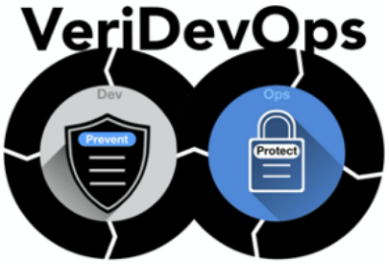


# VeriDevOps

*Automated Protection and Prevention to Meet Security*

*Requirements in DevOps Environments*

## D 3.3. Security monitoring - security flaws detection mechanisms and tools initial version

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957212. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.*


| | |
|---|---|
| **Contract number:** | 957212 |
| **Project acronym:** | VeriDevOps |
| **Project title:** | Automated Protection and Prevention to Meet Security Requirements in DevOps Environments |
| **Delivery Date:** | 31/12/2021 |
| **Coordinator:** | MONT |
| **Partners contributed:** | IKER, ABO |
| **Release Date:** | 31.3.2021 |
| **Version:** | 01 |
| **Revision:** | IKER, ABO |
| **Abstract:** | This deliverable provides an overview of the initial version of the security flaws detection mechanisms, implemented as monitoring solutions during the operation phase at the VeriDevOps project. |
| **Status:** | PU (Public) |

# TABLE OF CONTENTS

# Glossary of terms and abbreviations used

| Abbreviation | Description |
|---|---|
| AIDS | Anomaly-based Intrusion Detection Systems |
| AI | Artificial Intelligence |
| CVSS | Common Vulnerability Scoring System |
| DL | Deep Learning |
| DPI | Deep Packet Inspection |
| ESR | Engine Specific Requirement |
| FN | False Negative |
| FP | False Positive |
| ICS | Industrial Control System |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| KPI | Key Performance Indicator |
| LPS | Load Positioning System |
| MAC | Media Access Control |
| ML | Machine Learning |
| MMT | Montimage Monitoring Tool |
| NN | Neural Network |
| OT | Operation Technology |
| PLC | Programmable logic controller |
| RCO | Root Cause Analysis |
| SIDS | Signature-based Intrusion Detection Systems |
| TN | True Negative |
| TP | True Positive |


## Executive Abstract

This deliverable provides an overview of the initial version of the security flaws detection mechanisms, implemented as monitoring solutions during the operation phase of a critical industrial system. The overview builds upon and extends the rule-based security analysis, ML/AI-based anomaly detection and root cause analysis mechanisms implemented in the industrial tool MMT (Montimage Monitoring Tool) provided by Montimage. In addition, it depicts the general architecture of the monitoring solution, the requirements to be fulfilled by each of its composing engines, their functionalities, their current status of implementation, and their integration in the use cases provided by ABB and FAGOR partners.

# Introduction

The VeriDevOps project assumes a DevOps development process, starting from the initial security requirements of the system under test, ensuring preventive security at development and last but not least providing reactive protection at operations. WP3: Reactive Protection at Operations will incorporate security practices into operational environments. Our objective is to integrate operation and security activities to perform dynamic risk management in running environments based on the security properties formalised in WP2.

The problem of risk management first requires a risk assessment process that normally is addressed in several steps: preparation of the assessment, identifying threats sources, identifying threats events, identifying vulnerabilities in the system, determining the likelihood that the sources of the threats would exploit a specific vulnerability in the system, quantify the Impact that an exploitation a specific vulnerability would have in the system, determine the risk value and the uncertainty of its value, communicate risk assessment results, maintaining the risk assessment [1]. In VeriDevOps, the vulnerability identification mechanisms will be described in D3.1: Threat oracle engine specification, design and implementation initial version; while this deliverable will cover the preparation of the assessment, identification of threats sources, and identification of threats events tasks. Future steps of the risk assessment process will be covered in the final version of this deliverable, while the rest of the procedures of the risk management process will be covered in D3.5: Attack response - Root cause analysis and countermeasures initial version, as well as, D3.7: Reactive protection at operations.

The current deliverable D3.3 is divided into the corresponding parts:

- the *motivation* section, in which we revisit a general state-of-the-art on each monitoring of industrial control systems
- the *preliminaries* section, where we discuss the basic concepts on which the rest of the document relies on
- the *requirements and targeted KPIs* section, where we state the requirements and KPIs to be fulfilled by the engines
- the *architecture* section, in which we describe the global architecture, as well as, the functionalities, workflows and interaction with external tools of its components

- the *implementation status* section, where we provide details about the current implantation state of the tools
- the *usage in case studies* section when we described the current and potential application of the tools in the case studies

# 1 Motivation

Managing security risk during operation is one of the main objectives of dynamic risk management that targets evolving ICT systems and evolving risk landscape. In this sense, security at runtime has become an important part of today's software development projects where different pieces of code from several providers can be used and integrated into evolving ICT environments [2]. In particular, in ICS the cyber-physical infrastructures must guarantee the protection of the traditional (physical) elements, such as sensors, controllers and actuators; as well as the novel (cyber) capabilities, in terms of computing and communication protection[3]. The topic has been attracting increasing attention since the Stuxnet incident when the successful cyber-physical sabotage of a uranium enrichment plant in Iran took place [4]. More recently, the coronavirus pandemic has increased the interest of industry actors on operation technology (OT) cybersecurity. According to Kaspersky's research, the number of phishing and cyberattacks related to coronavirus since March went up. Moreover, its report of the state of industrial cybersecurity in the era of digitalization[1] showed that in many industrial companies the pandemic has affected cybersecurity priorities (see Figure 1.1).

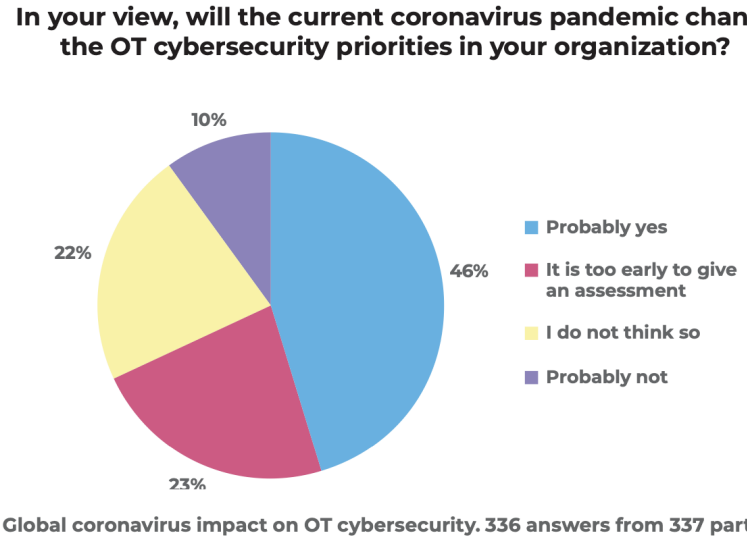


Figure 1.1 Change in the OT cybersecurity in industrial companies since the coronavirus pandemic[1]

[1] https://ics-cert.kaspersky.com/media/Kaspersky_ARC_ICS-2020-Trend-Report.pdf

Procuring cybersecurity in ICS becomes more critical over time because of the imperfection of the existing protection tools, and the increasing presence of vulnerabilities. For example, in 2018 the number of vulnerabilities identified in different ICS components and published on the US ICS-CERT[2] website was 415, 93 vulnerabilities more than in 2017. Even more, compared with the previous year's data, the proportion of vulnerabilities that have a high or critical severity score has grown. More than half of the vulnerabilities identified in ICS systems (284, compared with 194 in the previous year) were assigned CVSS v.3.0[3] base scores of 7 or higher, corresponding to a high or critical level of risk. Kaspersky[4] has also reported the main challenges according to industrial companies. Figure 1.2 depicts that companies are firstly concerned for injury or death of employees; as well as, damaging of products and services, and loss of proprietary or confidential information, due to cybersecurity incidents.


Figure 1.2 Cybersecurity challenges described by industrial companies in 2020[4]

Therefore, it is clear that in the current context of 2021 cybersecurity is a priority in industrial systems, and it affects factors as fundamental as the lives of employees or the

[2] https://us-cert.cisa.gov/ics

[3] https://www.first.org/cvss/calculator/3.0

[4] https://ics-cert.kaspersky.com/media/Kaspersky_ARC_ICS-2020-Trend-Report.pdf

quality of a company's products. Nevertheless, improving security requires that relevant actors inside the company acquire relevant knowledge and skills to guarantee security at deployment and operation phases, such that a system can resist attacks and handle security errors appropriately [5]. They also need to be supported by tools to ensure dynamic risk management techniques [6], [7], for the automatic detection of vulnerabilities and their mitigation at runtime; as well as, stay informed about relevant security standards, such as the ISO/IEC 31010 standard, that provides a comprehensive overview of risk management techniques[8], or the ISA/IEC 62443 that offers a system risk-oriented approach to solving the tasks of providing the security of industrial control systems (ICS) at all stages of the life cycle.

SecOps (Security + Operations)[5] is a movement created to facilitate collaboration between IT security and operations teams and integrate the technology and processes they use to keep systems and data secure — all in an effort to reduce risk and improve business agility. As illustrated in Figure 1.3, the WP3 of the VeriDevOps project will incorporate security practices into operational environments.

Concretely, VeriDevOps WP3 proposes, besides a vulnerability scanner to find vulnerabilities at the operation phase of a system (IKERLAN THOE tool presented in D3.1 deliverable), a monitoring solution, that will combine rule-based detection, and machine learning techniques to guarantee security at the operation phase, by triggering alerts when a potential incident is detected.

VeriDevOps will innovate by fully investigating diversity, prevention and tolerance combined with different risk management techniques that can be applied to industrial systems during their operation. Most of these techniques are applicable within a wide range of domains within cybersecurity.

[5] https://saltproject.io/the-power-of-secops-redefining-core-security-capabilities/

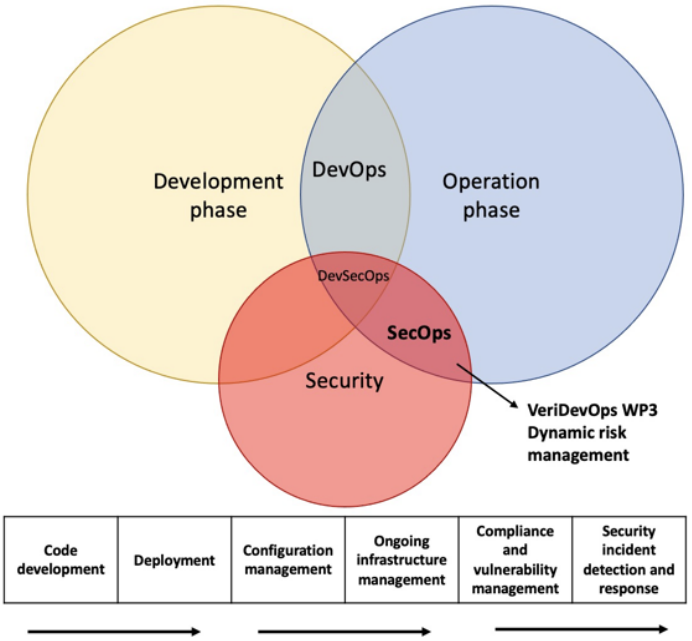


Figure 1.3 Context of the dynamic risk management process done in the VeriDevOps WP3

The project addresses updating the risk management at run-time based on data that is continuously collected through monitoring, during the operation phase of systems. This is not addressed, in particular, by traditional current risk management techniques and notations. Moreover, in each stage of the risk management process, the project aims to propose new approaches beyond the state-of-art, and to test them in actual industrial systems, in order to generate reliable new techniques in this field.

Moreover, in complex systems, determining the causal factors of observed anomalies can be drastically difficult and time-consuming due to the exceeding data sources (e.g., logs, traffic, metrics) identifying the status of the system. To address this issue, VeriDevOps will propose a knowledge-based Root Cause Analysis tool to systematise the experience in dealing with faults and problems, perform an accurate diagnostic of a newly detected security incident and automatically deploy the relevant security mechanism (according to the ratio impact over cost) according to the application running environment.

The combination of these tools will lead to bridging the gap of security in industrial systems during their operation. These tools will be validated in the two use cases in VeriDevOps to show effectiveness.

