable and configurable microservices-powered system and offered as Spring Cloud.

We will now mention a few features from Spring Cloud which empower this cloud-apps building suite.

Spring Framework

The Spring Framework[60] is the leading Open Source Java platform for the development of enterprise ready Java applications. Among others, it provides the tools and helpers needed to:

- Isolate deployment issues between application servers, making the application deployable in a wide array of vendor solutions both proprietary and Open Source
- Provide tools to implement different programming patterns like, e.g. Singleton or Publish-Subscribe
- Isolate and standardize access to different data sources, be it local or remote
- Provide a powerful framework for bean registration and dependency injection
- Implement mechanisms for Aspect Oriented Programming

As a basic framework, it is not tied to any concrete module of PROTECTIVE in particular. Its role will be to be the foundation upon which the different bespoke components extending the Warden & Mentat pipeline to our need in Java will be built, providing the tools and helpers needed to forget about deployment and wiring issues and allow PROTECTIVE developers to concentrate on business logic.

[58] https://github.com/Netflix/ribbon
[59] https://github.com/Netflix/zuul
[60] https://projects.spring.io/spring-framework/

Spring Boot

Spring Boot[61] is an entrypoint for Spring applications developers and manages all the mundane Spring Framework configuration and provision task. It basically simplifies the Spring-based applications stubs creation. It does so by providing a default configuration, from which the project can be expanded. It requires no XML configuration, one of the biggest powers and simultaneously curse of the Spring Framework. Spring Boot also provides non-functional features that are common in this kind of large projects, such as embedded servers, metrics, logging, security, health checks and others.

5.3.2.2 Distributed Streaming Platform

Alternatively to Spring Cloud other frameworks might be applied, dedicated to building distributed streaming applications. For those, the PROTECTIVE consortium has identified the following candidates:

- **Apache Samza**[62] is a distributed stream processing framework. It uses Apache Kafka for messaging, and Apache Hadoop YARN to provide fault tolerance, processor isolation, security, and resource management.
- **Apache Storm**[63] is similar to Samza although may use some other terms in order to describe similar concepts and definitions. A comparison may be found under a dedicated URL[64]
- **Apache Kafka**[65] is another distributed streaming platform especially usable in the scenarios of building real-time streaming data pipelines that reliably get data between systems or applications, and building real-time streaming applications that transform or react to the streams of data. Kafka is used in the Open Network Insight[66] – solution for packet and flow analytics on Hadoop. Kafka may be integrated with Spring through Reactor API[67].

PROTECTIVE development was also examined against utilizing **actor based** processing which e.g. **Akka**[68] offers. It is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM.

Akka has been designed to ease writing correct distributed, concurrent, fault-tolerant and scalable applications. In order to do that, the Akka designers proposed the Actor Model, thus raising the abstraction level and provide a better platform to build scalable, resilient and responsive applications. The concept of actor alleviates the developer from having to deal with explicit locking and thread management. Actors also provide the abstraction for transparent distribution and the basis for truly scalable and fault-tolerant applications. Actor systems can span over multiple JVMs to provide truly fault-tolerant systems.

For fault-tolerance the "let it crash" model (basing on terminating a process in case of error and suitable reporting it for the monitoring processes to let them intelligently restart the process or undertake any other relevant action) has been adopted which the telecom industry has used with great success to build applications that have the capabilities of self-healing and systems that never stop.

[61] http://projects.spring.io/spring-boot/
[62] http://samza.apache.org/
[63] http://storm.apache.org/, earlier http://www.storm-project.net/
[64] http://samza.apache.org/learn/documentation/0.10/comparisons/storm.html
[65] https://kafka.apache.org/
[66] https://github.com/Open-Network-Insight
[67] https://spring.io/blog/2016/12/15/reactor-kafka-1-0-0-m1-released
[68] http://akka.io/

Akka is Open Source and available under the Apache 2 License. It also supports cluster mode. Applications for Akka may be written in Java, Scala or using .NET Framework / Mono.

However, eventually PROTECTIVE consortium agreed that we will focus on domain driven development, create groups of µServices that share a common bounding context and therefore manually design the flow of information either within those groups and across the PROTECTIVE data pipeline.

5.3.3 Databases

Choosing a proper DBMS is a non-trivial task. In general, two approaches are possible – to select a single DB suitable enough for different purposes or to use multiple DB engines, optimized each for particular tasks.

It is possible to go with a single database - e.g. orientDB, which supports both documents and graphs but introduces performance pitfalls. On the other hand, using a single DB would reduce administrative overhead to prepare and maintain the analytic environment.

However, thanks to the decision that PROTECTIVE will be striving to provide µServices (microservices) architecture, this very problem can be relayed to the developers of particular modules themselves. Typical for µServices is data redundancy which means that developers provide each service with its own best fitting database engine. We envisage that in the final version of the PROTECTIVE, in order to increase the computational facilities of the provided system, we will use multiple engines although in the first pilot stage we may use a single DB in order to make a proof-of-concept solution. Regardless on the final decision, we have analyzed a number of engines in order to recognize their properties and applicability to PROTECTIVE as exactly as possible within the assumed timeframe.

Spring Cloud acting as PROTECTIVE bespoke modules may be relatively easy integrated at least to certain degree with several database engines that we consider to use:

- MongoDB[69] – open, non-relational, document database engine. MongoDB stores data in flexible, JSON-like documents, and the meaning fields can vary from document to document. Data structure can be changed over time as well. Ad hoc queries, indexing, and real time aggregation facilities are provided out of the box. MongoDB is free and open-source, published under the GNU Affero General Public License which does not cause problems if the software is going to be used “as is” like in this case (for the details, see section 4.3.2). CESNET has been using MongoDB in their projects related with TI concept. For higher loads, however, performance problems tend to occur.
- DSE Cassandra[70] – DataStax Enterprise (DSE) provides a comprehensive and operationally simple data management layer with a unique always-on architecture built on Apache Cassandra. DSE Cassandra provides scalability and clustering facilities out of the box, but is paid software. Only free trial is available for 6 months’ period of non-production use.
- Neo4j [71] is a highly scalable, native graph database purpose-built to leverage data but also its relationships. Neo4j's native graph storage and processing engine delivers constant, real-time performance. The Neo4j Community Edition is licensed under the free GNU General Public License (GPL) v3 while the Neo4j Enterprise Edition is dual licensed under Neo4j commercial license as well as under the free Affero General Public License (AGPL) v3. Neo4j provides their own query language – Cypher.

[69] https://www.mongodb.com/
[70] https://www.datastax.com/products/datastax-enterprise
[71] https://neo4j.com/

- OrientDB[72] is a distributed multi-model NoSQL with a graph database engine. It combines distributed engine with a in one product. OrientDB incorporates a search engine, and concepts along with a reactive model (with) and . The Community Edition is released under the Open Source, commercial friendly. The vendor claims that security, operations, APIs, performance and stability are advantages of OrientDB, as well as declares advantages over MongoDB and Neo4j. For instance, comparing OrientDB and Neo4j Community Editions, OrientDB is reported to have broader set of features and capabilities, better performance, more friendly licensing model, better scalability and more simple query language (SQL vs. Cypher).
- Elastic[73] is the rebranded ELK stack – Elastic, Logstash, Kibana – a set of acclaimed event data logging, processing and visualizing tools. PROTECTIVE consortium members already use it in their proprietary software and therefore might utilize that expertise to empower the bespoke solutions PROTECTIVE will consist of with Elastic backend to achieve optimal implementation and deployment.

5.3.4 Communication

PROTECTIVE concerns processing and exchanging data between multiple communities in a distributed manner, using heterogeneous systems and solutions. Therefore the consortium has explored a variety of communication paradigms and frameworks. These are briefly described in the following sections.

5.3.4.1 Messaging Queues

Following Wikipedia, message queue is a software-engineering component used for inter-process communication, or for inter-thread communication within the same process. They use a queue for messaging – the passing of control or of content. The message queue paradigm is a sibling of the publisher/subscriber pattern, and is typically one part of a larger message-oriented middleware system.

We have distinguished the following MQ solutions potentially the most useful for PROTECTIVE:

RabbitMQ:

RabbitMQ[74] is claimed to be the most widely deployed open source message broker with more than 35,000 production deployments both at small startups and large enterprises. It is lightweight and easy to deploy on premise and in the cloud. RabbitMQ supports multiple messaging protocols (AMQP, STOMP, MQTT, AMQP 1.0 as well as HTTP protocol) and can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements. RabbitMQ runs on many operating systems and cloud environments, and provides a wide range of developer tools for most popular programming languages (Java and other JVM languages, .NET, Ruby, Python, PHP, and many more). Official support covers the following operating systems: Linux, Windows, NT through 10, Windows Server 2003/2008/2012, Mac OS X, Solaris, FreeBSD, TRU64, VxWorks. Integration with SpringCloud is possible.

RabbitMQ implements a broker architecture, meaning that messages are queued on a central node before being sent to clients. Thus RabbitMQ is very easy to use and deploy as advanced scenarios like routing, load balancing or persistent message queuing are supported in just a few lines of code. However, it also makes it less scalable and potentially more prone to overload as the central node adds latency and message envelopes are relatively big.

[72] http://orientdb.com/
[73] https://www.elastic.co/
[74] https://www.rabbitmq.com/

ZeroMQ

ZeroMQ[75] (ZMQ, ØMQ) is a very lightweight messaging system specially designed for high throughput/low latency scenarios. It looks like an embeddable networking library but acts like a concurrency framework. It provides sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast. Sockets may be connected N-to-N with patterns like fan-out, pub-sub, task distribution, and request-reply. It is fast enough to be the fabric for clustered products. Its asynchronous I/O model allows to provide scalable multicore applications, built as asynchronous message-processing tasks. It has a score of language APIs (built on top of native C API) and runs on most operating systems.

ZeroMQ can be deployed both with broker and PHP technologies and supports many advanced messaging scenarios but contrary to RabbitMQ, implementing most of them requires combining various pieces of the framework (e.g. sockets and devices). ZeroMQ is very flexible but more background is necessary in order to start using it efficiently.

ØMQ is free software licensed under the LGPL (support is commercial).

5.3.4.2 Polyglot Communication - gRPC

PROTECTIVE stands in a situation where different modules developed in heterogeneous programming languages are either used or implemented, there is a need to introduce a broker tool to efficiently communicate them. For that we have selected gRPC[76]. It is a modern open source high performance RPC framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

The main usage scenarios cover e.g. Low latency, highly scalable, distributed systems; developing mobile clients which are communicating to a cloud server; designing a new protocol that needs to be accurate, efficient and language independent; layered design to enable extensions like authentication, load balancing, logging and monitoring etc.