# 2 Preliminaries

## 2.1 Risk Management

Risk management refers to a methodology to quantify vulnerabilities, threats and the probability of their explorations provoking undesired effects in the system. [1]. Inside the risk management methodology, there are the following basic definitions:

- An **asset** is a company resource that is intended to be protected through the risk management process
- A **vulnerability** is a security gap, weakness, or error in the protection efforts that can lead to a threat
- A **threat** is the exploitation, intentionally or accidentally, of a vulnerability that may cause damage to an asset. Threats are what the risk management process attempts to protect against
- **Risk** is the potential damage of an asset that a threat may cause. Normally, it is calculated as the multiplication of the probability for a threat to be exploited by the impacts that this weakness exploitation would have in the related assets

The problem of risk management first requires a risk assessment process that normally is addressed in several steps [1]:

1. Preparation of the assessment, that included identifying the objectives of the risk assessment procedure, its scope, the architecture and technology that will be utilised, the assumptions and constraints that will be considered, the potential sources of threats and vulnerabilities in the system, and the risk model and approach to be enforced
2. Conducting the risk assessment, which includes
    a. Identifying threats sources
    b. Identify threats events
    c. Identify vulnerabilities in the system
    d. Determine the likelihood that the sources of the threats would exploit a specific vulnerability in the system
    e. Quantify the Impact that an exploitation of a specific vulnerability would have on the system


f. Determine the risk value and the uncertainty of its value

3. Communicate risk assessment results, with the relevant organisational personnel
4. Maintaining the risk assessment, by monitoring risk factors to keep updated the risk computations

### 2.1.1 Security Monitoring

Security monitoring is one key activity in risk management, and in particular it is used in the risk assessment proceeding to identify threats events. In general, monitoring can be defined as the action of observing or inspecting a system at different points, in order to produce reports, notify an abnormal operation, draw operation baselines, provide inputs to the network or application management, etc. Security monitoring attempts to monitor a network or systems to detect malicious activity or policy violations. There is two main approaches to perform security monitoring:

- Signature-based detection (recognizing bad patterns, such as malware)
- Anomaly-based detection (detecting deviations from a model of "good" traffic, often relies on machine learning)

### 2.1.2 Rule-base monitoring

Signature-based Intrusion Detection Systems (SIDS) monitor events in a system and compare them with patterns and signatures of security policies or rules to be respected, or attacks and vulnerabilities to be avoided that exist in a database. Therefore, if one of these previously recorded behaviours is detected, an alert is triggered. In general, this approach has the advantage of presenting a very low false-positive alarm rate compared to others [9]. Nevertheless, as the attacks are previously defined in a specific manner, a simple modification of the attack could make it undetectable by the engine. Furthermore, other problems still remain such as a choice of techniques and algorithms to accurately and efficiently detect malicious behaviours and intrusions [10], and how to identify attacks that span across several packets [11].

Currently, there are numerous SIDS in the market, some examples include McAfee NSP[6], Palo Alto Networks[7], SolarWinds SEM[8], etc. Moreover, there are available open-source solutions such as Snort, Suricata, and Zeek, before called Bro.

### 2.1.3 ML Anomaly detection

Anomaly-based Intrusion Detection Systems (AIDS) work by comparing the actual comportment of the system with a previously-established "normal" model of the behaviour of the system. Any substantial deviance between the observed behaviour and the model is considered as an anomaly, which can be translated as an intrusion or attack into the system. AIDS has drawn interest from a lot of scholars due to its capacity to overcome the limitation of the Signature-based intrusion detection systems (SIDS) [11].

Normally, in AIDS, the normal model of the behaviour of a computer system is created using machine learning, statistical-based or knowledge-based methods. For the purposes of the VeriDevOps project, we will consider only the machine learning method.

The desire to use Machine Learning algorithms in various disciplines is constantly growing. Moreover, the increasing popularity of ML techniques has led to a creation boom of different ML libraries and frameworks, which can be applied to a multitude of problems, from image recognition to network traffic classification [12] [13]. The applications of machine learning techniques in the design of intrusion detection systems (IDS) have remained a trend in the last few years [14]. Therefore, there have been numerous anomaly-based IDS prototypes that implement these techniques.

We can distinguish several categories of ML tools that differ from one another in purpose, programming language implementation, usage of hardware support, availability of different programming languages support, being open-source, etc. These types are [12] [15]: general-purpose ones (sklearn); mathematical and statistical (scipy, numpy); interactive platforms (Tableau); libraries dedicated to specific ML methods (LibSVM); big data-oriented (SparkML); deep learning with hardware support (Tensorflow, Theano).

It can be noticed that those tools can vary from being a specific library to being a whole ecosystem. Such a vast number of different ML tools can be a great showcase of the fact that artificial intelligence is a constantly evolving discipline [13]. However, while the progress in creating methods and tools that are more and more precise is promising, in

[6] https://www.mcafee.com/
[7] https://www.paloaltonetworks.com/
[8] https://www.solarwinds.com/security-event-manager

many disciplines, especially industry-based ones, the application of ML techniques is still slow, and switching to testing ML methods takes a long time and much effort [12].

Usage of ML techniques in such practical fields is challenging for four main reasons [12]:

1. Need for an expert level of understanding ML libraries such as Keras, Tensorflow, in order to start building any model.
2. Time requirement connected with Machine Learning model prototyping, tuning, implementation, and afterward interpretation of the results.
3. Systems Interoperability problem appearing due to the fact that many enterprises already use some data-gathering systems, hence the process of connecting existing systems in order to transform the data into a form that is utilisable by an ML algorithm can be tricky; moreover, feature engineering and extraction also takes significant effort for enterprises not fluent in ML.
4. Necessity of experimentation with different ML methods and their parameters: in many cases, the initial prototype results are not sufficient, which can lead to the false conclusion that using ML is not beneficial or not advanced enough to be used in a particular discipline.

Due to these challenges many industries are still reserved towards artificial intelligence methods, despite the fact that utilising ML methods can reduce production cycle time, improve resource utilisation and help to discover implicit, unknown before knowledge [16]. In particular, in the context of VeriDevOps case studies, that are industrial systems, this challenges are relevant as (i) in general, operators of these type of systems are not specialised in ML techniques nor data analysis, (ii) the nature of these systems is highly heterogeneous; therefore, extracting data and relevant features to be used by ML algorithms is not trivial.

### 2.1.4 Root Cause Analysis

Root Cause Analysis (RCA) is a general term used in different domains, namely IT operations, telecommunications, manufacturing industry, medical diagnosis, and healthcare industry. RCA is defined as “a systematic process for identifying “root causes” of problems or events and for responding to them” [17].

In the context of VeriDevOps, we focus the analysis on Information and Communication Systems to infer the root causes of problems by analysing the causal chains

governing the system under monitoring. RCA plays a vital role in the Risk Management process which principally includes Vulnerability Scanning, Anomaly Detection, Root Cause Analysis, and Remediation. System administrators and DevOps engineers use RCA not only for detecting the problems but also for understanding their root causes to prevent the recurrences and/or mitigate the impact.

## 2.2 KPIs and performance metrics definitions

In this section we present metrics to define the KPIs described in Section 3, that will be used to measure the performance of the VeriDevOps solutions proposed in this deliverable. The metrics that we planned to apply are defined as follows:

### 2.2.1 Accuracy

Accuracy is the closeness of a measured value to the true quantity value [18]. Therefore, is the proportion of correctly classified or predicted values among the total of inspected cases. The accuracy is a good general performance measure; nevertheless, it can be easily biassed in the presence of unbalanced datasets, hence it must be considered together with other performance metrics such as the described below in this section.

$$Accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ positives + false\ negatives}$$

### 2.2.2 Recall

Recall or true positive rate can be interpreted as a rate of discovery of real positives. It is defined by the proportion of **real positive** values that were correctly classified or predicted [19] as positive. When the application is different from the binary case, there is a recall measure for each of the classes or categories in the dataset.

$$Recall = true\ positive\ rate = \frac{true\ positives}{real\ positives} = \frac{true\ positives}{true\ positives + false\ negatives}$$

### 2.2.3 Precision

Precision is a measure of accuracy of the predicted positive cases. It is defined by the proportion of **predicted positive** values that were correctly classified or predicted [19].

When the application is different from the binary case, there is a precision measure for each of the classes or categories in the dataset.

$$Precision = true\ positive\ rate = \frac{true\ positives}{predicted\ positives} = \frac{true\ positives}{true\ positives + false\ positives}$$

### 2.2.4 F-measure

F-score is an accuracy measure considering precision and recall values. It is the harmonic mean of precision and recall and it provides a better measure of correctly classified values, in the case of unbalanced datasets.

$$F - measure = 2\frac{precision * recall}{precision + recall} = \frac{true\ positives}{true\ positives + \frac{1}{2}(false\ positives + false\ negatives)}$$

### 2.2.5 Confusion matrix

Confusion matrices are commonly used in machine learning to present an inventory of the results of a classification problem. Figure 2.2.4.1 depicts the content of a confusion matrix, that includes the true positives and negatives; as well as, the false positives and negatives, per class in the dataset.

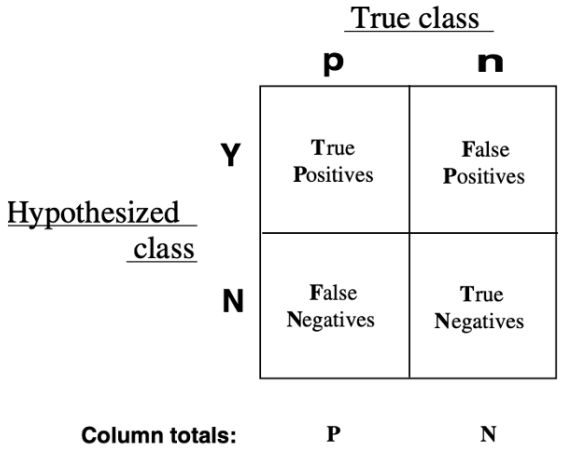


Figure 2.2.4.1 Confusion matrix [20]

### 2.2.6 Similarity score

Suppose that the temporal state of the system can be reflected by ***n*** metrics (i.e., ***n*** attributes). This set of ***n*** attributes can be represented by a vector in a multi-dimensional space of ***n*** dimensions. Calculating the similarity/ dissimilarity of two states becomes the problem of measuring the distance of orientation (the angle) and of magnitude (the length) of their two representing vectors. Figure 2.2.6.1 depicts an example in a 3-dimensional space.

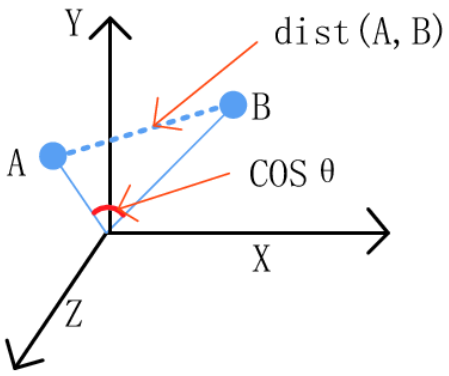


Figure 2.2.6.1: Similarity calculation in 3-dimensional space.

# 3 Requirements and targeted KPIs

The table below provides the list of the VeriDevOps Monitoring Solution modules requirements. For measuring the metrics mentioned in the table, we assume the following hardware resources:

For processing **less than 100 Mbp** (megabit per second):

1. 2 cores of CPU
2. 2 GB of RAM
3. 4GB of free Solid State Drive
4. 2 NICs: one for capturing network traffic and another for administering

For capturing **more than 100 Mbp until 1Gps**:

1. 16 cores of CPU
2. 32 GB of RAM
3. 20 GB of free space Solid State Drive
4. 2 NICs: one for capturing network traffic and another for administering. The capturing NIC should be either Intel X710 or Intel X520 card

Moreover, the experiments to obtain the measure will be repeated at least 100 times. Other factors that will influence the KPI measurements are exposed on the column *Context* of the table 3.1.

| ID | Requirement | | | Related KPIs | KPI value | Context |
|---|---|---|---|---|---|---|
| | Object | Commitment | Action + Subject + (constraint) | | | |
| General requirements | | | | | | |
| ESR-1 | The VDO monitoring solution | SHALL | monitor continuously (i.e., capture and analyse) the relevant network traffic and log applications at runtime to detect potential security incidents in the context of VeriDevOps case studies | Accuracy | Between 90% and 96% | Size and quality of the dataset, monitoring technique used (rule-based or anomaly-based) |

| | | | | | | |
|---|---|---|---|---|---|---|
| ESR-2 | The VDO monitoring solution | SHOULD | detect known attacks at runtime in the context of VeriDevOps case studies | Detection time | ~ less than 1 second, until 60 seconds | Hardware Running of the case studies |
| ESR-3 | The VDO monitoring solution | SHOULD | provide a list of potential threats in the context of VeriDevOps case studies | Number of potential threats | ~ more than 3 attributes | Running of the case studies |
| ESR-4 | The VDO monitoring solution | SHALL | identify the root causes of the incidents if similar events are available in the history in near real time, in the context of VeriDevOps case studies | Response time | Less than 5 seconds | Running of the case studies |
| ESR-5 | The VDO monitoring solution | SHALL | recommend corrective actions to manage identified risks in the context of VeriDevOps case studies | Time | Less than 60 sec | Running of the case studies |
| Rule-based monitoring requirements | | | | | | |
| ESR-6 | MMT-Extract | SHALL | retrieve relevant security attributes from system and application traces in the context of VeriDevOps case studies | Number of security attributes | ~ more than 10 attributes | Running of the case studies |
| ESR-7 | MMT-Security | SHALL | detect incidents registered in its database in the context of VeriDevOps case studies | Detection time | Less than 1 sec | Running of the case studies |
| ESR-8 | MMT-Security | SHOULD | rely on adequate security properties that are specified in XML and inspired from LTL in the | Number of security properties specified | ~ more than 5 | Given specifications of security properties |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | context of VeriDevOps case studies | | | |
| Machine Learning framework requirements | | | | | | |
| ESR-9 | MMT-ML | SHALL | extract relevant features for a deep insight of the networks or applications of the case studies | Number of relevant features | ~ more than 100 | Running of the case studies |
| ESR-10 | MMT-ML | SHOULD | be able to detect potential cyber-attacks and anomalies | Detection time | ~ less than 1 second, until 3 minutes | Running of the case studies |
| | | | | Accuracy | Between 90% and 96% | ML algorithm, Size and quality of the dataset |
| ESR-11 | MMT-ML | SHALL | implement and provide different ML techniques | Number of implemented techniques | ~ more than 3 machine learning techniques | Analysis in the context of the case studies |
| ESR-12 | MMT-ML | SHALL | provide exemplary trained models, tested for use-cases | Size and quality of the dataset | 20% of improvement | % of improvements in the context of case studies |
| ESR-13 | MMT-ML | SHALL | provide the results of the application of MMT/ML to the use cases | Detection time | Less than 3 minutes | Running of the case studies |
| Root cause Analysis requirements | | | | | | |
| ESR-14 | MMT-RCA | SHALL | determine accurately the root causes of the recurring incident | Accuracy | ~ more than 95% | Algorithms, Size and quality of the dataset |

| | | | | | | |
|---|---|---|---|---|---|---|
| ESR-15 | MMT-RCA | SHOULD | find a high similarity score between an incident and its recurrence | Similarity score | ~ more than 90% | Algorithm, Size and quality of the dataset |
| ESR-16 | MMT-RCA | SHALL | observe the evolution of the incident before, during, and after it happens as well as predict its impact in near-real time. | Response time | ~ less than 5 seconds | Running of the case studies |

Table 3.1 WP3 monitoring solution requirements and KPIs

# 4 Architecture

The VeriDevOps monitoring solution during operations is composed of five main modules, depicted in Figure 4.1. The main architecture takes network traffic or application logs as inputs (and in general any structured data), the feature extraction module extracts relevant events, such as network protocol field values, QoS metrics, etc, that therefore are analysed by using three different techniques: rule-based monitoring, ML/AI-based anomaly detection and root cause analysis. Finally, the results are shown in a dashboard with relevant charts that resume the status of the network or application and depict alerts when security flaws are suspected or detected. Section 4.1 describes each of the modules that compose the main architecture.

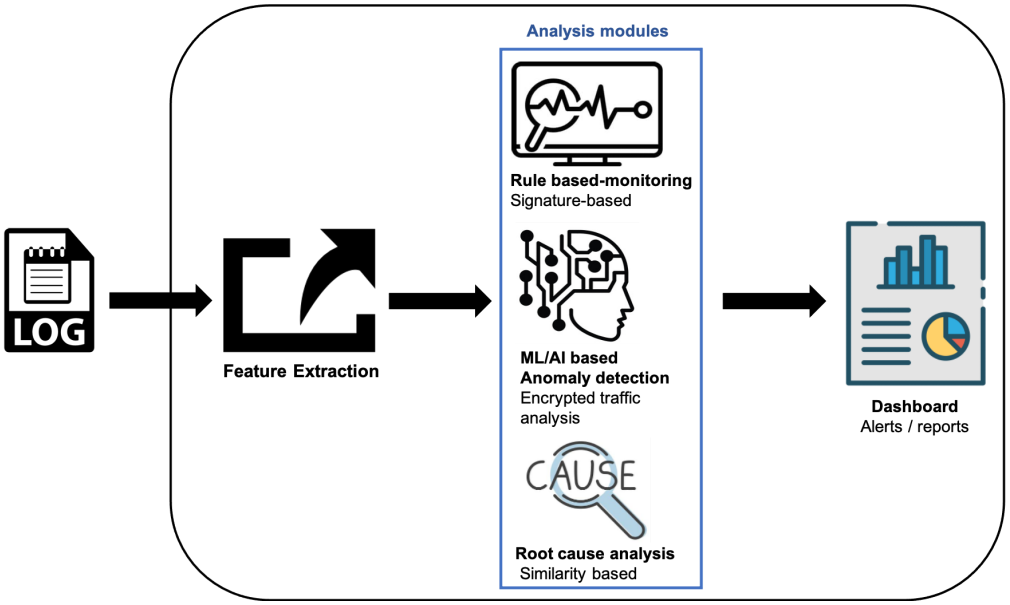


Figure 4.1 Security flaws detection mechanisms architecture

## 4.1 Functionalities

### 4.1.1 Feature Extraction library

MMT-Extract is a C library that parses network traffic and application logs, to extract network and application based events, such as protocol field values. Moreover, the library uses Deep Packet Inspection (DPI) techniques to parse the payload of hundreds of application protocols, when analysing network traffic.

MMT-Extract allows parsing a wide range of network protocols (e.g., TCP, UDP, ARP, HTTP, etc.) and extracting various performance indicators. The extraction is powered by a plugin architecture that allows adding new protocols or format of application messages to be analysed. In the case of network monitoring, prior to extracting protocol packet attributes, the library uses DPI techniques for application identification and classification. This is essential when analysing applications that do not have an standard port number (e.g., P2P, Skype). To be able to classify encrypted packets such as Skype, both signature detection and flow state are used.

The Figure 4.1.1.1 shows how a User Program interacts with the library. It captures events that are processed by the library's Event Processing Engine. The library will parse the event using the integrated or added Plugins, classify it to determine what type of message it is, extract the needed data from it and send notifications back to the User Program.

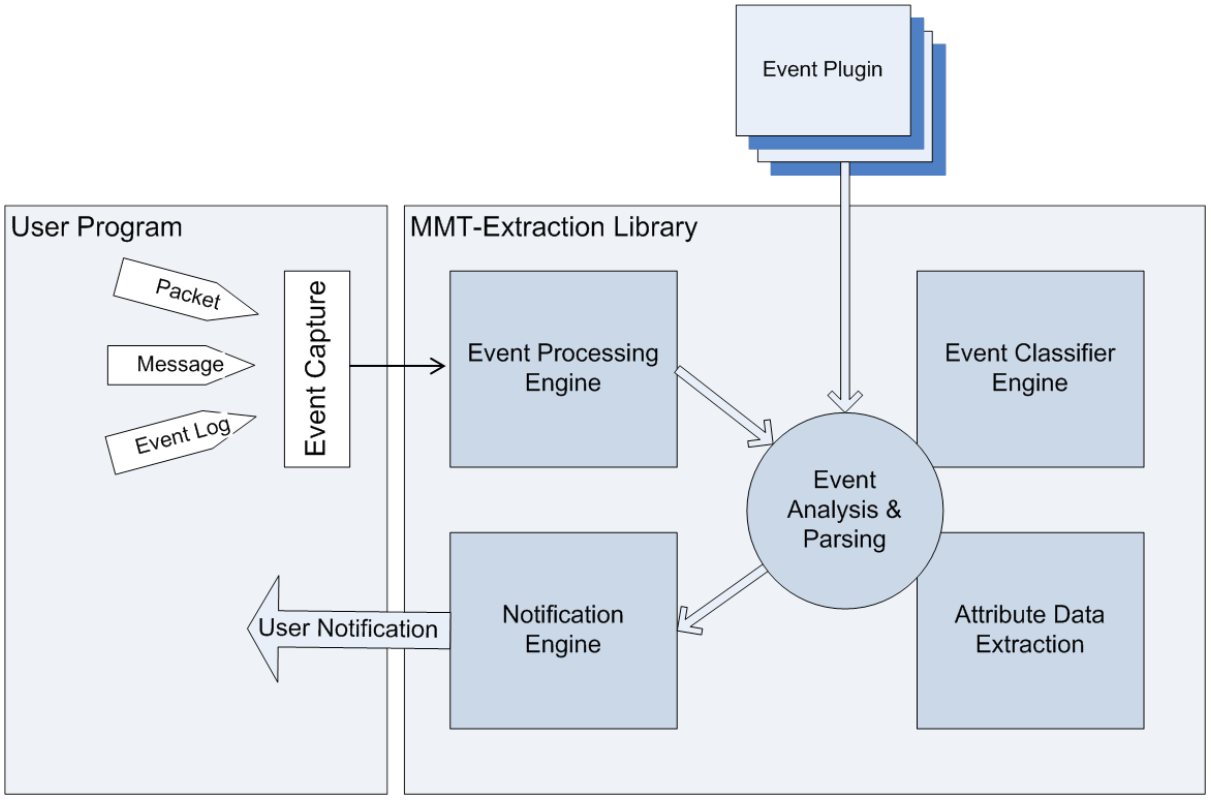


Figure 4.1.1.1: MMT-Extract Feature extraction library

### 4.1.2. Rule-based monitoring

MMT-Security is an event-based monitoring solution that allows analysing network traffic according to a set of security properties referred to as MMT-Security properties. The main objective of these properties is to formally specify security goals and attack behaviours related to the application or protocol that is being monitored. The MMT-Security property model is inspired by Linear Temporal Logic and can refer to two types of properties:

- Security rules that describe the normal, legitimate behaviour of the application or protocol under analysis. In consequence, the non-respect of the property indicates an anomaly that could be a potential violation of a functional or security requirement. ; e.g., all the ports in a computer must be closed unless they are being used by an authorised application.
- Attacks that describe malicious behaviour corresponding to an attack model, a vulnerability or misbehaviour. In this case, the respect of the property indicates the detection of potential incident; e.g. a big number of requests in a short period of time could be a behavioural denial-of-service attack.

XML format was chosen as the language of MMT-Security properties, due to its simplicity and straightforward structure verification. Each property begins with a <property> tag and ends with </property>. A property is a "general ordered tree" as shown in Figure 4.1.2.1, where there are property nodes that are required, operator nodes that are optional, and event nodes that are required. The property node must be the root node, while the event nodes are necessarily leaf nodes.

Each property is composed of a context, in the left branch, and a trigger, in the right branch. Then a property is valid when the trigger is valid, and the trigger is inspected only if the context is valid. In other words.

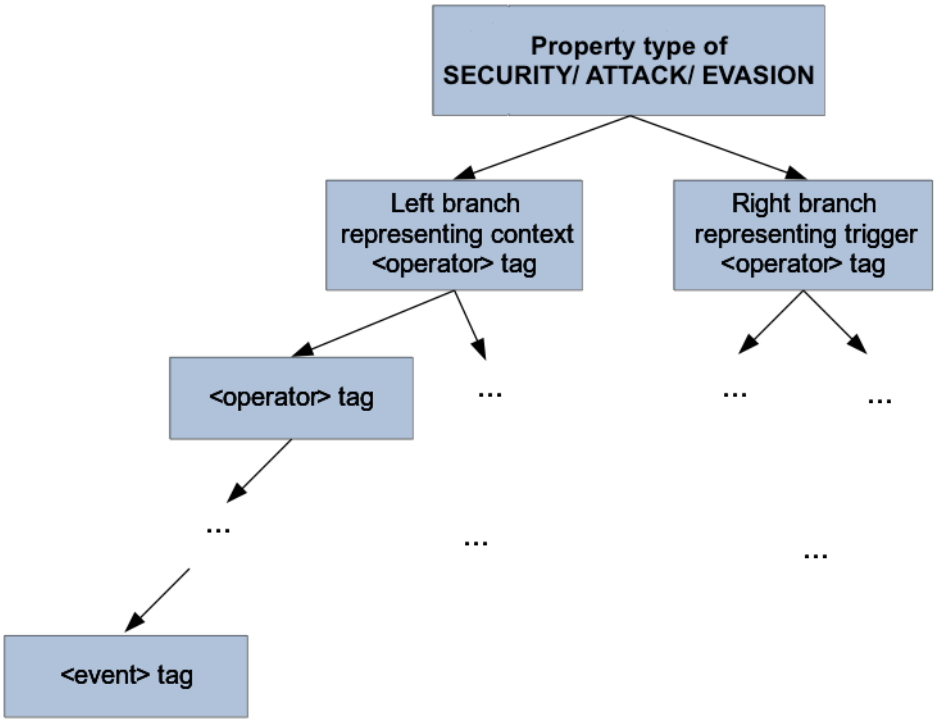


Figure 4.1.2.1 MMT Security property structure [21]

Table 4.1.2.1 illustrates a security property depicted in Figure 4.1.2.2 and derived from the case study described in section 6.1. The rule describes a potential attack to be avoided. The rule context is a detected packet with a type identifier equal to 7, and the alert is triggered if the packet payload contains an *ABB confidential variable* different than 3.

```
<beginning>
<!-- Property 90: Number [redacted] different of 3.-->

<property value="THEN" delay_units="ms" delay_min="0" delay_max="0" property_id="91" type_property="ATTACK"
    description="Number [redacted] different to 3">
<event value="COMPUTE" event_id="1"
    description="[redacted]"
    boolean_expression="(ABB_[redacted].type == 7)"/>
<event value="COMPUTE" event_id="2"
    description="Number [redacted] different to 3"
    boolean_expression="(ABB_[redacted]_number != 3)"/>
</property>
</beginning>
```

Figure 4.1.2.2 MMT Security property structure

| XML code | Explanation |
|---|---|
| `<property value="THEN" delay_units="ms" delay_min="0" delay_max="0" property_id="91" type_property="ATTACK"` | Potential attack (that should be avoided) with id=90 of the form: *if the context holds then at least X time unit (*`delay_units`*) before* (`delay_min`) and until Y time of unit after (`delay_max`) *the trigger should have occurred* |
| `description=" ">` | Textual description of the property |
| `<event value="COMPUTE" event_id="1"` | Event with id=1 that occurs when the boolean expression computes to true |
| `description=" "` | Textual description of the event |
| `boolean_expression="(ABB_xxx.type== 7)"/>` | Boolean expression. Note that the data identified by `ABB_xxx.type` is captured when this event occurs. |
| `<event value="COMPUTE" event_id="2"` | Event with id=2 that occurs when the boolean expression computes to true |

| | |
|---|---|
| `description=" "` | Textual description of the event |
| `boolean_expression="(ABB_xxx_number != 3)"/>` | Boolean expression. Note that the data identified by `ABB_xxx_number` is captured when this event occurs. |
| `</property>`<br>`</beginning>` | End of property |

Table 4.1.2.1 Syntax of MMT-Security rules

The originality of the MMT security properties with respect to existing intrusion detection tools is that it is not based on just signatures matching, which is the case of SNORT [22], nor requires writing executable scripts, such as BRO [23]. MMT security rules syntax allows a more abstract description of a sequence of events that can represent normal/abnormal behaviour.