By providing Interface Definition Languages (IDLs) it is possible to assure cross communication with C++, Java, Python, Go, Ruby, C#, Node.js, Android Java, Objective C and PHP. Another means of support are also available for other programming languages, e.g. Perl.

gRPC is available under the 3-clause BSD license.

5.3.5 Multiple-criteria Decision-making (MCDM) Tools

MCDM tools will be necessary in order to support prioritization of processed alerts and meta-alerts. In the DoA the jRank tool was mentioned as the most apparent candidate to be used.

- jRank[77] is a decision support tool for dealing with multi-criteria choice and ranking problems. It is a command-line Java application, based on java Rough Sets (jRS) library (http://www.cs.put.poznan.pl/mszelag/Software/software.html java Rough Sets), which implements methods of data analysis provided by the Dominance-based Rough Set Approach and Variable Consistency Dominance-based Rough Set Approaches. The tool is

[75] http://zeromq.org/
[76] http://www.grpc.io/
[77] http://www.cs.put.poznan.pl/mszelag/Software/jRank/jRank.html

available under the PUT IDSS license[78] stating that the software may be used for research, education, development and private, non-profit purposes.

Some other Multi Criteria Decision Aid (MCDA) tools are mentioned also here:

- A set of tools provided on the webpage of EURO Working Group on Multicriteria Decision Aiding[79], including e.g. diviz, D-Sight, MCDA-ULaval, jMAF and ca. 30 other tools
- Decision Deck[80], also listed on the aforementioned Web page. The project develops Open Source software tools to support MCDA process, intended to be interoperable in order to create a coherent ecosystem. Decision Deck supports XMCDA (a data standard which allows to represent MCDA data elements in XML according to a clearly defined grammar) with Java, Python and R reference libraries.

5.3.6 Data Stream Mining and Machine Learning

For any kind of data stream mining and machine learning (ML) the following solutions have been identified to be used.

Apache Spark[81] – fast and general engine for large-scale data processing. It is claimed to run applications up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing. Apache Spark MLib[82] – scalable machine learning library for Apache Spark, usable in Java, Scala, Python and R, able to run on existing Hadoop clusters to maximize performance. MLib implements a broad set ML algorithms including classification and recommendation algorithms, decision trees, clustering etc.

Spark may be run using its standalone cluster mode, on EC2, on Hadoop YARN, or on Apache Mesos. It is able to access data in HDFS, Cassandra, HBase, Hive, Tachyon, and any Hadoop data source. Custom data sources may also be defined. Applications for Spark, including interactive shell programs, may be built in Java, Scala, Python and R programming languages. Spark may combine SQL, streaming and other complex analytic engines as the same application may seamlessly combine e.g. SQL and DataFrames (for SQL), MLib (for machine learning), GraphX (for graph analysis) and Spark Streaming (for stream processing) libraries:

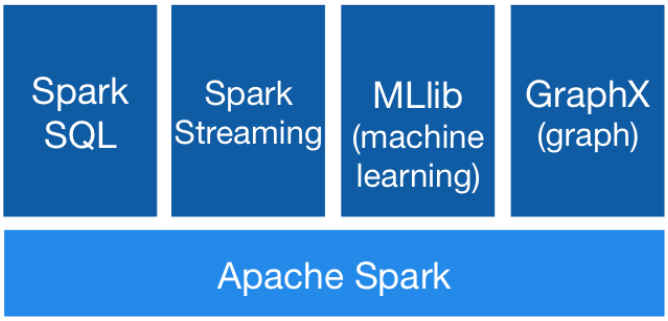


Figure 27. Apache Spark combining different analytic engines

For fault tolerance, Spark provides stateful exactly-one semantics out of the box. Spark Streaming recovers both lost work and operator state (e.g. sliding windows) without any extra coding required.

[78] http://idss.cs.put.poznan.pl/site/60.html
[79] http://www.cs.put.poznan.pl/ewgmcda/index.php/software
[80] http://www.decision-deck.org/project/
[81] https://spark.apache.org/
[82] http://spark.apache.org/mllib/

In PROTECTIVE Apache Spark, should the scale of data that we will process deem it needed, might be ultimately used to provide efficient means of events correlations.

Apache SAMOA[83] enables development of new ML algorithms without directly dealing with the complexity of underlying distributed stream processing engines (DSPE, such as Apache Storm, Apache Flink, and Apache Samza). Apache SAMOA users can develop distributed streaming ML algorithms once and execute them on multiple DSPEs. It bases on MOA[84] (Massive Online Analysis) – an open source framework for data stream mining. MOA includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, concept drift detection and recommender systems) as well as tools for evaluation. It is well-suited for developing new machine learning schemes.

Pandas[85] is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It aims to be the fundamental high-level building block for performing data analysis in real scenarios with Python. Pandas is not itself for machine learning but it rather is intended to help understand the processed data in the ML Python stack. According to the vendor, pandas is well suited for many different kinds of data: tabular data (from SQL databases), ordered and unordered time series data, arbitrary matrix data and also other observational or statistical data sets.

Scikit-learn[86] – open source and commercially usable (BSD license) toolkit for data mining and data analysis. Provides tools e.g. for classification, regression, clustering, dimensionality reduction, model selection, preprocessing. Using a dedicated module sklearn-pandas[87], it is feasible to integrate Scikit-learn with Pandas.

[83] https://samoa.incubator.apache.org/
[84] http://moa.cms.waikato.ac.nz/
[85] http://pandas.pydata.org/
[86] http://scikit-learn.org/stable/
[87] https://github.com/paulgb/sklearn-pandas

6 Implementation Framework Procedures and Tools

6.1 Introduction

It is a difficult task to define a proper framework for PROTECTIVE's distributed project development that spans across Europe. Our approach is to adapt agile development strategies which, with a few derivations from the original manifesto[88], are already acknowledged in other H2020 projects (e.g. the SymbIoTe H2020 project[89]). For the preparation of certain part of this section, we have partially referred to the symbIoTe Deliverable D5.1 "Implementation Framework"[90] as PSNC is also a member of the symbIoTe consortium and co-author of that deliverable, especially concerning its implementation framework part.

The document starts by derivations from their dictionary defining the various roles within the project, inspired by the Agile approach for project development (on the example of Scrum which has also been described). The considered roles are Product Owner, Scrum Master, Component Owners and Development Team. The deliverable then describes the system release iteration, which states the main implementation workflow from feature planning to system release. This workflow follows a cycle of four steps, briefly described below:

- **Feature Planning**, where new features and system validation tests are specified
- **Development**, where the collaborative implementation for the system takes place using the implantation framework procedures and tools defined in this deliverable, along with the continuous integration process
- **System Testing**, where the entire integrated system is tested
- **System Release**, following successful system testing, the system is then released and a new release cycle can commence

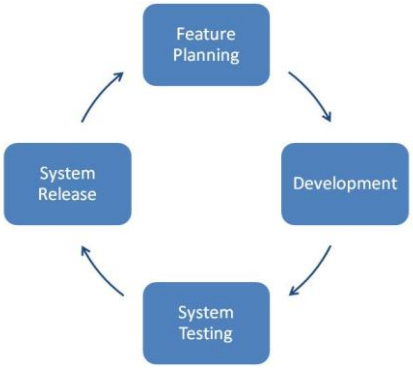


Figure 28. PROTECTIVE release cycle

Additionally, software security has been addressed in an emphasized way as PROTECTIVE tools will be sensitive – they are intended to increase security, not introduce security vulnerabilities themselves (as security solutions they will be, very probably, intentionally attacked).

Finally, the second part of this section considers the actual toolset selected in order to actually implement the assumed procedures.

[88] http://agilemanifesto.org/
[89] https://www.symbiote-h2020.eu/
[90] https://pl.scribd.com/document/323359670/D5-1-Implementation-Framework

6.2 Procedures

6.2.1 Agile and Scrum

6.2.1.1 Introduction

To effectively manage the development tasks, we propose to apply the **Scrum methodology**[91], effectively applied by consortium members in many R&D projects (e.g. PSNC applies it in the GN4-2 H2020 project). Scrum methodology is characterised by end-product's high quality level and client's satisfaction. The quality is assured by many factors such as: involving the client on every project development stage, regular meetings and a high team involvement. First we introduce general concepts, terms and roles and then we map them with the PROTECTIVE pipeline to the maximum possible extent as for the time of issuing this document. Scrum methodology is one of many *agile* software development methodologies used for highly complex and innovative projects. In Scrum the emphasis is laid mainly on the large number of interactions with a client and dividing the project into *sprints* – smaller, partially independent tasks. Figure 29 shows the Scrum approach at a glance.

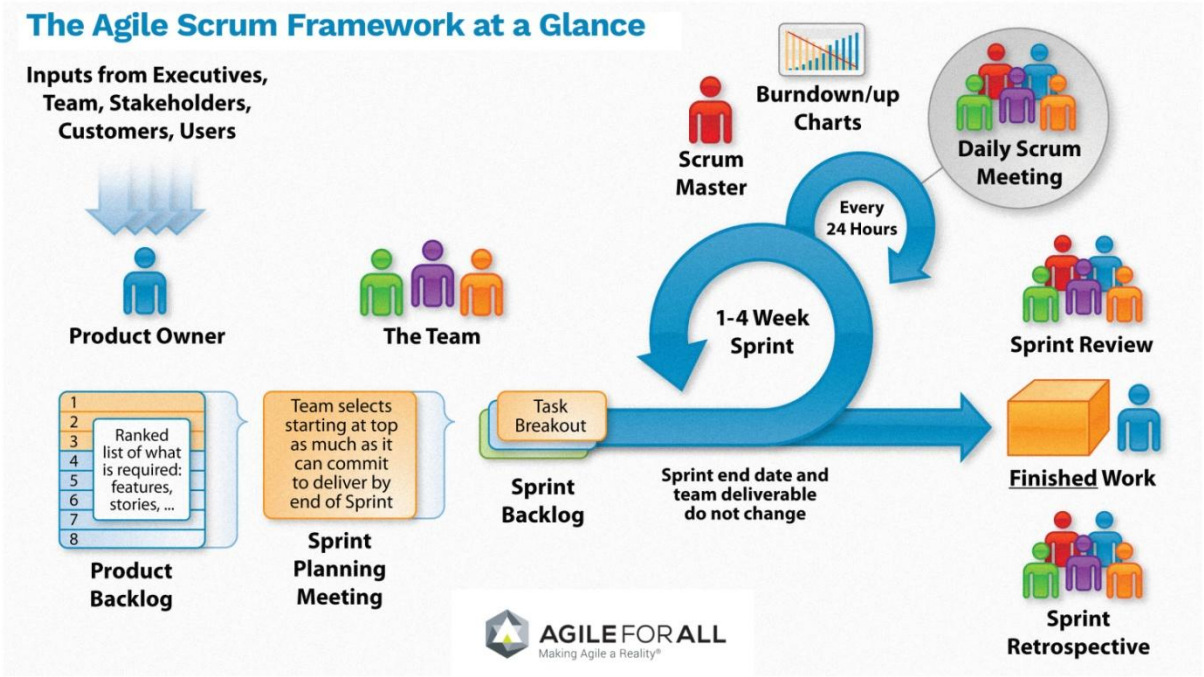


Figure 29. Scrum methodology[92]

The project team size may depend on the size of the module that is implemented. In PROTECTIVE, there are several modules expected to be provided (certain modules may be developed by single partners but other will involve multiple partners), but we expect it consist of 2-5 people, each of whom enriches the project with different competences, which in turn allows it to perform particular tasks. Experience in collaboration e.g. between NRENs in European GÉANT-related projects will facilitate self-organisation of tasks even between multiple partners. Therefore, the distribution of work among the members should be smooth and effective.