### 4.1.3. Machine Learning/Artificial intelligence anomaly detection

MMT-ML allows different actors, who may or may not be familiar with different Machine Learning (ML) methods, to quickly and efficiently build ML models that would be adaptable to the particular problem. In order to provide it, the architecture of our proposed system is modular, which can be seen in Figure 4.1.3.1. The system allows users to create different ML pipelines using already built, but still configurable components. Figure 4.1.3.1 provides an overview of the system from the point of view of the data flow: data is gathered, prepared, used for ML modelling, and the final results are obtained. As can be noticed in the figure, the modularity of the system has a big advantage of the whole structure being flexible - the modules are built in a way so that they are exchangeable plug-ins, therefore the system can be easily extended and developed in the future, also beyond the project duration. To obtain a result on a particular dataset, the user will need to create a pipeline that deals with the preparation of the data, selecting an algorithm and using the part of the data (so-called train data) to create a model, applying the rest of the data (test data, production data) to obtain predictions, and then perform interpretation and visualisation of the results. It is envisioned

that the system will propose a user-friendly GUI that will allow users to design and manipulate the pipeline in an easy way. Moreover, it would allow a user that is not familiar with programming to utilise the tool, select different ML methods and their parameters, prototype models and execute various experiments.

As mentioned, the system consists of three main modules: preparation module, ML module, and application module. Each of them is responsible for separate functionalities, indispensable for obtaining the final result. The preparation module is a component responsible for data preparation, i.e. taking raw gathered data (outside of the system, in grey colour), and transforming them into a form that is meaningful and adaptable for an ML input. ML module is a part that can work in two modes: training and predicting. Training mode is building the ML model based on the inputted data, selected algorithm and input parameters, while the prediction mode applies the already trained model in order to provide a final result. The application module is in charge of the explanation of the results - by generation of visualisations (graphs, charts), statistics, etc. It can also be connected further to tools that are providing further analysis or explanations of the provided results, e.g. tool providing root cause analysis.

Each of the components consists of a set of actions characteristic to its usage. In the following subsections, we present details of each of them, according to the schema shown in Figure 4.1.3.1.

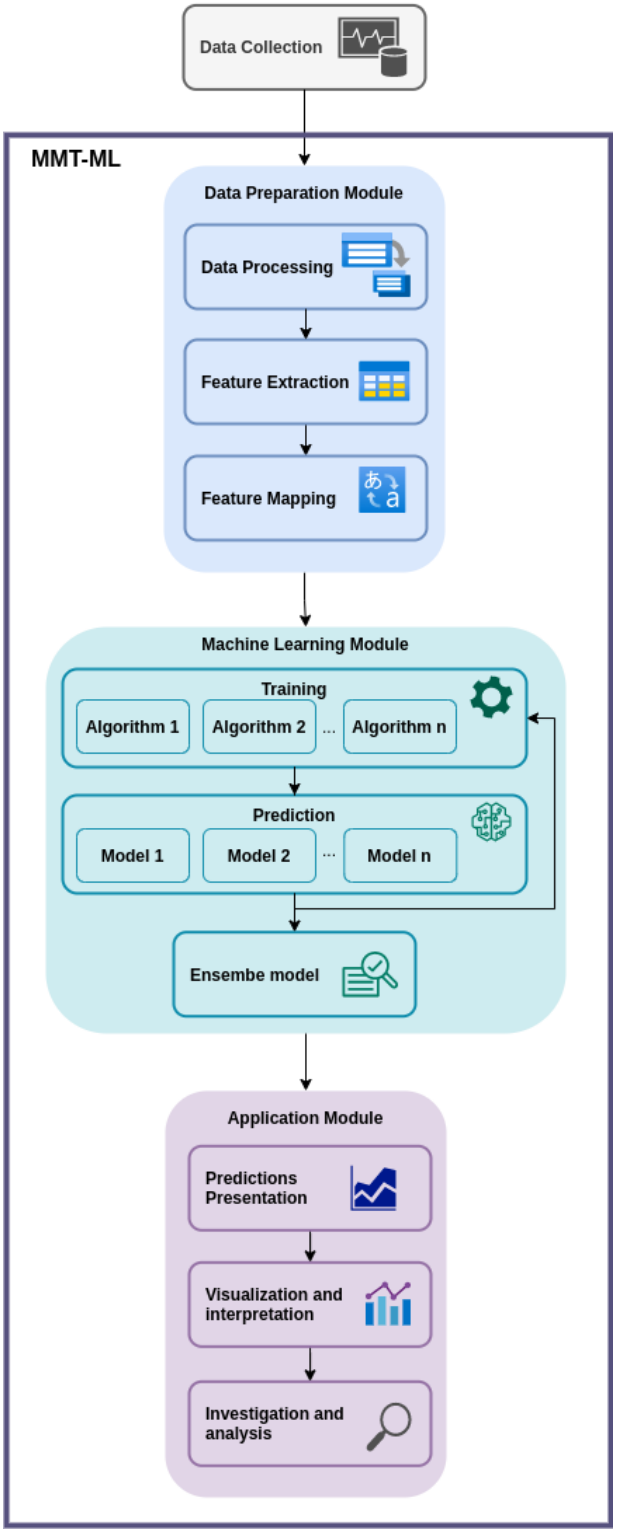


Figure 4.1.3.1 MMT-ML Framework

#### 4.1.3.1 Data Preparation Module

It is the first module in the pipeline. As it receives directly raw data, the main goal of this module is to verify and convert them. This process consists of the following steps:

- Data Processing - It is a module that includes all the tasks connected to data verification, validation, cleaning, and transforming in order to make it the most representative and correct: identifying and correcting possible mistakes, verifying ranges of variables, getting rid of incorrect or empty samples, deciding on the strategy towards out-of-scale variables (e.g. infinity), etc.
- Feature Extraction - It involves the selection of the features from the raw data, and/or calculating additional ones, in order to obtain a concise and meaningful representation of the initial data. In this step, correlation tests of the features might be useful so that the engineered features are the most compact possible, and do not contain redundant information. Moreover, depending on the ML algorithm used in the next module, this step may also require expert knowledge in feature selection in order to extract the most important features. It is assumed that for Feature extraction, the system will be using the MMT-Extract described in Section 4.1.1. That being said, a pipeline containing another module for Feature Extraction can also be used within MMT-ML.
- Feature Mapping - It consists of the translation of all extracted features into a numeric form that can be easily used by any ML model. It involves data scaling (values, distribution), encoding the categorical values to numeric form, etc.

#### 4.1.3.2 Machine Learning Module

This component is a core component of the proposed system. It is responsible for building a model (which is dependable on the data and the chosen ML algorithm), as well as utilising existing ones. We can therefore distinguish its two modes of operating (that are matching the actions *Training* and *Prediction* presented in Figure 4.1.3.1:

- Training - It is designed to create and parametrize the model based on already cleaned and transformed data. This signifies that executes selected by user algorithm, in which a model is built by using the training data in order to find weights and biases of the model that would lead to the best results - as a metrics of result during training a loss function is used, with penalised bad predictions. Depending on a selected algorithm, this step includes also experimenting with different values of algorithm parameters, such as learning rate, activation functions, batch size, etc. In the proposed system, it is assumed that this step is either done by a user a bit

familiar with hypertuning, or it is done by utilisation of the values directly suggested by the system.

- Model evaluation - in order to evaluate if the parameters and training of the model were sufficient, the training needs to be done using a separate dataset from the testing dataset, so that the model is actually tested on the completely new samples - in which case the evaluation checks whether the model is generalised enough. To investigate the results, the correctness of classification is verified using the following terms:
  - True Positive (TP), True Negative (TN) - samples that are correctly assigned to the (normal, anomaly) classes
  - False Positive (FP), False Negative (FN) - samples that are incorrectly assigned to positive or negative classes

- Prediction - It is done after the model is trained (training mode), and the satisfactory results are obtained. It involves utilising the model directly on new, unseen data (in a real-life case scenario these are just production data) and obtaining the results, such as probabilities, classifications, etc. Importantly, the accuracy of the results of the prediction can also be used in order to further hypertune the model.

Obviously, as could be deduced by their descriptions, *Training* needs to be done before *Prediction*. Besides the two actions, which are responsible for creating an adequate model and then producing the predictions, the ML module also consists of an optional element called the Ensemble model. As the system aims to make it easier to prototype and utilise Machine Learning for practical applications, it is assumed that the user may want to create multiple different models - as it is shown in Figure 4.1.3.1. Therefore, instead of selecting one model's predictions from one particular model, it can be beneficial to combine the results of different models together. Therefore, this final (and optional) step of the ML module consists of the ensemble part that is capable of joining the results together. It is envisioned that the ensemble part includes simple methods, such as averaging, as well as more complex ones, such as tournament competition. The particular method can be changed and should be fitted according to the initial ML methods used (so that the type of results can actually be joined together) and to the complexity of the overall problem to solve.

#### 4.1.3.3 Application Module

The module is responsible for presenting the results (obtained from the ML module) in an easy-to-understand and meaningful way. Hence, there are three parts envisioned:

- Predictions - they include the visual and/or numerical presentation of the predictions. The exact way of presentation needs to be adapted to the problem and selected algorithms in the step before. For example, for a classification problem involving detection of correctly and incorrectly positioned markers, the algorithm in the ML module could return a list of incorrect markers - therefore in the Predictions step, one could expect a visual chart of their positions with colours signifying their correctness; on the contrary, in the attack detection in network traffic, ML module could return a list of IPs of possible attackers, accompanied by a certainty value of the prediction.
- Visualisation and interpretation - this part includes all additional information needed for comprehending the results and understanding where they came from. It includes the statistics of the data, graphs of current (and possibly previous) results, etc.
- Further investigation - it is envisioned that depending on the problem, a further investigation and analysis will be necessary in order to fully resolve the initial problem. Therefore, this module assumes that other methods or systems can be integrated or attached. For instance, in the case of attack detection in network traffic, further analysis of the detected anomalies can be required, in which case a module using Root Cause Analysis could be integrated in this step.

### 4.1.4. Root cause Analysis

MMT-RCA relies on machine learning algorithms to identify the most probable cause(s) of detected anomalies based on the knowledge of similar observed ones. It enables systematizing the experience in dealing with incidents to build a historical database and verify whether a newly detected incident is similar enough to an observed one with known causes. Thanks to MMT-RCA's suggestions, remediation actions could be timely and wisely taken to prevent or mitigate the damage of the recurrence of problems. Figure 4.1.4.1 shows the high-level architecture of the tool.

The **data collector** allows gathering information from different sources (e.g., network, application, system, hardware) by relying on dedicated monitoring agents. It has a plug-in architecture that enhances its extension to new data formats. Parsing such data allows extracting various attribute values relevant to the origin of any detected incident. We automatically select the most relevant attributes by using several feature selection algorithms. These attributes increase the analysis accuracy and reduce the data dimensions as well as the computation resources needed.

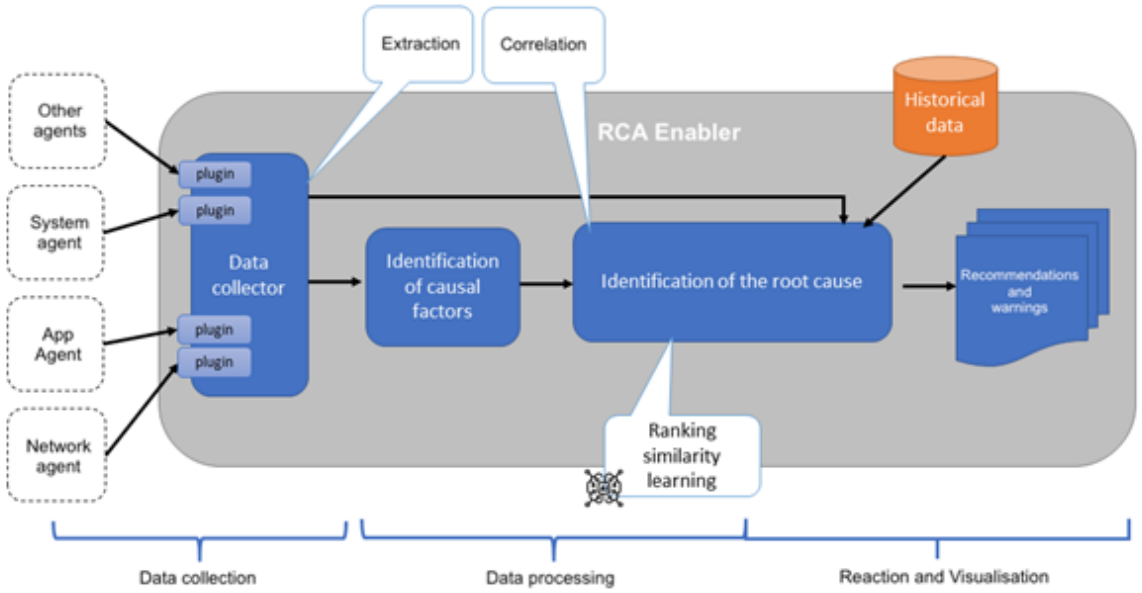


Figure 4.1.4.1 MMT-RCA high-level architecture

The **historical data** is a set of data used for learning purposes. It consists of labelled records collected over time. These records describe the original cause of several incidents (e.g., a sensor is no longer permitted to send data to the central gateway) and the relative attribute values (e.g., downstream data bit-rate measured in the central gateway decreased). The **historical data** is constructed by two means:

- Active learning: By actively performing different tests including the injection of known failures and attacks. In this case, the collected data can be easily labelled since we deal with a controlled system.
- Passive learning: Once an incident is detected without knowing its origin, thanks to the aid of the system experts, classical RCA is performed by debugging different logs and correlating various events to determine the corresponding root causes. The result of this task can be stored in the database with the values of its relevant attributes.

The historical data are derived from these two sources. The idea is to determine when the system reaches a known undesirable state with a known cause. It involves using the concept of Similarity Learning [24], i.e., Ranking Similarity Learning. MMT-RCA calculates the similarity of the new state with the known ones. It presents the most similar states in the

relative similarity order. The final goal is to recognize the incident's root origin by using historical data. In this way, the tool can recommend to the operator which countermeasures to perform based on known mitigation strategies.

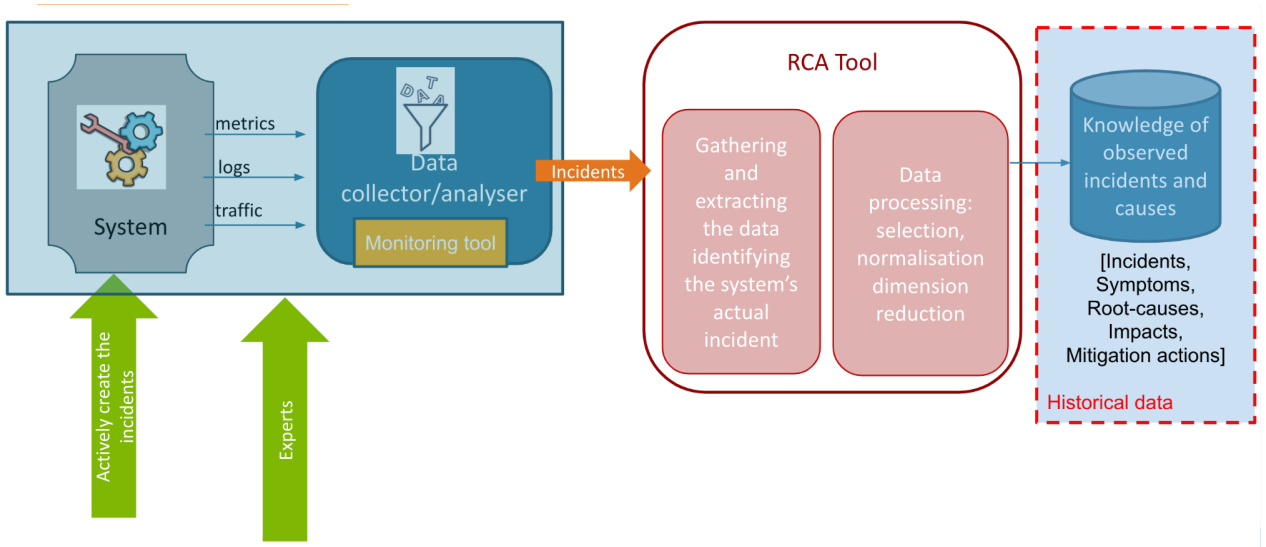


Figure 4.1.4.2 MMT-RCA Knowledge acquisition phase

The MMT-RCA works following two phases: the knowledge acquisition phase (Figure 4.1.4.2) and the monitoring phase (Figure 4.1.4.3). The former is for building a historical database of known problems and incidents. The latter consists of monitoring the system in real-time, analysing the newly-coming incident by querying the historical data, and suggesting possible root causes. It is worth noting that passive learning in the knowledge acquisition phase can be continuously run during the monitoring phase.

Analysing a system requires different statistics and data, i.e., the logs, metrics, network traffic, and any data that could identify the system state. A **data collector** is necessary and can be provided by the system, or an enabler can be deployed to collect different types of data, namely:

- Capturing network traffic: For example, MMT-Probe (TCP/IP networks) and MMT-IoT (IoT- 6LoWPAN) are able to sniff and record the network traffic.
- Reading and extracting logs: The current version of the MMT-RCA supports by default reading the data input in the form of JSON (e.g., MQTT) and CSV files. Other formats can be rapidly taken into account thanks to the extensibility of MMT-Probe (e.g., creating new plug-ins).

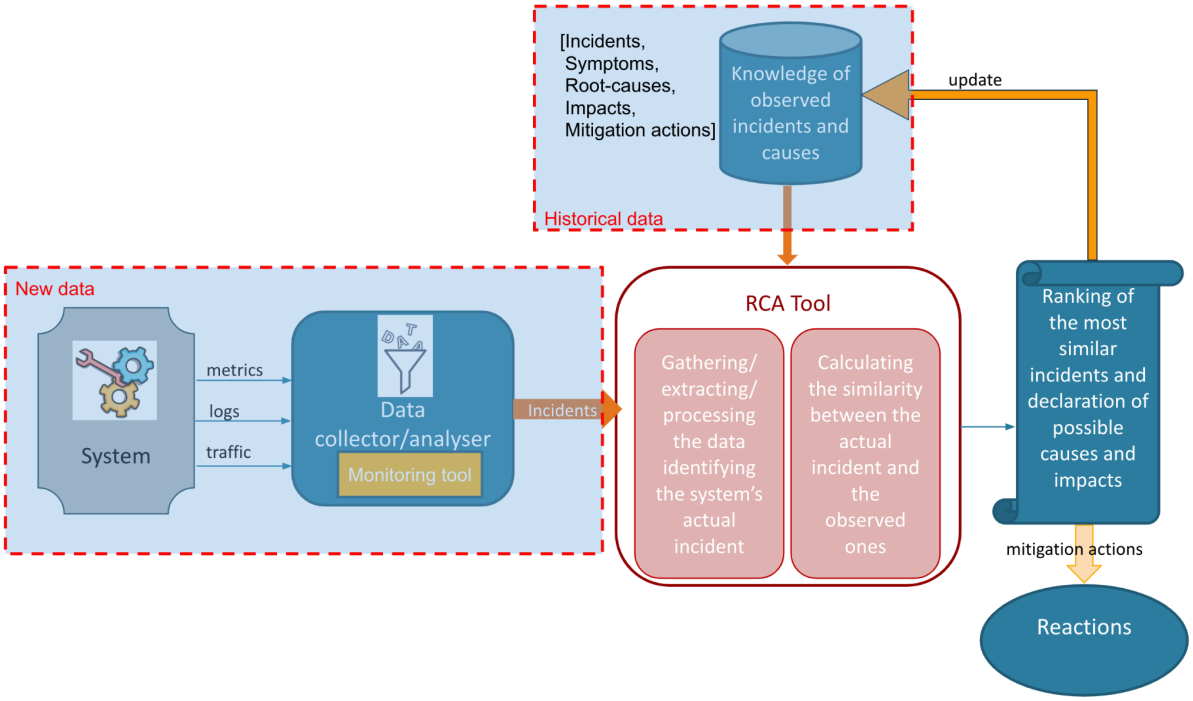


Figure 4.1.4.3 MMT-RCA Monitoring phase

In the knowledge acquisition phase, the data can be collected in two ways:

- Actively injecting or reproducing known failures and attacks, then collecting labelled corresponding data.
- Passively monitoring the system, debugging different logs and traces, correlating various events and particularly consulting the system experts to determine the corresponding root causes as well as the relevant data.

In the monitoring phase, the data is collected and transmitted to MMT-RCA in real-time. MMT-RCA can work in two manners:

- MMT-RCA calculates continuously the similarity of the system's current state with all observed ones in the historical database with a predefined set of analysis features. An alert will be raised when:


- Signature-based approach: MMT-RCA sees that the current state is very similar (i.e., the similarity score is higher than a defined threshold) to a learned incident.
- Anomaly-based approach: MMT-RCA sees that the current state is very different (i.e., the similarity score is lower than a defined threshold) from all the records saved when the system was considered normally functioning.

- MMT-RCA receives an alert about an incident from another Anomaly Detection module (e.g., MMT-Security, MMT-ML). It will calculate the similarity score of the newly detected incident with all observed ones in the historical database with a dynamic subset of all possibly collected features. MMT-RCA will then sort out a ranking of the most similar learned incidents with the known causes.

In theory, there is no restriction in the type of data to be gathered. On the contrary, a maximum of data for identifying the system functionalities is desirable. Even though some data could be redundant, data processing steps are performed to extract the most pertinent data. Therefore, the following data processing steps will be performed:

- **Attribute selection** (also known as feature selection ): It is one of the core concepts in Machine Learning that tremendously impacts the model performance. For complex systems, it is common that the data collected is too complicated or redundant. In other words, there might be some irrelevant or less important attributes (i.e., noises) contributing less to the target variable. Removing the noises helps not only to improve the accuracy but also to reduce the training time. It is the first and most essential step that should be performed automatically based on the feature selection techniques or manually by system experts.
- **Data normalization: T**he data used as the input of MMT-RCA is normally heterogeneous. A step of **data normalization** is needed for eliminating measurement units and making the attributes comparable despite different value ranges.
- **Similarity calculation:** Our RCA approach relies on similarity learning to identify the most probable cause(s) of detected anomalies based on the knowledge of similar observed ones. The accuracy of the results depends on the algorithm used for calculating the **similarity score**. Thus, computing the similarity score based on more than one similarity and distance measure will help improve the confidence and precision of the results. In the training phase, we determine the measures. Therefore, when we compare a known state and its repetition, we have a similarity score as high as possible. Besides, to avoid false positives, the similarity between a common proper state and each known malicious state should be as low as possible.

### 4.1.5. Dashboard

MMT-Operator is a visualisation Web application. It allows collecting and aggregating metadata and reports provided by MMT-Probe, and presents them via a graphical user interface (e.g., alarms, line charts). MMT-Operator is customisable: the user is able to define additional charts or customise the large list of predefined ones. Different chart types and graphs can be used including pie, histograms, time charts, stacked area charts, sequence charts, tables, hierarchical tables, etc. Figures 4.1.5.1 and 4.1.5.2 show examples of the charts that MMT-Operator can generate. Figure 4.1.5.1 depicts data volumes over time, in the top chart per traffic flow, i.e. incoming and outgoing, and in the lower graph by network protocol. Figure 4.1.5.2 shows pie charts that provide information about the top users in the traffic (left), including their IP and MAC addresses, and the volume of data they are producing; as well as, the top traffic profiles (right).