6.2.1.2 Roles

In the Scrum methodology, the person binding the whole team is a **Scrum Master** (SM). SM main task could be identified as removing any obstacles preventing the working team from carrying out the sprint, but not leading or coordinating the project himself/herself. Instead, SM should help the

[91] http://scrummethodology.com/
[92] http://agileforall.com/resources/introduction-to-agile/

others to communicate efficiently and provide them with technical support. According to the methodology guidelines, SM should be the person with the biggest experience in the particular field.

Scrum requires appointing a **Project Owner** (PO). It should be either a client's (in our case – e.g. a participating NREN or SME representative that deploys a particular pilot) representative familiar with the Scrum methodology. PO specifies project requirements (Product Backlog).

The **Development Team**, differently from other methodologies, is a team that not just receives its tasks from the project leader. It rather decides self-dependent, which requirements or User Stories it can accomplish in one sprint. It constructs the tasks and is responsible for the permutation of those – the team becomes a manager.

In order to keep the development process more consistent, we propose, where applicable (for more complex modules) to introduce the role of **Component Owner** (CO), with the purpose of facilitating feature planning and coordination of the development process. CO task will be to handle the development process within subsets of the development team responsible for individual components of the architecture.

6.2.1.3 Scrum Stages

The Scrum project is initially divided into the following steps:

- **Product Backlog** is a set of requirements set for the whole project. On its basis the tasks will be defined and assigned to particular sprints. Every task has to be assigned ID, subject, content, priority, sprint ID, and status. The team collaborating with the PO compose the so called User Stories, i.e. a set of scenarios describing the particular product's functionality
- **Sprint Planning** consists of planning the team availability and capability and assigning the tasks to a particular sprint and a so called Planning Poker – a discussion among the team members about the tasks' time consumption.
- **Sprint Backlog** contains functionalities acquired in User Stories for a particular sprint. All tasks assigned to a sprint should be put on the sticky notes and placed on a white-board divided into the following columns denoting status(To Do, In Progress, Done). Repositioning of the sticky notes is done by task performers and supervised by the PO.

Sprints should go on for periods between 1-4 weeks. We propose to use 2 weeks sprint which facilitates reporting the sprint results during biweekly Executive Board meetings. 2 weeks should also be enough time to keep the organizational sprint activities on relatively low time consumption rate. We propose to have sprint meetings every 2 weeks while the internal communication within the partner entity or between multiple partners will be held on demand as necessary. Each sprint will finish in tasks review. Should a task not get finished in time due to unexpected problems, it can be postponed to the next sprint but also the risk of not finishing the deliverable resulting from this tasks will be escalated and managed. A task is considered done only when the PO accepts it.

6.2.2 Feature Planning and Issue Tracking

The PO will define the set of component features to be implemented during the current iteration. On the lower level, the relevant Task Leaders will specify the list of features that will be reviewed (and, if necessary, modified) by the Project Coordinator – assuming cooperation with TLs. The final feature definition will generally reflect the established requirements, overall architecture design and detailed design performed within relevant task – and will be the basis for the Development Teams activities. The SM will support the feature specification process by providing input related to the current software architecture and its relation to the broader system architecture e.g., information concerning the inter-component communication interfaces.

This stage will also include a detailed specification of system tests, oriented towards validating proper functioning and interoperability of the components. In order to achieve that, the system test specification will include a set of execution scenarios, with precise descriptions of the expected outcome. The feature list as well as and system test specifications will be documented in the Release Specification report.

The SM will also provide priorities for the feature implementation tasks and assign the defined features as well as whole components to Development Teams who will become CO, responsible for the internal distribution of development tasks. They will be also monitoring the development process within and help the SM to plan further activities. The feature or component assignment will be done by the SM e.g. by issuing relevant tickets. SM will also assign the system tests to the relevant development or testing team.

6.2.3 Development

The development process will be composed of several stages, shown in Figure 30 and then described in more detail below.

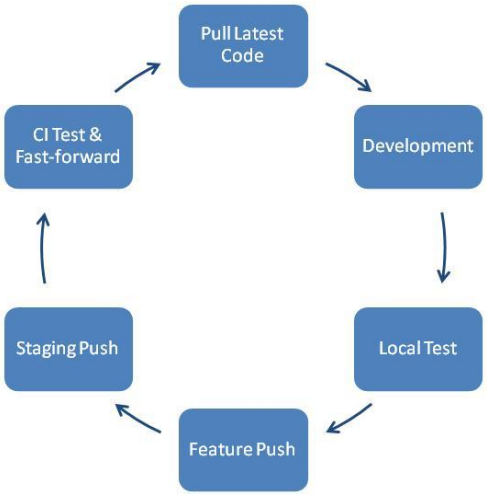


Figure 30. PROTECTIVE development cycle.

- **Pull Latest Code** – the Developer Teams, accordingly to the set of assigned tasks, pull the most recent stable code from the source code repository and the develop branch for implementing features but also e.g. code refactoring or fixing bugs.
- **Development** – the members of Development Teams will be working on their assigned tasks on a dedicated branch derived from the develop branch via a Version Control tool.
- **Local Test** – the Development Team members perform locally unit tests.
- **Feature Push** – the Development Team members periodically commit their changes locally and push them to the remote feature branch in order to maintain a backup of their work. This may also concern unfinished tasks.
- **Staging Push** – after completion of the code to the degree that it may be merged into the main develop branch, the Development Team members push the code to a temporal staging branch in the online repository.
- **CI Test & Fast Forward** – the continuous integration server hooks to changes on a staging branch and, if it detects such, triggers the build of the modified component in its own environment, and runs all test relevant to the component. If all tests are successful, the code is forwarded from the staging branch to the develop branch. Otherwise the bugs must be fixed before the next push.

6.2.4 System Testing

System testing stage is performed to thoroughly test the entire integrated system. The scope of the tests will be specified basing on the Release Specification document prepared in the Feature Planning stage. In order to appropriately handle the identified bugs, tickets will be issued for them.
This stage is coordinated by the SM. System testing is performed either by a relevant part of the Development Team or by a separate team (it will also be specified in the Release Specification document).

6.2.5 System Release

After the system tests stage is successfully finished, the release may be completed and the next iteration of the development lifecycle may be initiated.

6.2.6 Software Quality Assurance

In order to produce high quality software in PROTECTIVE a set of quality standards and procedures must be developed and followed. Besides keeping to the rules defined for the implementation process, the quality criteria principles have been defined that are described in this section.

- **Coding Style** – identifies good practices and provides guidelines that allow easier maintainability of the code, e.g. a new developer is able to easier understand the existing sources. Coding style guidelines will be proposed and defined by development teams of particular components. The used style guidelines must be consistent with the well-known conventions for the programming language used for implementation (the most probable languages are Java and Python).
- **Unit testing** – allows to detect bugs in recently modified areas of the code or these at an early stage. It should be aimed that automated tests are performed by the use of unit testing facilities of the used Continuous Integration software. The Development Teams must configure and use relevant functionalities in order to conduct the unit tests. At least the functionalities considered as critical must be covered.
- **Integration testing** – allows to evaluate whether multiple developed modules interact correctly in real scenarios (individual components are combined and tested as a group). Particular components must pass all the integration tests defined. Similarly to unit testing, optimally integration tests should be run automatically by continuous integration server for all components that have their dependencies defined and updated.
- **System testing** – assures verifying a complete and integrated system and evaluate its compliance with specified requirements. System test suites must be defined and monitored.
- **Performance testing** – besides functional testing focused on filling security requirements, other types of tests are present, for instance performance testing which is especially meaningful for PROTECTIVE tools that will process large sets of information. Thus PROTECTIVE components must have defined performance requirements.
- **Security testing** – although security may be considered as a part of quality assurance, due to the criticality of the issue for PROTECTIVE toolset, security testing has been distinguished (and more thoroughly described in the section 6.2.7). PROTECTIVE tools must not cause security vulnerabilities themselves. Relevant security tests must be defined and executed for particular components as well as the final PROTECTIVE environment.
- **Documentation** – must be provided by PROTECTIVE Development Teams for all software modules being released. Documentation will be stored in Basecamp.

6.2.7 Software Security

It has been shown that the cost of addressing a software bug detected at an early stage of development lifecycle is significantly lower[93]. Therefore, PROTECTIVE consortium will have security in mind while developing bespoke software and will embed elements of Secure Development Life Cycle (SDLC) into the development process. These elements have been described below.

6.2.7.1 Secure Coding Guidelines

In order to ensure that proper security measures are used for the developed software, one of the existing standards may be used. We propose to apply OWASP ASVS (Application Security Verification Standard[94]) for that purpose with respect to the developed Web based applications. The current version is 3.0.1.

OWASP ASVS recognized the following security verification levels (see Figure 31):

- **ASVS Level 1** (Opportunistic) – for all software
- **ASVS Level 2** (Standard) – for applications that contain sensitive data requiring protection
- **ASVS Level 3** (Advanced) – for the most critical applications (e.g. performing high value transactions or containing medical data).

Figure 31. OWASP ASVS levels[95]

For the PROTECTIVE software expected to deliver within the project, we propose to apply ASVS Level 2 as particular modules will contain data that may be interesting for the attackers (e.g. data about infrastructure or detected security events) however it they will not perform any money flows or contain critical data. As OWASP ASVS document states "Level 2 ensures that security controls are in place, effective, and used within the application. Level 2 is typically appropriate for applications that handle significant business-to-business transactions, including those that process healthcare information, implement business critical or sensitive functions, or process other sensitive assets". Apparently, in the case of further development of PROTECTIVE in more sensitive environments (like critical infrastructure, LEA or military deployments) the requirements should be upgraded to ASVS Level 3 (Advanced).

[93] https://www.blackhat.com/presentations/bh-usa-06/bh-us-06-Morana-R3.0.pdf
[94] https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project
[95] Source: OWASP Application Security Verification Standard v. 3.0.1

Particular verification requirements are grouped in 19 lists (V1-V19), assuring an increasing level of security as the ASVS Levels rise (see Figure 32). Lists are finalized with references sections, containing links to instructions, cheat sheets, and descriptions of mentioned technologies.