Figure 4.1.5.1 Examples of dashboards: Data volume per traffic flow (top) and network protocols (lower)

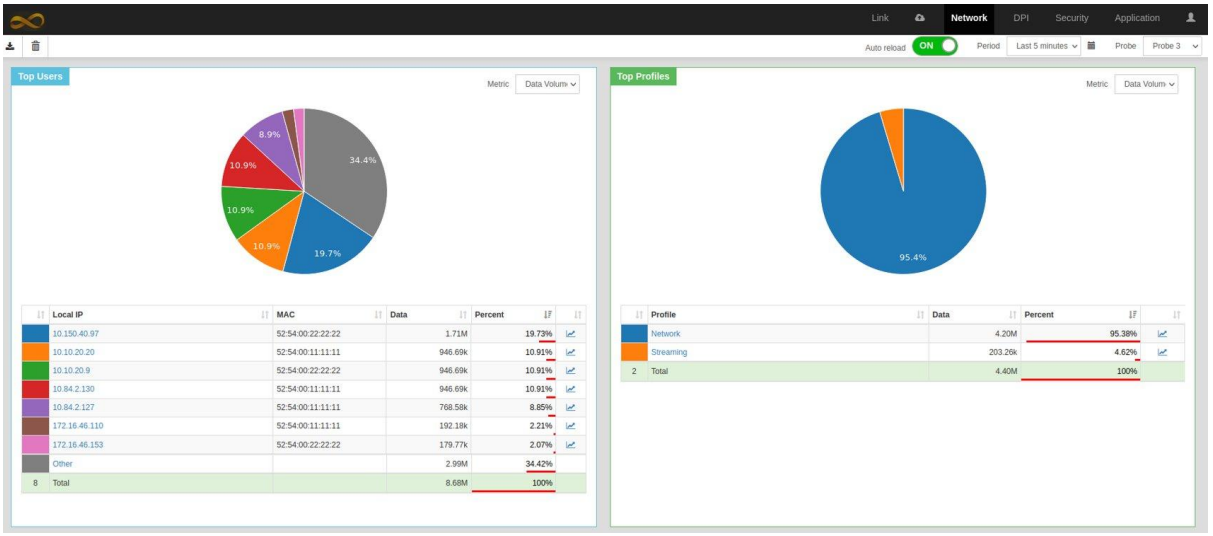


Figure 4.1.5.2 Examples of dashboards: Top users (left) and traffic profiles (right)

## 4.2 Workflow

The monitoring solution conceived in VeriDevOps has a sequential workflow that starts with **the capture of different logs and traces** to be analysed at runtime. We can either have a unique source of traces or many of them. In this last case, the correlation between events from different sources should take potential delays and miss-synchronisation between capture times of these events into account. A second step is **the extraction of relevant attributes** from these logs. These attributes can be included in the raw data of traces or can be computed on one or several incoming information. These attributes (called also metadata or features) will be used by MMT-Security to correlate them in order to **check security properties** or by MMT-ML/AI to **detect drifts and anomalies** from the learned baseline. More attributes can also be used to **determine the origin of a security incident** using the MMT-RCA tool. All the **analysis reports** should be sent to a security operator in order to constitute a **global view** of the security status of a running critical industrial system.

## 4.3 APIs

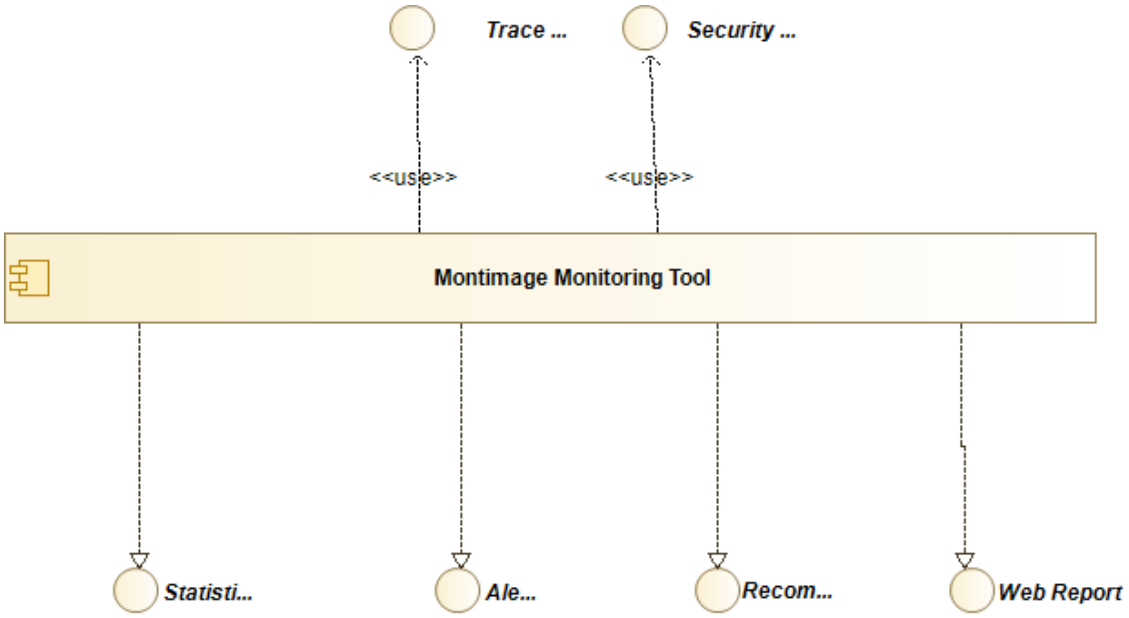


Figure 4.3.1. Services and Interfaces (D1.4 VeriDevOps Framework Architecture and Roadmap)

As presented in Figure 4.3.1, obtained from deliverable D1.4 VeriDevOps Framework Architecture and Roadmap, the MMT tool has two main inputs, the collected traces that can be provided as a pre-recorded file or in streaming mode and the security properties to be analysed (a security property can denote a security requirement or a security problem). The analysis of the traces will provide several statistics and reports to the critical industrial system operator to understand its usage status. It will also provide a set of alerts according to some security issues (security requirements violations, attack and intrusion detection, anomaly detection) and recommendations to mitigate their impact on the system in the format of Web reports.

# 5. Implementation status and planning

## 5.1 Feature Extraction library

MMT-Extract was developed by Montimage and has been widely tested and commercially used since 2012, and although it is able to parse numerous protocols, including the common ones used on the Internet, sometimes it is necessary to include new plugins to be able to process network traffic or logs of less common applications.

In the context of the VeriDevOps project, ABB has provided Montimage with data from their Load Positioning Systems (LPS) in two formats: network traffic containing their internal protocol, and csv files. Furthermore, following the workflow depicted in Figure 5.1.1, after studying the data format of the ABB case study, two plugins for MMT-Extract have been incorporated into the architecture in order to be able to identify, classify and extract events of ABB LPS use case described in Section 6.1., in both formats: network traffic and csv files.

### 5.1.1 Planning for the upcoming months

In the upcoming months of the project, the Fagor case study will be analyzed in order to determine if new plugins to add new protocols are required, and if it is the case these protocols will be developed.

## 5.2 Rule-based monitoring

MMT-Security has also been widely tested since its creation by Montimage in 2010. In the context of the VeriDevOps project, ABB has provided Montimage with data from their Load Position Systems (LPS) use case described in Section 6.1, and Montimage has included the rule shown in Figure 4.1.2.1, as well as, the experiments described in section 6.1.1 to solved the proposed problematic.

### 5.2.1 Planning for the upcoming months

In the upcoming months of the project, ABB and Fagor case studies will be analysed in order to determine relevant incidents that could be detected by creating new MMT Security rules.

Moreover, we planned to improve the performance of the rule-based monitoring performed on ABB LPS case study.

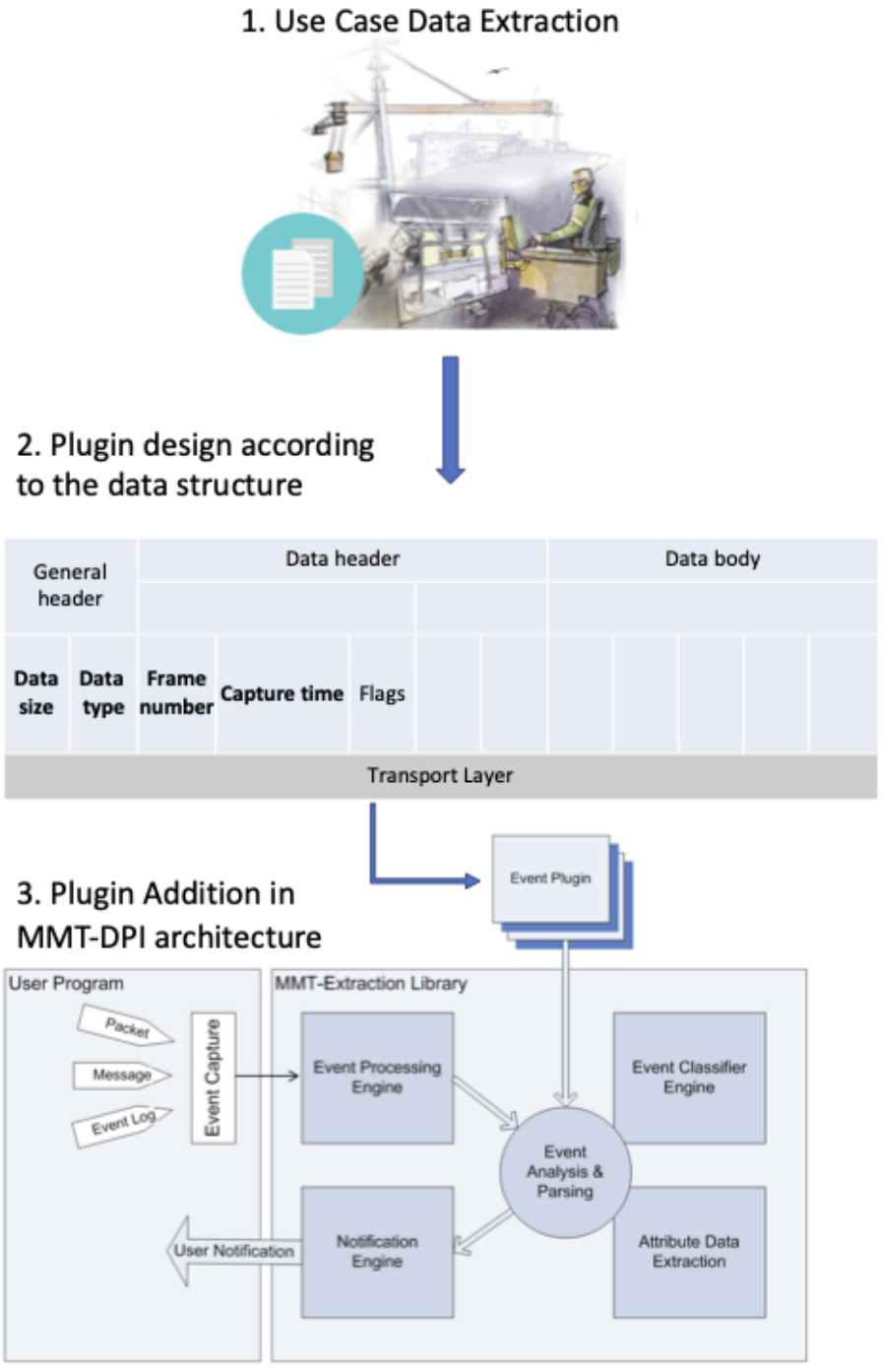


Figure 5.1.1 Plugin generation workflow

## 5.3 Machine Learning/Artificial intelligence anomaly detection

The current state of implementation of MMT-ML consists of a whole pipeline including a data preparation module, machine learning module, and basic application module. It can be noted that each of the particular usages of the MMT-ML system will consist of different needs for data transformation and ML model. Therefore, we intend to extend each module with time, which is why the modular structure of the system is a perfect fit. To date, the MMT-Extract is used as a part of the preparation module (i.e. Data processing and feature extraction). Feature mapping in Data Preparation module, as well as Machine Learning module, and Application module utilise Python3. The Machine Learning module is also using various libraries such as Keras, Tensorflow and sklearn for major machine learning functionalities. In all parts, we also utilise data analysis and statistical libraries: pandas, numpy, matplotlib, seaborn.