#	Description	1	2	3	Since
1.1	Verify that all application components are identified and are known to be needed.	✓	✓	✓	1.0
1.2	Verify that all components, such as libraries, modules, and external systems, that are not part of the application but that the application relies on to operate are identified.		✓	✓	1.0
1.3	Verify that a high-level architecture for the application has been defined.		✓	✓	1.0
1.4	Verify that all application components are defined in terms of the business functions and/or security functions they provide.			✓	1.0

Figure 32. Exemplary OWASP ASVS 3.0 requirements for different levels[96]

For additional coding guidelines, we will refer to the recognized advisories and best practices, for instance in the case of Java programming language – SEI CERT Oracle Coding Standard for Java from Software Engineering Institute at Carnegie Mellon University[97] will be used. It defines several dozens of rules and recommendations that are assigned priorities. A priority is the result of multiplying three factors (each valued 1, 2 or 3):

- **Severity** (1 – low, 2 – medium, 3 – high),
- **Likelihood** (1 – unlikely, 2 – probable, 3 – likely),
- **Remediation cost** (1 – high, 2 – medium, 3 – low).

For the existing software, according to the "Rec.: Priority and Levels" page[98], rules with priorities P12-P27 must, with P6-P9 should, and with P1-P4 may be enforced. For the new implemented software, it is recommended to keep to all rules.

All resources (including the source code) in the project must be encoded to UTF-8 in order to prevent bugs and problems for the international development teams.

6.2.7.2 Security Code Reviews

The developed source code will be periodically reviewed (at least before each release). As the source code review is highly resource-consuming, it is expected to use static analysis tools (code scanners) for that purpose. It will be analyzed whether it is possible to incorporate specific scans into consecutive software builds.

Exemplary tools that may be used for the most probable programming languages:

- For Java:
  - FindBugs[99] – easily integrated with the most recognized IDEs (e.g. Eclipse and IntelliJ) as plugins
  - Find Security Bugs[100] – actually a plugin for FindBugs, facilitating integration e.g. with Jenkins as well as directly with IDEs, including Eclipse and IntelliJ

[96] Item V1: Architecture, design and thread modeling – 1.1-1.4 items shown out of 1.1-1.6 (source: OWASP Application Security Verification Standard v. 3.0.1)
[97] https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java
[98] https://www.securecoding.cert.org/confluence/display/java/Rec.%3A+Priority+and+Levels
[99] http://findbugs.sourceforge.net/
[100] http://find-sec-bugs.github.io/

  - Visual Code Greppe[101] – a standalone, multilingual tool
- For Python:
  - Flake8[102]
  - Bandit[103]
  - Pylint[104]

Majority of the aforementioned tools, plus many others, may be executed online on SWAMP – Software Assurance Marketplace[105]. However, it is planned to keep the testing process solely within the PROTECTIVE consortium infrastructure (that is, installing standalone tools on developers workstation or installing a SWAMP instance locally by deploying "SWAMP-in-a-box" feature[106]. SWAMP supports continuous integration with git and Jenkins as well as provides Eclipse plugin which allows to embed scans into consecutive builds. On the other hand, SWAMP not always provides the most recent versions of the used tools, which would require custom updates to the SWAMP-in-a-box deployment or just using the most recent tools standalone. During the deployment it will be analyzed which is the optimal solution at the time.

As the tools tend to detect both false positives and false negatives, we plan to additionally perform the following activities in order to mitigate this:

- Manually verify the results returned by the automated tools
- If we identify any particular security problem working with the developed solutions (either during testing or standard use), it will be traced in the source code in order to reach its origin

Formally, according to the "The Art of Software Security Assessment – Identifying and Preventing Software Vulnerabilities" book[107], it will be a Candidate Point (CP) strategy of code review – either Automated Source Analysis Tool CP, or Black Box Generated CP. We do not plan full code security review (formally: Code Comprehension strategy) as it would be extremely time consuming.

Additionally, we plan to use OWASP Dependency Check (see the section 6.3.5.2) in order to assure no dependencies with known vulnerabilities are used.

6.2.7.3 Security Evaluation

As a part of the pilot evaluation, the consortium will provide a security assessment of the pilot installations (within the confines of T6.5), with special emphasis put on elements that have been developed within PROTECTIVE. Common security tools will be used like e.g.:

- **General purpose security tools** such as nmap, Nessus, nExpose, OpenVAS etc.
- **Tools for assessment of particular protocols or services** such as sslscan for testing SSL/TLS configuration, etc.
- **Auxiliary tools** such as local http proxy, e.g. burp suite etc.

The tests will be complemented with manual tests performed by experienced penetration testers from partner entities, including persons not involved in the design and development of PROTECTIVE tools. The performed scenarios will include attempts of bypassing the proposed security solutions.

[101] https://sourceforge.net/projects/visualcodegrepp/
[102] https://gitlab.com/pycqa/flake8/
[103] https://wiki.openstack.org/wiki/Security/Projects/Bandit/
[104] http://www.pylint.org/
[105] https://continuousassurance.org/
[106] https://continuousassurance.org/swamp-in-a-box/
[107] https://www.amazon.com/Art-Software-Security-Assessment-Vulnerabilities/dp/0321444426

Security tests will be conducted according to the relevant standards, e.g. for Web components – the most recent OWASP Testing Guide at the time (currently in version 4[108]).

6.2.8 Organization

In order to support the smoothness of the development process, relevant organizational framework must be in place.

PROTECTIVE Executive Board (EB) meetings are biweekly therefore we suggest to maintain 2-weeks sprints in order to be able to report sprints via WP leaders during EB meetings. WP leaders should be responsible for collecting reports from Development Teams at least on biweekly basis, before EB meetings (which are scheduled every second Tuesday) – e.g. by the end of the previous week or on the day before EB. Regular development team calls should also be organized in to summarize the previous sprint and to plan the next one. Emergency calls in the case of need should be also organized for the development teams affected by a problem with timely implementing planned features with the Task or Work Package leader or, if the problem needs to be escalated – with the Project Coordinator. The PROTECTIVE Basecamp[109] will be used to maintain the meetings schedule. The pure development tasks will be, however, handled in the JIRA.

6.3 Implementation Framework Tools

This section enumerates (see Table 8) the tools that will be used to support the implementation framework described later in the section. The toolset has been selected basing on the experience of the Development Teams (especially PSNC who have contributed to this part of the report). A similar set of tools is used in the symbIoTe H2020 project(symbIoTe, 2016) where the PSNC Development Teams are also involved. Whenever necessary, relevant guidelines will be provided for other PROTECTIVE development teams on how to effectively use the mentioned tools.

Purpose	Tool
Version control	Git
Code repository	Gitlab
Feature planning and issue tracking	Gitlab
Continuous Integration	Travis
Artifact repository	Docker Registry and other
Building tool	Gradle

Table 8. PROTECTIVE implementation basic toolset

Tools intended to be used for code security assessment have been briefly described in the 6.2.7.2 section. Additionally, a short notice about assuring security of the implementation framework itself has been provided.

6.3.1 Version Control

6.3.1.1 Git

We have proposed Git for version control as it better supports development efforts that are distributed among different Development Teams (which will be the case for PROTECTIVE). Git was designed from the ground up as a distributed version control system. Being a distributed version

[108] https://www.owasp.org/images/1/19/OTGv4.pdf
[109] https://3.basecamp.com/3521314/projects/1400766

control system means that multiple redundant repositories and branching are first class concepts of the tool (GitSvnComparison, 2017*)*.

The most relevant features are as follows:

- Git allows developers to have a full local copy of the entire repository, including history, which facilitates easy and prompt access to history.
- Git allows full functionality when disconnected from the network – via the local cloning feature.
- Possibility to easily maintain a complete backup of the repository.
- Support for branching, including support for multiple local branches.
- Git is faster compared to SVN and creates significantly smaller repositories
- Git facilitates proper maintaining of line ending when Development Teams work on heterogeneous client operating system.

Subversion also has several advantages over Git but we have not found them as critical (e.g. user interfaces maturity, better access control due to a single repository, better handling of binary files, opportunity to check out only a part of the repository or shorter and more predictable version numbers).

In PROTECTIVE we will create a number of repositories for different developed modules, owned by relevant Development Teams. Within all repositories the branching model described below is suggested to be used.

6.3.1.2 Branching Model

We propose to use a Git branching model introduced in (GitBranching, 2010). We plan to create separate repositories for different PROTECTIVE modules. The suggested model uses a number of branches in order to facilitate concurrent development of features, bug fixes as well as release preparations. We will use two types of main branches and four types of supporting branches. The main branches will last for the project lifetime while the supporting branches are temporary (and will be eventually removed). The Table 9 summarizes the properties of particular branches:

Name	Type	Description
Master	Main	The branch where the source code of HEAD always reflects a production-ready state.
Develop	Main	The branch where the source code of HEAD always reflects a state with the latest delivered development changes for the forthcoming release. Automatic *nightly builds* are based on this branch.
Feature	Supporting	Created from the develop branch in order to implement a new feature. Supports parallel development of multiple features of the component by different teams. If the feature is successfully implemented, it will be merged with the develop branch, otherwise – discarded.
Release	Supporting	Created from the develop branch in order to ease preparing a new product release after all assumed features have been merged. In this branch only minor bug fixes plus limited code tuning (commenting etc.) should be acceptable. Eventually this branch will be merged to the master branch, producing a new release. That approach facilitates simultaneous adding new features for the next release, while preparing the current release. The release branch is also merged with the develop branch to include the most recent changes.
Hotfix	Supporting	Created from the master branch in the case of appearing a critical bug to be promptly fixed. Immediately after the bug is fixed, the hotfix branch is merged with the master branch and a new minor release is issued. Additionally, the hotfix branch must be

Name	Type	Description
		merged with the develop branch in order to assure that the hotfix will be included in future releases as well. But the bug fixing itself does not impact the daily work on the develop branch.
Staging	Supporting	Periodically the develop branch will be frozen (changes to it will be blocked) Upon being done with their tasks developers will be required to push their changes from the feature branch into the staging branch. The continuous integration server will detect modifications of the staging branch and build the environment for the relevant component and run all associated tests. If all tests pass, the CI server forwards them from the staging branch to the the develop branch. If some tests fail, the developer will be notified about the code failure, the push will be rejected and the bug will have to be fixed before merging with the develop branch.

Table 9. Proposed branching model

The branching model (except the staging branch, which is an extension) is shown in the Figure 33.

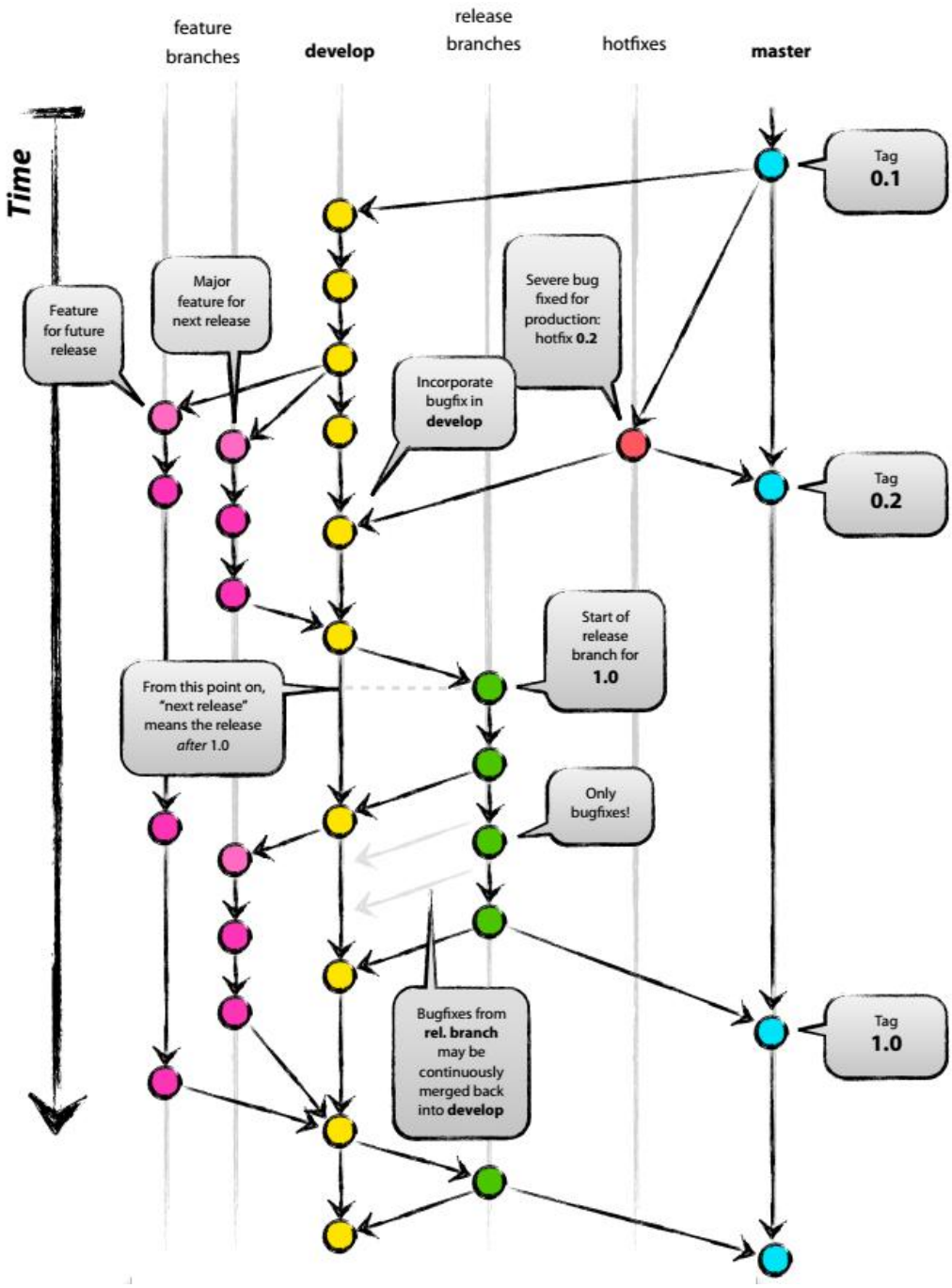


Figure 33. Git branching model (author: Vincent Dressien[110])

6.3.1.3 Semantic Versioning

In order to further facilitate maintaining distributed code that has a lot of dependencies, we additionally propose to use the semantic versioning[111]. Semantic versioning distinguishes at least three numbers: major version number, minor version number and a patch version number (e.g. version 5.0.2). The aforementioned numbers should be modified in the following situations:

- MAJOR version – after changes that cause the current API to be incompatible
- MINOR version – after extending functionality in a backwards-compatible manner
- PATCH version – after implementing backwards-compatible bug fixes.

[110] http://nvie.com/posts/a-successful-git-branching-model/
[111] http://semver.org/

Additionally, extra labels for pre-release and build metadata are applicable as extensions.

Consistent applying semantic versioning can spare a lot of effort when managing dependencies throughout multiple source code components.

6.3.2 Code Repository, Feature Planning and Issue Tracking: Gitlab

We have selected Gitlab for building and maintaining the source code repository. The main repository will be located in the Gitlab cloud and the backup one on Gitlab CE in the RoEduNet instance.

The main features of Gitlab that are significant for development of PROTECTIVE tools are as follows:

- Possibility of organizing a repository into a set of private, internal or public projects
- Managing access permissions with 5 different roles
- Creating websites for projects, groups and user accounts
- Keeping the documentation of the project using built-in Wiki
- Importing existing projects from a wide set of repositories
- Ability to set read/write permissions to particular branches as well as locking any file or directory when its merging is not feasible
- Ticketing facilities for the development teams where individuals may be assigned single issues (as tickets) – from fixing a simple bug to implement a defined feature; the tickets may be easily reviewed either by assigner or by assignee.

We have assessed Gitlab as having the most advantages over other comparable tools; for instance, in comparison to Github.com it has got free support for private projects, built-in facilities for integration with Continuous Integration (CI), exporting project capabilities, more flexible access permissions or indicating “work-in-progress” code facilities.

Recently Gitlab introduced paid subscriptions[112], however it should not impact the development projects of a PROTECTIVE level. The main difference for our purposes will be limiting the monthly number of private CI minutes on the shared runners to 2000 (which is equal to running a 10-minutes pipeline 5 times per each working day in the month). Gitlab is also going to start actually enforcing code size limit of 10 GB repository storage.

In an extreme case we will consider either establishing a local runner by one of the partners or purchasing a relevant license.

6.3.3 Continuous Integration

Currently, the most popular Continuous Integration tools are Travis CI and Jenkins. However we have cosen Gitlab CI and Gitlab Runner as they are integrated with Gitlab selected as the source code repository. Travis better integrates with Github. Additionally in the case of private project it is more convenient to use local GitLab instance to better assure the privacy of the source code.

Additionally, other supporting software may be used if a particular Development Team wish (due to the popularity of Jenkins it has been described as an example).

6.3.3.1 Gitlab CI & Gitlab Runner

Gitlab CI and Gitlab Runner[113] are both built-in parts of GitLab and may be used for free as its parts.

[112] https://about.gitlab.com/2017/04/11/introducing-subscriptions-on-gitlab-dot-com
[113] https://about.gitlab.com/gitlab-ci/

GitLab CI is a Web application with an API that stores its state in a database. It manages projects/builds and provides an easy to use GUI. The most significant features of Gitlab CI are described below:

- Is multi-platform (including Windows and Unix/Linux)
- Is multi-language (including Java and Python)
- Opportunity to define complex pipelines (multiple jobs per stage, triggering other builds)
- Scalability and parallel builds in order to speed up the process when necessary
- Tests versioning
- Possibility of reproducing tests locally
- Support for Docker – facilitates using custom Docker images, building new images, running on Kubernetes

Thanks to the tight integration with GitLab, it is extremely fast to establish a CI project.

GitLab Runner is an application that processes builds. It can be deployed separately and works with GitLab CI through an API. In order to run tests, at least one GitLab instance and one GitLab Runner are necessary. GitLab Runner may be run on every platform that supports Go programming language, including Windows and Linux (and Docker). It can also test actually every programming language, including probably the mostly used Java and Python. Runner has got a set of interesting features, e.g. autoscaling, wide Docker support as well as ability to run multiple jobs simultaneously.

6.3.3.2 Jenkins

Development teams may use their favourite tools as supplementary services. For instance, Jenkins[114] is a self-contained, open source automation server which can be used to automate all sorts of tasks such as building, testing, and deploying software. Jenkins can be installed through native system packages, Docker, or run standalone by any machine with the Java Runtime Environment installed.

Jenkins is equipped with a set of plugins to integrate with GitLab using hooks and API, e.g.:

- https://docs.gitlab.com/ee/integration/jenkins.html
- https://medium.com/@teeks99/continuous-integration-with-jenkins-and-gitlab-fa770c62e88a#.6tvy0lqno
- https://github.com/jenkinsci/gitlab-plugin
- https://wiki.jenkins-ci.org/display/JENKINS/Gitlab+Hook+Plugin

6.3.4 Artifacts Repository: Docker Registry and Artifactory by Jfrog

A repository serving build artifacts is an integral part of a CI environment – especially in a PROTECTIVE-like modular system where producing at least a few dependencies on each other modules (Java jars) is expected. While GitLab provides capabilities for handling binary artifacts, we plan to use private Artifactory repository for jars and Docker Registry running on premises for in house images keeping.. The Java jars are the result of each successful build and are published to the repository so that there is always a built module available to serve as a dependency for other modules using maven/ivy dependency model which Gradle (see below) ingests without problems. Artifactory contains a plugin for Gradle allowing to publish artifacts to it.

6.3.5 Building Tools

6.3.5.1 Gradle

An automated build tool is required to facilitate the implementation process and assure the desired quality of the delivered products. Additionally, automated build allows to easier handle certain

[114] https://jenkins.io/

development concerns, like dependency handling as well as enhances the performance of the build process.

For Java programming language, which will be used at least for modules developed in PSNC, the most recognized tools for automated build are Ant, Maven and Gradle. However, as Gradle is actually built on the top of both Ant and Maven, it will be the only tool described there.

Gradle was released in 2012. It is actually built on Ant and Maven. The previous tools used XML for writing build scripts, while Gradle introduced a Groovy-based Domain Specific Language (DSL) for that purpose. Thanks to DSL, creating and maintaining build scripts is simple, and the scripts themselves are noticeably shorter and easier to understand. Gradle is supported by a highly active community and offers excellent documentation. Build speed is comparable to Maven. As Gradle is younger, the set of integration facilities and plugins is smaller than for Ant or Maven, but still wide and well matching other parts of the selected implementation framework. In the case of need, Gradle features a simple and well-documented process for creating plugins.

Gradle furthermore features a simple and well-documented process for creating custom plugins which might be needed for our process.

6.3.5.2 OWASP Dependency Check

We will use OWASP Dependency Check[115] in order to assure that no used dependencies contain publicly disclosed security vulnerabilities. For the most selected programming languages, OWASP DC currently fully supports Java and contains experimental support for Python. The tool check automatically updates itself using the NVD Data Feeds hosted by NIST. OWASP Dependency Check may be integrated via plugins with Maven, Gradle, Jenkins, Ant and others.