### 5.3.1 Data Preparation module

As mentioned, a pipeline of MMT-ML can be created by using different parts for each module. Hence, in the current stage of the implementation, in the Data Preparation module, we tested the modularity of the MMT-ML and therefore we have utilised two similar components for Feature Extraction that are using raw network traffic directly (i.e. pcap file): Montimage's MMT-Extract (described in Section 5.1), and external tool CICFlowMeter provided by Canadian Institute of Cybersecurity at the University of Brunswick[9]. In order to further transform them and calculate some of the features, a python script using the feature values extracted from MMT-Extract is used. Currently, MMT-Extract is providing features listed in Table 6.2.1.1.

| **Feature** |
|---|
| Flow duration |
| Sequence of packet length |
| Total packets in the forward direction |
| Total packets in the backward direction |
| Total size of packet in forward direction |

[9] https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter

| |
|---|
| Total size of packet in backward direction |
| Sequence of packet time (evry 50ms from 0 to 10000ms) |
| Sequence of packet size in forward direction |
| Sequence of packet length in backward direction |
| Number of packets with FIN in forward direction |
| Number of packets with SYN in forward direction |
| Number of packets with RST in forward direction |
| Number of packets with PUSH in forward direction |
| Number of packets with ACK in forward direction |
| Number of packets with URG in forward direction |
| Number of packets with FIN in backward direction |
| Number of packets with SYN in backward direction |
| Number of packets with RST in backward direction |
| Number of packets with PUSH in backward direction |
| Number of packets with ACK in backward direction |
| Number of packets with URG in backward direction |
| Number of packets where source port is greater to 1024 |
| Number of packets where source port is less or equal to 1024 |
| Number of packets where destination port is greater to 1024 |
| Number of packets where destination port is less or equal to 1024 |
| Maximum size of packet in forward direction |
| Minimum size of packet in forward direction |
| Mean size of packet in forward direction |
| Standard deviation size of packet in forward direction |
| Entropy of size of packet in forward direction |
| Maximum size of packet in backward direction |
| Minimum size of packet in backward direction |

| |
|---|
| Mean size of packet in backward direction |
| Standard deviation size of packet in backward direction |
| Entropy of size of packet in backward direction |
| Maximum packets number in forward direction |
| Minimum packets number in forward direction |
| Mean of packets number in forward direction |
| Standard deviation of packets number in forward direction |
| Entropy of packets number in forward direction |
| Maximum packets number in backward direction |
| Minimum packets number in backward direction |
| Mean of packets number in backward direction |
| Standard deviation of packets number in backward direction |
| Entropy of packets number in backward direction |

Table 6.2.1.1. Extracted features

The choice of the features that we are interested in is dictated by an attempt to best characterise and distinguish normal and malicious traffic. In order to do that, the features are focused on information about the time of packets (e.g. average time between packets, etc.), the length of the packets (in bytes), as well as flag usage, and others.

The final step of the Data Preparation module consists of data transformation, in which the categorical features are transformed to numeric ones, and the data is normalised, which is done by the use of *sklearn* library in Python.

### 5.3.2 Machine Learning module

At present, there are several models available in the ML module for training and testing. The implementation is done in Python with the use of *Keras* and *Tensorflow* libraries for the Machine Learning/Deep Learning algorithms, along with *pandas* and *numpy* libraries for various calculations. In particular, the currently available ML template models are:

- Neural Networks - it is a model created from a collection of connected units (neurons) grouped in layers. They are used for learning a function that aims to transpose an input (samples) into desired output (e.g. sample labels). The process of

learning utilises the changing of weights of each neuron and its connection between each other in order to find the best weight values. The learning process is performed multiple times during which the weights of neurons are corrected by using backpropagation that is using the error rates between the actual and desired values of output.

- Stacked Autoencoder with Logistic Regression - it is a structure made from two different models. Logistic Regression is a linear ML model that tries to find a hyper-plane that best divides input samples (points in multidimensional space) into different classes (output). Here, as an input of Logistic Regression, we pass the result of a Stacked Autoencoder. Autoencoder is a neural network structure used to learn a compressed representation of raw data. It has an encoding and decoding layer(s), which are responsible for reducing the dimensionality of input data (encoder) and then reconstructing the input from such a compressed form (decoder). It is exactly this encoded, already dimensionality-reduced form of raw data input that is passed into a Logistic Regression.

### 5.3.2 Application module

In the current stage of the implementation where the focus is put on the building and correctly tuning the parameters of different models, the application module consists of the simple visualisation of confusion matrix (that provides information of samples being either correctly or wrongly assigned to the classes - in the network traffic usage these are normal and malicious classes), as well as values of different evaluation metrics such as accuracy, precision or recall. This is due to the fact that thanks to such visualisation of results, we are able to evaluate how well each of the models performs.

### 5.3.1 Planning for the upcoming months

One of our main challenges is to generalise our ML implemented models (see Section 6) to perform a real-world check of unseen datasets, and confirm there is no overfitting in the solutions of ABB and Fagor case studies. For the moment, only traces of ABB were directly analysed; therefore, in the incoming month, Fagor traces will be considered.

In future stages of implementation the following charts for anomaly detection in network traffic are envisioned:

- Percentage of normal/malicious in the dataset
- List of features with classification to be exported in .csv form

- List of IPs with packets classified as malicious

## 5.4 Root cause Analysis

### 5.4.1 Current status

The enabler consists of the Python libraries running in the background for receiving and analyzing the monitoring data and the NodeJS / Node-RED GUIs for visualizing the collected statistics as well as the analysis results. The Python libraries address the three main data processing steps as follows:

- Attribute selection: The current version of the MMT-RCA has been integrated with the following feature selection techniques:
    - Univariate feature selection: The selection of the best features is based on univariate statistical tests. Each feature is compared to the target variable while the other features are temporarily ignored. The goal is to determine whether there is any statistically significant relationship between them. Each feature has its test score. The bigger the score is, the more likely the feature is important. The features with top scores should be selected. Currently, the test score is the average of the scores calculated based on the chi-square test, the f test, and the mutual information classification test [feature_selection_book]
    - Recursive feature elimination (RFE): Select features by recursively considering smaller and smaller sets of features. The idea is to use an external estimator (logistic regression model and random forest model [feature_selection_model]) that assigns weights to features (e.g., the coefficients of a linear model). The least important features are step by step pruned from the current set of features. This procedure is recursively repeated on the pruned set until the desired number of features left is eventually reached. Compared to univariate feature selection, RFE considers all features at once, thus can capture interactions.

- Data normalization: It consists of two steps:

    - Standardizing data to have a mean of zero and a standard deviation (s) of 1 (Figure 5.4.1.1):

$$x_{standardized} = \frac{x - mean(x)}{s}$$

- Rescaling the data to have values between 0 and 1:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

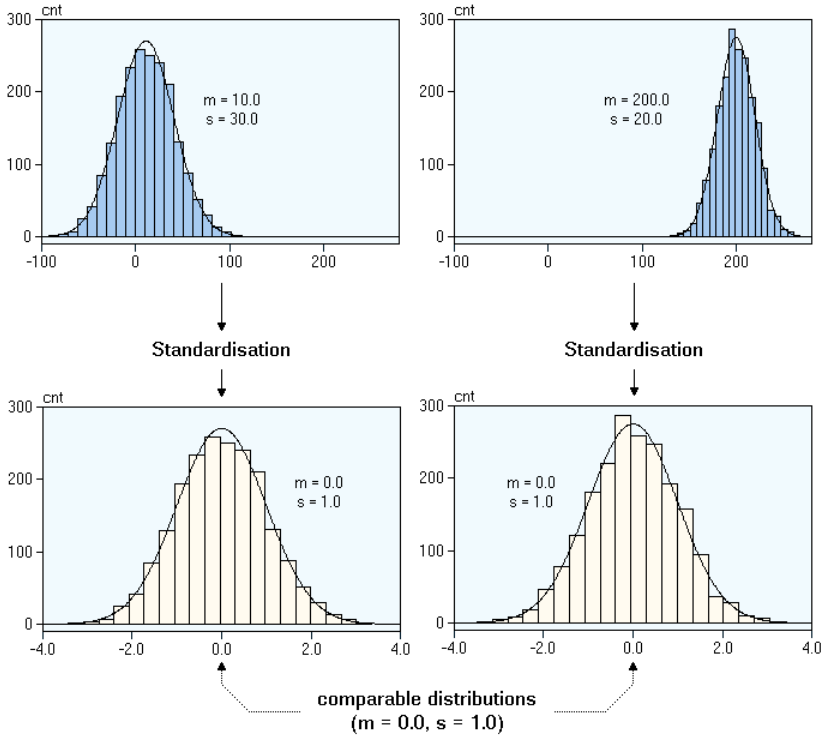


Figure 5.4.1.1 Data standardization

- Similarity calculation: The current version of MMT-RCA has been integrated with the following similarity/ distance measures:
  - Cosine similarity [24]
  - Adjusted cosine similarity [24]
  - Jaccard similarity [24]
  - Euclidean distance [24]
  - Manhattan distance [24]
  - Minkowski distance [24]

In addition, MMT-RCA provides a set of generic APIs to communicate with other modules. In table 5.4.1.1, we provide details about the API exposed by MMT-RCA that enables its integration with other tools in VerDevOps. A set of high-level commands are exposed by MMT-RCA in the form of a REST API.

| Method | Resource | Content-type | Description |
|---|---|---|---|
| GET | /rca/getall | Response: application/json | To retrieve all the records (known incidents) stored in the historical database |
| GET | /rca/getone | Response: application/json | To retrieve a record (a known incident) stored in the historical database identified by its ID |
| POST | /rca/insertone | Response: application/json Parameter: application/json | To report a newly detected incident with all the related traces so that it could be inserted to the historical database |
| GET | /rca/logs | Response: text/plain | To retrieve all the logs of the tool |
| GET | /rca/status | Response: application/json | To retrieve the current status of the monitored system and how it looks similar (in percentage) to the known incidents |
| GET | /rca/help | Response: application/json | To retrieve the API documentation |

| | | | |
|---|---|---|---|
| POST | /rca/update | Response: application/json Parameter: application/json | To push an update regarding a record (a known incident) in the historical database |
| POST | /rca/deleteone | Response: application/json Parameter: application/json | To delete a record (a known incident) in the historical database |
| POST | /rca/deletemany | Response: application/json Parameter: application/json | To delete multiple records (known incidents) in the historical database |

Table 5.4.1.1: MMT-RCA's general APIs

### 5.4.2 Planning for the upcoming months

In the upcoming months, a collaboration with the use case provider will be needed in order to define and to implement:

- Datasets collection:
  - The communication means that MMT-RCA collects data from the use case provider (e.g, MQTT broker, Kafka bus, offline logs).
  - The monitoring features that could be collected, extracted and inferred.
  - The possibility to actively inject faults/ attacks to the system to train MMT-RCA

In addition, we plan to improve the performance (e.g., accuracy, response time) of the analysis based on advanced Machine Learning techniques to achieve the KPIs defined previously in Section 3 as well as to enhance the GUI for an interactive and more friendly version.

## 5.5 Dashboard

MMT-Operator has also been widely tested since its creation by Montimage in 2010. In the context of the VeriDevOps project, the library has already been used to visualise events of

ABB traffic, nevertheless, it requires to be adapted to visualise the LPS variables information, and to create relevant dashboards adapted to ABB and Fagor case studies.

### 5.5.1 Planning for the upcoming months

In the incoming months a set of graphics adapted to case studies will be designed and implemented by using the MMT-Operator library as a starting point.

# 6. Case study applications

## 6.1 ABB Load Position System case study

Load Position System (LPS) provides information that controls a crane. To perform this task, the system's sensors monitor spatial variables; therefore, the measurements are validated by a Programmable logic controller (PLC), whose algorithm has sometimes been shown to be incorrect, compromising the safety of the system. VeriDevOps monitoring solution methods and devices aim to increase safety and security by corroborating PLC results using different methods. For more details about the ABB case study please refer to D1.1 Case Studies: Description, requirements analysis, and implementation.

### 6.1.1 Rule-based detection approach

We propose the following methodology in order to provide a solution to LPS variables identification problem:

1. **Data preprocessing:** relevant attributes were extracted from the original data, and correlated to compute new features such as the Euclidean distances and angles between two consecutive measures were calculated
2. **Define strategies to identify invalid measurements:** data was studied in order to define strategies to identify invalid measurements, using the extracted and calculated features.
    a. **Number of variables detected by the PLC:** If the number of detected spatial variables is higher than a determined threshold, the measure is tagged as invalid.
    b. **Speed variation:** If the variation speed of a measure (displacement/ time interval) is less than the threshold, then the measure is tagged as invalid.
    c. **Uneven distance:** If the magnitude of the displacement of spatial variables was very uneven, then the measure is tagged as invalid.

d. **Displacement magnitude:** If the magnitude of the displacement between the variables was higher than the threshold, then the measure is tagged as invalid.

3. **Tuning of algorithms:** all strategies defined above require determining a threshold value. For that purpose, we used the Dual Annealing optimization algorithm, as all the objective functions were non-differentiable.
4. **Final decision:** Finally, a voting system will be established, on which we combine the verdicts of the different strategies to provide a final decision. Therefore, measurement is valid only if three of the four enumerated strategies give a valid verdict.

The number of detected variables by the PLC demonstrated to be the most efficient strategy to identify the invalid measures; therefore, we implemented and tested the security rule shown in Figure 4.1.2.1. Figure 6.1.1.1 shows invalid measure occurs when the number of detected variables is high. Other proposed strategies were not effective and must be substituted, as the number of detected variables is not enough to classify all measurements: Figure 6.1.1.1 shows that numerous valid measures also occur with an elevated number of detected variables.