6.3.6 Runtime: OpenStack & Docker

In order to maximize the project's reach and possibly to simplify the deployment and maintenance efforts as well as to minimize the (both internal and external) contributors' entry difficulty threshold we will deliver the PROTECTIVE suite utilizing two tools – Docker and Openstack.

6.3.6.1 Docker – Optimizing Virtual Resources

Before we describe them in depth, a short introduction to Docker's basics is required. Therefore, taken from Docker Docs[116]:

A brief explanation of containers

An **image** is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and configuration files.

A **container** is a runtime instance of an image – what the image becomes in memory when actually executed. It runs completely isolated from the host environment by default, only accessing host files and ports if configured to do so.

Containers run apps natively on the host machine's kernel. They have better performance characteristics than virtual machines that only get virtual access to host resources through a hypervisor. Containers can get native access, each one running in a discrete process, taking no more memory than any other executable.

[115] https://www.owasp.org/index.php/OWASP_Dependency_Check
[116] https://docs.docker.com/get-started/

Containers vs. virtual machines

Consider this diagram comparing virtual machines to containers:

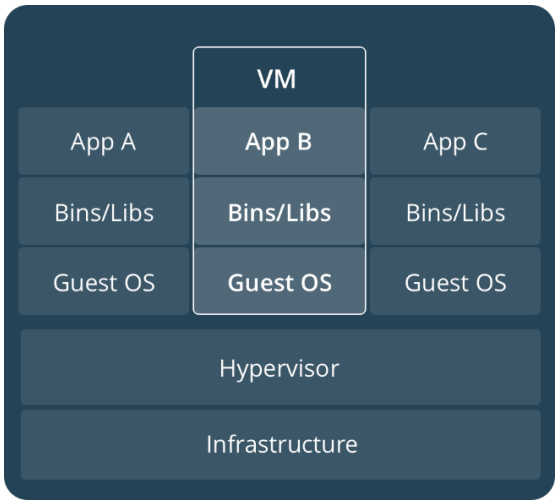


Figure 34. Virtual Machine diagram[117]

Virtual machines run guest operating systems – note the OS layer in each box. This is resource intensive, and the resulting disk image and application state is an entanglement of OS settings, system-installed dependencies, OS security patches, and other easy-to-lose, hard-to-replicate ephemera.

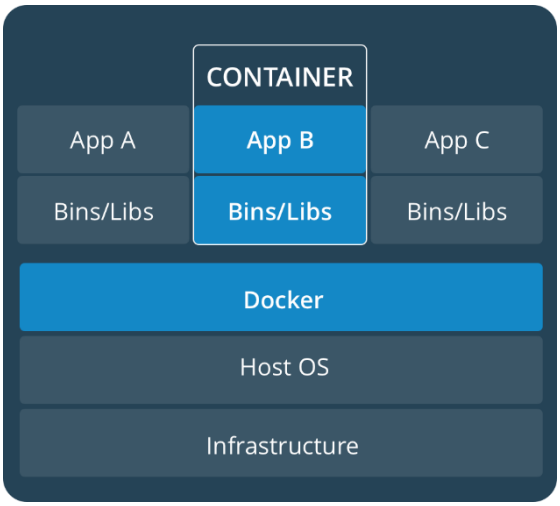


Figure 35. Container diagram[117]

Containers can share a single kernel, and the only information that needs to be in a container image is the executable and its package dependencies, which never need to be installed on the host system. These processes run like native processes, and may be managed individually by running commands like *docker ps* – just like running *ps* on Linux to see active processes. Finally, because they contain all their dependencies, there is no configuration entanglement; a containerized app "runs anywhere."

We are able to state that it will allow us to clearly define the to-be-developed microservices domains bounds and provide them as coherent Docker images, consisting of all the services (project results),

[117] https://docs.docker.com/get-started/#containers-vs-virtual-machines

their dependencies (e.g. Linux, JRE, databases) packed into minimal read-only layers, provisioned with all the necessary to run configuration data externalized to a separate layer and running as lightweight container (Virtual Machines) thus maximizing the utilization of provided resources (CPU, RAM, drive space). The Figure 36 shows an example of how PROTECTIVE results container would be built.

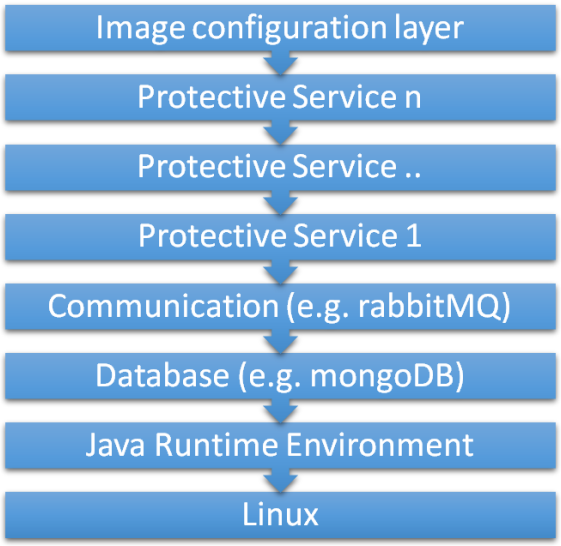


Figure 36. Example docker container with PROTECTIVE tools

It is also worth noting that, thanks to the design of Docker system, layers up to Communication can be reused and at the runtime host across all the PROTECTIVE containers will consist of components allowing to maximize the resources dedicated to a particular PROTECTIVE deployment.

Moreover, the dependency layers are already prepared and maintained by their developers in the Docker Hub which is available publicly. This availability and clear separation between components in the runtime will facilitate efficient maintaining the whole runtime environment by robustly acquiring necessary updates that introduce identified bug's fixes and new features required by the developers.

Docker provides us with another advantage. Thanks to the Docker Registry project, all the PROTECTIVE services layers will be kept private in the consortium internal images registry and exposed only to involved NRENs and SMEs clients.

The Docker Registry hosted in the consortium infrastructure will be automatically fed with updated layers as the final step of the developed continuous integration pipeline.

Last but not least, the dockerized delivery combined with the microservices architecture of PROTECTIVE tools, besides allowing for a centralized and simplified configuration of particular deployments, has a major impact on the project scalability. By extending the basic containers deployment and delegating their lifetime to orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos we will have measures to automatically load balance the containers, if-needed spool up additional ones (e.g. when a large flow of event for correlation occurs) or when we will simply want to have the final ranking a prioritization layer services to be running as separate containers per each operation in order to separate the influence of one's actions on another's and provide best quality of service.

6.3.6.2 Openstack – Optimizing Hardware Resources

The previous section described our approach to efficient handling of virtual resource (CPUs, RAM, storage space) by the delivered PROTECTIVE services. We would like to however further maximize

the efficiency of the available hardware resources. As the NRENs typically work with their datacenters, we will use the Openstack project in the development, integration and production environments for NREN deployments.

As such, PSNC will provide a large partition of resources (CPUs, RAM, storage & routable ipv4 subnet) for the project development and integration environments in their own data center located in Poznań (Poland). This partition will allow the developers and project integrations to easily create new virtual machines contained within the Warden and Mentat enabled secured networks and work on data flowing in from integrated probes to those tools. This partition will work on PSNC's experimental Openstack deployment in which we will be able to stress test our tools scalability and demand new features from Openstack administrators at PSNC.

For the Pilot+ phases of the project and production release (including the project results maintenance period) of the software PSNC will offer a stable Openstack environment and recommend each NREN to have its own local deployment.

We are deliberately not describing Openstack in detail as we intend to use it as a service from the datacenters associated with each NREN. We only find worth noting that during PROTECTIVE suite development we will collaborate with the administrators of such services in our datacenters evaluating possible integration of Docker orchestration engines with Openstack. One of such technologies that we find promising and willing to evaluate is *Magnum*[118]*:*

Magnum is an OpenStack API service developed by the OpenStack Containers Team making container orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources in OpenStack. Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either virtual machines or bare metal in a cluster configuration.

6.3.7 Implementation Environment Security

The consortium will assure that the software components used to build the common implementation framework will be secured against known cyber-attacks, thus preventing from e.g. stealing the source code before it is released, maliciously modifying the source code (e.g. in order to embed a backdoor) or attacking developers' accounts.

Relevant procedures will be employed, especially with regard to:

- Maintaining the list of used component versions and monitoring whether
  - There are any notifications about disclosed security vulnerabilities in the component
  - Vendor issued a new version of the component, especially described as a security patch (however in certain cases no explicit notifications on addressing security bugs are issued by the vendor)
- Periodic security scan of the publicly exposed interfaces with general purpose tools like nmap, nExpose, OpenVAS etc.

Monitoring the security level of particular components is a key issue for the development environment itself. There are cases that critical security vulnerabilities are identified in the proposed components which have to be exposed at least for the PROTECTIVE consortium partners. An example might be the case of disclosing a critical security vulnerability in Jenkins on May 1st, 2017, which, when exploited, allows an unauthenticated attacker to execute code on the vulnerable server[119].

[118] https://wiki.openstack.org/wiki/Magnum
[119] https://blogs.securiteam.com/index.php/archives/3171

7 References

CESNET. (2017). *IDEA*. Retrieved 4 25, 2017, from https://idea.cesnet.cz/en/index

ENISA. (2014, Jan). *Actionable Information for Security Incidence Response.* Retrieved Nov 2016, from https://www.enisa.europa.eu/news/enisa-news/new-guide-by-enisa-actionable-information-for-security-incident-response

ENISA. (2013). *Detect, SHARE, Protect Solutions for Improving Threat Data Exchange among CERTs.*

ENISA. (2014). *ENISA report: Standards and tools for exchange and processing of actionable information.*

FIRST. (1995 - 2017). *Global Initiatives - Cybersecurity structured information*. Retrieved 04 25, 2017, from https://www.first.org/

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.

Habib, S. M., Volk, F., Hauke, S., & Mühlhäuser, M. (2015). Computational trust methods for security quantification in the cloud ecosystem. In R. Ko, & K.-K. R. Choo (Eds.), *The Cloud Security Ecosystem -- Technical, Legal, Business and Management Issues* (pp. 463-493). Elsevier/Syngress.

IntelMQ. (2017). *IntelMQ*. Retrieved 04 25, 2017, from https://github.com/certtools/intelmq

Kijewski, P. P. (2012). *Proactive Detection and Automated Exchange of Network Security.* CERT.PL.

Mentat. (2017). *Mentat*. Retrieved 04 25, 2017, from https://mentat.cesnet.cz/en/index

MISP. (2013). *MISP*. Retrieved 04 25, 2017

OASIS-CTI-TC. (2017). *STIX Project*. Retrieved 04 25, 2017, from https://oasis-open.github.io/cti-documentation/stix/about.html

OASIS-CTI-TC. (2017). *TAXII*. Retrieved 04 25, 2017, from https://taxiiproject.github.io/

ONI. (2015). *ONI*. Retrieved 04 25, 2017, from https://github.com/Open-Network-Insight/

Spot, A. (2017). *Apache Spot*. Retrieved 04 25, 2017, from https://github.com/apache/incubator-spot

symbIoTe. (2016). Retrieved from D5.1 - Implementation Framework: https://pl.scribd.com/document/323359670/D5-1-Implementation-Framework

TheHive. (2016). *TheHive*. Retrieved 04 25, 2017, from https://thehive-project.org/

Warden. (2014). *WARDEN*. Retrieved 04 25, 2017, from https://warden.cesnet.cz/en/index

Willis, B. (2012). *Sharing Cyber-Threat Information: An Outcomes-based Approach.* Intel Corporation.

8 Annexes

8.1 Annex A: STIX 2.0 Versus 1.x

This annex provides a quick overview of the differences between STIX 1.x/CybOX 2.x and STIX 2.0[120].

One Standard

First, it has been decided to merge the two specifications into one. Cyber Observable eXpression (CybOX™) objects are now called STIX Cyber Observables.

JSON vs. XML

STIX 2.0 requires implementations to support JSON serialization[121], while STIX 1.x was defined using XML. Though both XML and JSON have benefits, it has been determined that JSON was more lightweight, and sufficient to express the semantics of cyber TI information. It is simpler to use and increasingly preferred by developers.

STIX Domain Objects

All objects in STIX 2.0 are at the top-level[122], rather than being embedded in other objects. These objects are called STIX Domain Objects (SDO). Some object properties use a reference to another object's id directly, but most relationships are expressed using the top-level Relationship object. The generic TTP and Exploit Target types from STIX 1.x have been split into separate top-level objects (Attack Pattern, Malware, Tool and Vulnerability) with specific purposes in STIX 2.0.

Relationships as Top-Level Objects

STIX 2.0 introduces a top-level Relationship object[123], which links two other top-level objects via a named relationship type. STIX 2.0 content can be thought of as a connected graph, where nodes are SDOs and edges are Relationship Objects. The STIX 2.0 specification suggests different named relationships, but content producers are able to define their own. In STIX 1.x relationships were "embedded" in other objects. The types of relationships supported was restricted by the STIX 1.x specification. Because STIX 1.x relationships themselves were not top-level objects, you could not express a relationship between two objects without changing one of them. It is often desirable for others to assert a relationship. Using this new Relationship object, others, besides the original content creator, can add to the shared knowledge in an independent way.

Streamlined Model

Experience with STIX 1.x showed that a common set of features were widely used and well understood while many other features lacked shared understanding and had only limited, if any use at all. In addition, almost all properties of objects were optional. Overall, the breadth of STIX 1.x was an impediment to sharing intelligence, and necessitated a formal agreement among threat groups on what should be shared.

[120] https://oasis-open.github.io/cti-documentation/stix/compare.html
[121] http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html#_Toc482357256
[122] http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html#_Toc482357251
[123] http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html#_Toc482357252

STIX 2.0 takes a different approach – many properties are required, and the number of objects and properties have been reduced to a core set of well understood features.

Data Markings

Data Markings[124] no longer use a serialization specific language, there are two types of data markings: object marking – applicable to a whole object, and granular markings – applicable to a property or properties of an object. Data markings scope is only within the object where they are defined.

Indicator Pattern Language

Indicator patterns in STIX 1.x were expressed using XML syntax. This made all but the simplest patterns difficult to create and to understand. STIX 2.0 takes a different approach, specifying which is independent of the serialization language. Patterns written in the STIX patterning language are more compact and easier to read. Additionally, there is no confusion between patterns and observations, because a pattern is not a top-level object, but a property of an indicator object.

[124] http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html#_Toc482357288

8.2 Annex B: IDEA Examples

This annex provides several examples of different types of security events, described in the IDEA format.

Scanning

```
{
   "Format": "IDEA0",
   "ID": "3ad275e3-559a-45c0-8299-6807148ce157",
   "DetectTime": "2014-03-22T10:12:56Z",
   "Category": ["Recon.Scanning"],
   "ConnCount": 633,
   "Description": "Ping scan",
   "Source": [
      {
         "IP4": ["93.184.216.119"],
         "Proto": ["icmp"]
      }
   ],
   "Target": [
      {
         "Proto": ["icmp"],
         "IP4": ["93.184.216.0/24"],
         "Anonymised": true
      }
   ]
}
```

Listing 3. IDEA scanning example

Honeypot

```
{
   "Format": "IDEA0",
   "ID": "2E4A3926-B1B9-41E3-89AE-B6B474EB0A54",
   "DetectTime": "2014-03-22T10:12:31Z",
   "Category": ["Recon.Scanning"],
   "ConnCount": 633,
   "Description": "EPMAPPER exploitation attempt",
   "Ref": ["cve:CVE-2003-0605"],
   "Source": [
      {
         "IP4": ["93.184.216.119"],
         "Proto": ["tcp", "epmap"],
         "Port": [24508]
      }
   ],
   "Target": [
      {
         "Proto": ["tcp", "epmap"],
         "Port": [135]
      }
   ]
}
```

Listing 4. IDEA honeypot event example

Info extracted from spam messages

```
{
   "Format": "IDEA0",
   "ID": "4d52640a-5363-497a-a7d9-bcbde759cb7d",
   "DetectTime": "2014-02-21T16:01:32Z",
   "Category": ["Abusive.Spam"],
   "Description": "Spam URL reference",
   "Source": [
      {
```

```
            "Type": ["OriginSpam"],
            "URL": ["http://www.example.com/"],
            "Proto": ["tcp", "http", "www"]
        }
    ]
}
```


Listing 5. IDEA spam alert example

Blacklists


```
{
    "Format": "IDEA0",
    "ID": "c34bf422-931c-4535-9c6b-257128185265",
    "DetectTime": "2014-11-03T10:33:12Z",
    "Category": ["Vulnerable.Open"],
    "Confidence": 0.5,
    "Description": "Open Recursive Resolver",
    "Source": [
        {
            "Type": ["Open"],
            "IP4": ["93.184.216.119"],
            "Proto": ["udp", "domain"]
        }
    ]
}
```


Listing 6. IDEA blacklist example

Botnet C&C


```
{
    "Format": "IDEA0",
    "ID": "cca3325c-a989-4f8c-998f-5b0e971f6ef0",
    "DetectTime": "2014-03-05T15:52:22Z",
    "Category": ["Intrusion.Botnet"],
    "Description": "Botnet Command and Control",
    "Source": [
        {
            "Type": ["Botnet", "CC"],
            "IP4": ["93.184.216.119"],
            "Proto": ["tcp", "ircu"],
            "Port": [6667]
        }
    ]
}
```


Listing 7. IDEA botnet detection example

Suspicious search on company site


```
{
    "Format": "IDEA0",
    "ID": "b7dd112c-9326-49e6-a743-b1dce8b69650",
    "DetectTime": "2014-02-13T02:21:15Z",
    "Category": ["Recon.Searching"],
    "Description": "Suspicious search",
    "Source": [
        {
            "IP4": ["93.184.216.119"],
            "Proto": ["tcp", "http", "www"]
        }
    ],
    "Target": [
        {
            "URL":
["http://www.example.com/search=%20union%20select%20password%20from%20users%20%2D%2
D"]
```

```
        }
    ]
}
```

Listing 8. IDEA suspicious search example

8.3 Annex C: Additional Technology Scouting

This annex contains additional information on technologies that have been analyzed either because they may be to some extent similar to PROTECTIVE or may be used as one of the existing components at a later stage for analytics. The most significant solutions for PROTECTIVE have been described in the section 5.2. The technologies for implementation and security are described in the chapter 6).

8.3.1 Base Tools and Approaches

8.3.1.1 FACT (Federated Analysis of Cyber Threats)

This project is in certain facets similar to PROTECTIVE in its aims and it the approach of federating existing tools, using a standardised information exchange protocol.

FACT is a research initiative funded by MITRE[125], aiming at building a Cyber Defence Situational Awareness (CDSA) solution targeted towards architecture assessment and incident responses platform. A crucial difference between traditional CDSAs is that they focus on real-time or quasi real-time situational awareness for cyber defenders, while FACT is designed to be used to support the system engineering, recovery, and re-architecting processes required for incident response or acquisition.

FACT integrates the functionality of several efforts conducted by MITRE, especially:

- CRIT[126] (Collaborative Research into Threats) – used to analyze SIEM and sensor data to identify and correlate cyber threat indicators with campaigns (intrusion sets) and threat actors.
- TARA[127] (Threat Assessment and Remediation Analysis) – the solution defines a methodology for cyber architecture assessment in order to identify cyber vulnerabilities and evaluate countermeasure effectiveness
- CyCS[128] (Cyber Command System) – used to assess mission impact based on a mission model reflecting the mission's functional decomposition and allocation to cyber resources

[125] https://www.mitre.org/
[126] https://crits.github.io/
[127] https://www.mitre.org/publications/technical-papers/threat-assessment--remediation-analysis-tara
[128] https://www.mitre.org/research/technology-transfer/technology-licensing/cyber-command-system-cycs

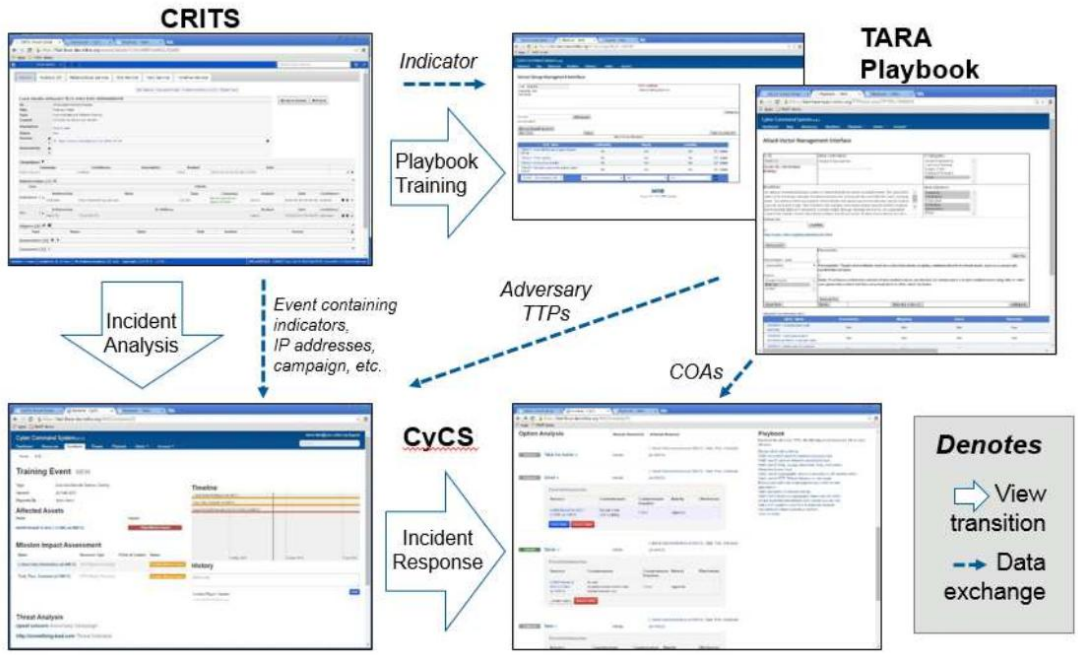


Figure 37. MITRE Federated Analysis of Cyber Threats[129]

MITRE-based solutions inherently use STIX format.

8.3.1.2 CyGraph

CyGraph[130] is another MITRE-based emerging initiative, focused on real-time cyber situational awareness, bringing together isolated data and events into an ongoing overall picture for decision support and situational awareness. It is a tool for cyber warfare analytics, visualization, and knowledge management. It helps prioritize exposed vulnerabilities in the context of mission-critical assets.

For ongoing attacks, CyGraph provides context for correlating intrusion alerts and matching them to known vulnerability paths, and for suggesting optimal countermeasures. For further forensics, the system suggests vulnerable paths that may warrant deeper inspection.

CyGraph incorporates an attack-graph model that maps the potential attack paths through a network. This includes any network attributes that potentially contribute to attack success, such as network topology, firewall rules, host configurations and vulnerabilities. It also incorporates mission dependencies, showing how mission objectives, tasks, and information depend on cyber assets.

Source codes are not available.

8.3.1.3 Open SOC

OpenSOC[131] is a collaborative open source development project dedicated to providing an extensible and scalable advanced security analytics tool. It has strong foundations in the Apache Hadoop Framework and values collaboration for high-quality community-based open source development. OpenSOC originated from Cisco, Inc. It provides the following main features:

The framework provides the following capabilities:

[129] https://www.mitre.org/sites/default/files/publications/pr-15-2592-overview-of-mitre-cyber-situational-awareness-solutions.pdf
[130] https://neo4j.com/blog/cygraph-cybersecurity-situational-awareness/
[131] http://opensoc.github.io/

- Extensible spouts and parsers for attaching OpenSOC to monitor any telemetry source
- Extensible enrichment framework for any telemetry stream
- Anomaly detection and real-time rules-based alerts for any telemetry stream
- Hadoop-backed storage for telemetry stream with a customizable retention time
- Automated real-time indexin for telemetry streams backed by Elastic Search
- Telemetry correlation and SQL query capability for data stored in Hadoop backed by Hive
- ODBC/JDBC compatibility and integration with existing analytics tools

OpenSOC consists of two repositories:

- OpenSOC-Streaming: modules for processing, enriching, indexing, and correlating data, PCAP reconstruction service, and other data services.
- OpenSOC-UI: user interface for log and network packet analytics, displaying alerts and errors.

The source code is available under Apache 2.0 License. However, since 2015 the source codes have not been updated.

8.3.1.4 Apache Metron

Apache Metron[132] provides a scalable advanced security analytics framework built with the Hadoop Community, evolving from the Cisco OpenSOC Project. It composes a cyber-security application framework that provides organizations the ability to detect cyber anomalies and enable organizations to rapidly respond to identified anomalies.

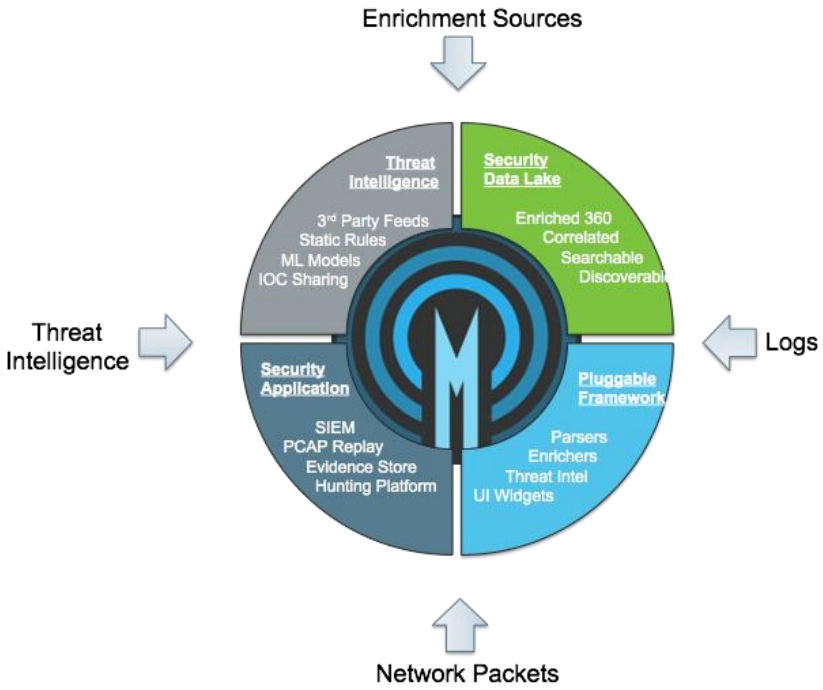


Figure 38. Apache Metron Core Capabilities[133]

The Metron framework core capabilities are concentrated on:

- Security Data Lake – storing acquired data for a long period of time for further analysis

[132] http://metron.apache.org/
[133] https://cwiki.apache.org/confluence/display/METRON/Metron+Core+Capabilities+and+Functional+Themes

- Pluggable Framework – provides e.g. set of parsers and allows to connect enrichment services as well as dashboard customization capabilities
- Security Application – standard SIEM functionalities, extended by evidence gathering, packet replay utilities and hunting services
- TI Platform – anomaly detection and machine learning algorithms for realtime streaming analysis

In functional terms, Metron focuses on the following aspects:

- Platform performance, scalability, extensibility and maintainability
- Data Collection
- Data Processing (based on Storm Topologies) oriented towards performing crucial actions (e.g. normalization, enrichment, alerting, indexing) in realtime
- User Interface

8.3.2 Other Data Sinks, Probes and Filtering Software

8.3.2.1 Palo Alto Networks MineMeld

Palo Alto (PA) MineMeld[134] is an open-source application that streamlines the aggregation, enforcement and sharing of TI. It is available as the source code or predefined VM images. It has an extensible, modular architecture. MineMeld covers a set of interesting scenarios related with PROTECTIVE goals, like:

- Aggregation and correlation of TI feeds
- Share indicators with trusted peers (PA has created a TI sharing ecosystem with several partners e.g. Spamhaus, that may be joined by other partners)
- Evaluation of a particular TI feed for the particular environment
- Enforcement of prevention controls like IP blacklists

MineMeld consists of two main components:

- Core component
- WebUI – Web user interface

The two components communicate using the API provided by the core. The Core itself contains three key services:

- engine – the main working process
- API – service for monitoring the engine status and reporting to the WebUI
- traced daemon – service for storing traces of messages processed by engine

The 3 services communicate using a simple RPC mechanism over an external message broker, RabbitMQ. Redis is used for streaming long answers and for retrieving output feeds from the engine:

[134] https://www.paloaltonetworks.com/products/secure-the-network/subscriptions/minemeld

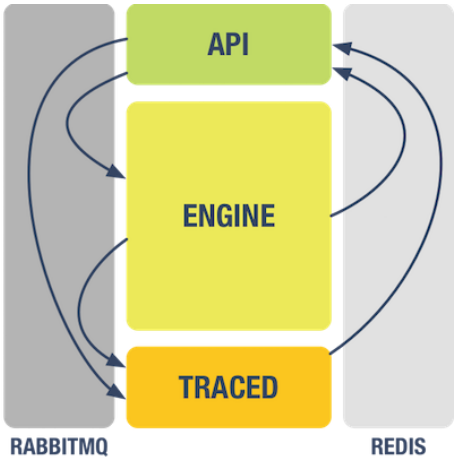


Figure 39. MineMeld core internal flow[135]

The MineMeld engine stores metrics for each of its nodes in a RRD database using collectd.

MineMeld is written almost solely in Python. As the product is continuously contributed by communities, it will be monitored by PROTECTIVE consortium and cooperation or integration may be considered in the future.

8.3.2.2 Mcafee ® Open DXL

A conceptually similar solution to MineMeld is Mcafee® Open DXL. Open Data Exchange Layer has been provided to enable security devices to share intelligence and orchestrate security operations in realtime. It leverages the McAfee Data Exchange Layer (DXL), which many vendors and enterprises already utilize, and delivers a simple, open path for integrating security technologies regardless of the vendor.

The DXL architecture allows DXL clients (services) to communicate with each over the message bus nearly in realtime. Applications publish and subscribe to message topics, or make calls to DXL services in a request/response invocation similar to RESTful APIs. The fabric delivers the messages and calls immediately.

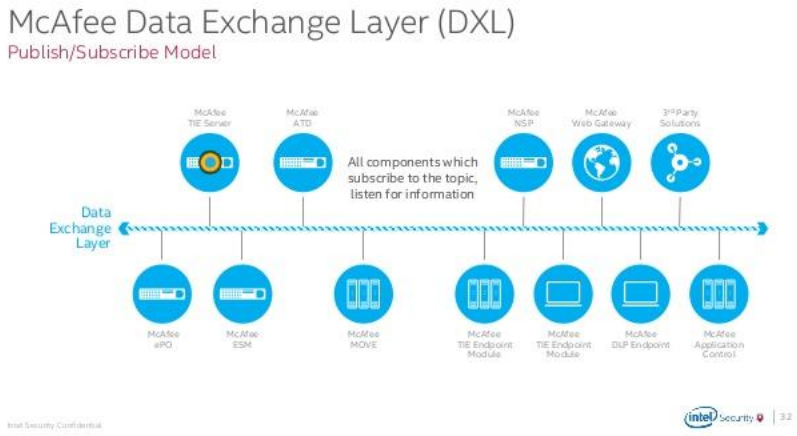


Figure 40. McAffee DXL in publish/subscribe model (source: Intel Security)

The main architecture elements are:

- Message bus

[135] https://github.com/PaloAltoNetworks/minemeld/wiki/Architecture

- Brokers that are responsible for routing messages between clients connected to the message bus
- Clients that connect to brokers for the purposes of exchanging messages. Communication between brokers and clients is over a TLS-based connection with bi-directional authentication (PKI).
- McAfee ePO (ePolicy Orchestrator) is used to manage McAfee products, including DXL, maintaining its fabric topology information and authorization rules for the fabric, and provides views for visualizing the fabric's current state.

Software, written in Python, is available at https://github.com/opendxl.