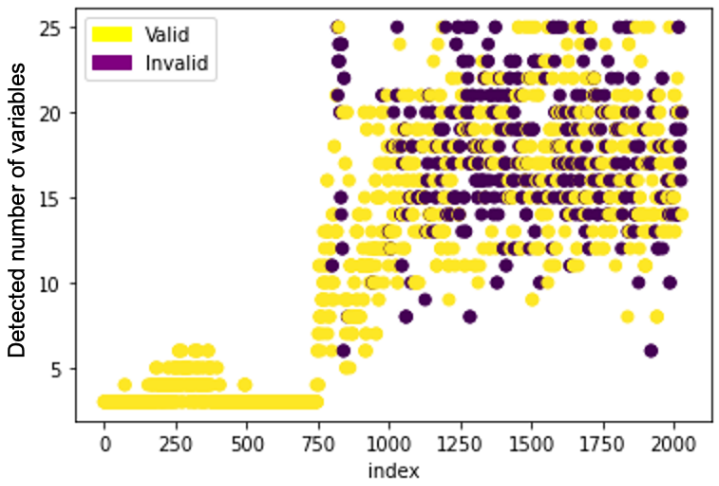


Figure 6.1.1.1 Relation between detected number of variables and their validity

Table 6.1.1.1 shows the partial result we obtained by using strategy number 1, which needs to be combined in step four of our methodology with new strategies to improve the performance of the engine. From the table we conclude that applying this simple strategy

works well enough classifying good measures, but, in particular the precision in the detection of the malicious class, in this case, the invalid measures, shows that the fraction of truly invalid measures instances among all the retrieved instances need to be improved.

| Strategy | Metric | Normal Class: valid measures (%) | Malicious Class: invalid measures (%) |
|---|---|---|---|
| Detected number of variables | Accuracy | 0.73 | |
| | Precision | 0,92 | 0,46 |
| | Recall | 0,71 | 0,79 |

Table 6.1.1.1 Partial results of rule-based detection applied to ABB case study

## 6.1.2 ML anomaly detection approach

In order to propose an alternative solution to ABB LPS case study, targeting a better performance, MMT-ML was used.

### 6.1.2.1 Data Preparation module

Machine learning applications require important amounts of data that additionally must contain a fair representation of the classes, in this case: valid and invalid measures. In a first instance, ABB provided Montimage with the following type of datasets:

1. Recordings without incidents, therefore without invalid measurements
2. Recordings with only incidents, therefore with only invalid measurements
3. Recordings with a single incident, therefore with both valid and invalid measurements. Nevertheless, the invalid measurements class was not representative enough in the data, because the proportion of valid measurements is much higher

Relevant attributes were extracted from the original data, and correlated to compute new features the same used in the rule-based schema proposed in Section 6.1.1: Euclidean distances, angles, and speed variation between two consecutive measures.

Moreover, we create a new dataset, with the 10935 number of samples and a balanced representation of both classes, by combining datasets provided by ABB. This

dataset was divided to create the training and verification datasets, using proportions of 0.75, and 0.25 of the data respectively. Finally, as the chosen ML model is sensitive to feature scaling, we scaled each attribute on the input vector to [-1, +1].

#### 6.1.2.2 Machine Learning module

As we counted with sufficient amounts of labelled data, we decided to target the problem by using a supervised learning approach. Table 6.1.2.2.1 shows the available features and those that were already tested, individually and grouped.

| Features ID | Features description | Already tested |
|---|---|---|
| F1 | Detected number of variables | Yes |
| F2 | Angle displacement | Yes |
| F3 | Speed variation | Yes |
| | **Variables that depends on the LPS configuration** | **No** |

Table 6.1.2.2.1 Features used for the supervised-learning algorithm

We used a multi-layer Perceptron classifier, with 100 neurons in the hidden layer, LBFGS solver[25], and alpha variable equal to 0.0001. Table 6.1.2.5.2 summarise our results.

| Features | Accuracy | Recall invalid measurements | Recall valid measurements | Precision invalid measurements | Precision valid measurements |
|---|---|---|---|---|---|
| F3 | 0.954 | 0.990 | 0.917 | 0.925 | 0.989 |
| F1, F2 | 0.965 | 0.930 | 1.0 | 1.0 | 0.935 |
| F1, F3 | 0.995 | 0.991 | 1.0 | 1.0 | 0.992 |
| F1, F2, F3 | 0.950 | 0.990 | 0.909 | 0.918 | 0.989 |

Table 6.1.2.5.2 summarise our results

As we created a new dataset from the combination of recording of two different

instantiations , with different configurations, of the same LPS system architecture, we could not take into consideration features dependent on the LPS configuration. The model provided good results when it was tested with the verification dataset issue of the division of the original dataset, which proves that our model is actually able to predict accurately the validity of the measures. The next challenge is to generalise our model to perform a real-world check of an unseen dataset to confirm its effectiveness to solve the problem stated in the case study.

## 6.2 Fagor encrypted traffic

In the context of the VeriDevOps project Fagor has created a test environment, based on Azure infrastructure, and containing two virtual machines with CentOS7 OS, that runs an Apache nifi server and an internal Fagor application, as shown in Figure 6.2.1. Fagor proposes using this platform to test VeriDevOps facilities. In the case of WP3 Security monitoring, we are interested in performing anomaly detection in the encrypted network traffic exchanged between the cloud and the final user, therefore we should be able to detect anomalies without considering features related to the packet payloads. In the following sections, we explain a first study we performed about this subject using MMT-ML, for the moment without analysing Fagor platform traffic. For more details about Fagor case study please refer to D1.1 Case Studies: Description, requirements analysis, and implementation.

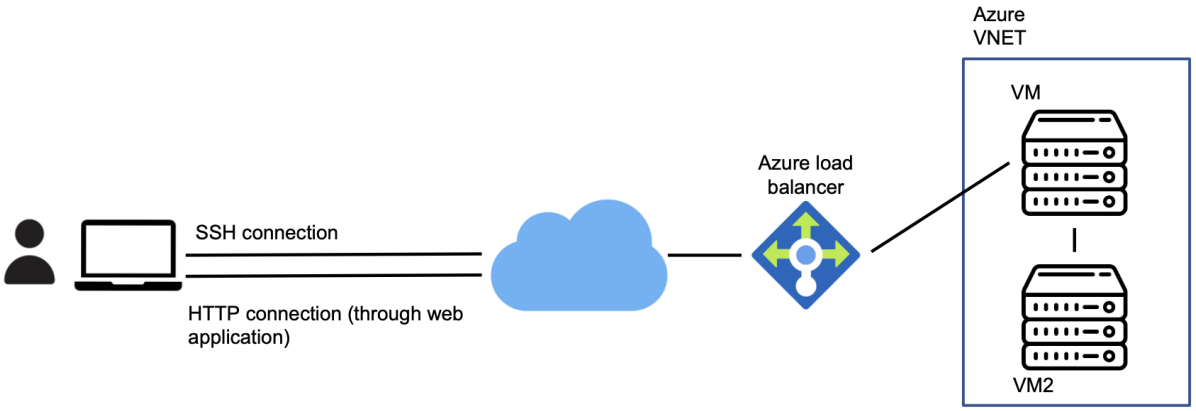


Figure 6.2.1 Fagor test environment

### 6.2.1 Encrypted traffic anomaly detection

MMT-ML is made in order to allow users with different levels of knowledge in both program and machine learning to easily create ML solutions and quickly obtain results. This implies that examples of use may vary greatly, some of the examples could be quality verification of products, positioning of elements in production, or anomaly detection on various production data. To give a concrete example, we will focus on anomaly detection for network traffic with the use of Neural Networks (NN). To put it simply, the goal is to evaluate whether or not a particular traffic flow is a malicious one, by checking if it has out-of-ordinary characteristics. Such a case implies that MMT-ML operates on raw network traffic (e.g. pcap file). In the experiments, we have used two corresponding components mentioned before responsible for feature extraction: MMT-Extract and CICFlowMeter. The feature mapping was done with the use of python. In order to create a NN model the dataset was divided into training and test sets. The dataset on which our tests were performed is an open-source IDS2018 database from Canadian Institute of Cybersecurity, University of Brunswick[10]. The features in the numeric form from the training set were utilised as an input for Deep Neural Network structure and utilised to train the model. The model structure and parameters were tuned in order to get satisfactory results. The current structure of deep NN is shown in Figure 6.2.1.1. The input layer consists of the 76 numeric features. We utilise 2 hidden fully connected layers (seen as “Dense” with the corresponding number of neurons in the figure). In order to prevent overfitting of the model, we utilise batch normalisation and dropout layers. The last layer (“dense_2”) is giving an estimation of classification, where class 0 is normal traffic, and class 1 signifies a malicious one.

[10] https://www.unb.ca/cic/datasets/ids-2018.html

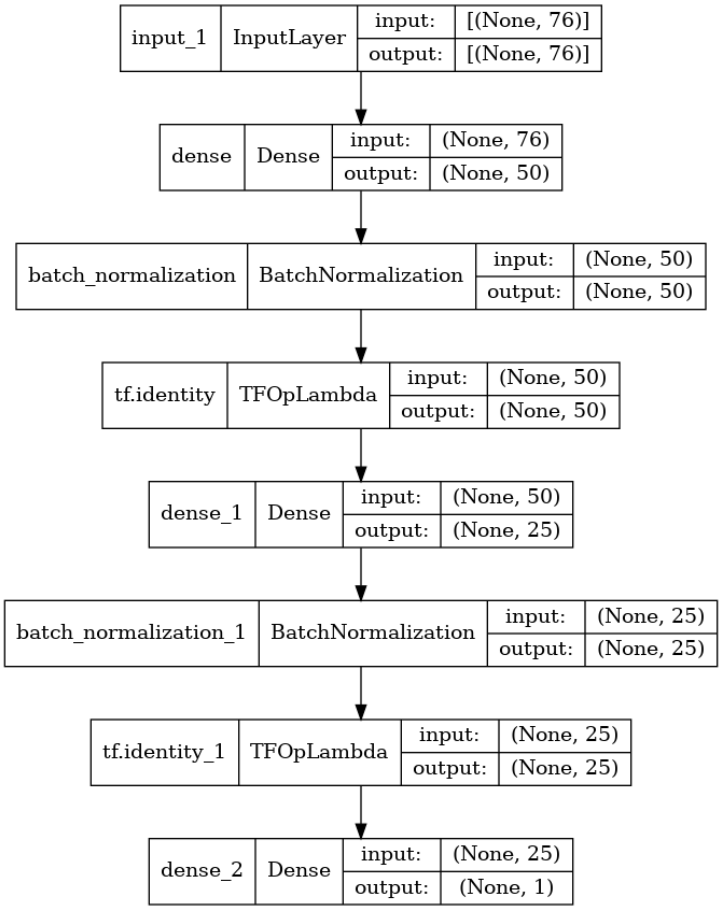


Figure 6.2.1.1. Structure of the used deep NN

Next, the test portion of the dataset was used in order to verify the correctness of the model. The preliminary tests using Deep Neural Networks have already given satisfactory results. Table 6.2.1.1 presents the outcome of the model, tested on 200K samples (unseen samples, 50/50 split between normal and malicious traffic) from the IDS2018 database. The details showing the true and false ratio of classification, where 0 symbolises normal traffic class and 1 symbolises malicious one, can be seen in the confusion matrix shown in Figure 6.2.1.2.

Table 6.2.1.2: Confusion Matrix containing values of TP, TN, FP, FN for test dataset; the numbers on the X-axis signify predicted normal (0) and malicious (1) class, while Y-axis values signify true normal (0) and malicious (1) class.

| Metric | Value for Normal Class (%) | Value for Malicious Class (%) |
|---|---|---|
| Precision | 0.8 | 0.99 |
| Recall | 0.99 | 0.75 |
| F1 | 0.88 | 0.85 |
| **Metric** | Value | |
| Accuracy | 0.88 | |

Table 6.2.1.1 Outcome of the model

| True class | | | |
|---|---|---|---|
| **Predicted class** | | Positive | Negative |
| | Positive | 99817 | 183 |
| | Negative | 24609 | 75391 |

Table 6.2.1.2 Confusion matrix

In the incoming months we will analyse Fagor testbed network traffic in order to adapt our already tested model to predict malicious activity on it.

# 7. Conclusions and perspective

The work described in the present document addressed partially the preparation of the risk assessment process, identification of threats sources, and identification of threats events tasks. Vulnerability identification mechanisms are described in D3.1: Threat oracle engine specification, design and implementation initial version.

The work presented in this deliverable is still in progress, and in the following months, we will continue implementing and testing the different MMT modules exposed in this deliverable; as well as, expanding the already prototyped tools to address the objectives of the monitoring at operation tasks in VeriDevOps. Future steps of the risk assessment process will be covered in the final version of this deliverable, while the rest of the procedures of the risk management process will be covered in D3.5: Attack response - Root cause analysis and countermeasures initial version, and in D3.7: Reactive protection at operations.

# 8. References


[1] National Institute National Institute of Standards & Technology, *Guide for Conducting Risk Assessments: NIST SP 800-30 Rev 1*. 2019.

[2] D. Prince, "Cybersecurity: The Security and Protection Challenges of Our Digital World," *Computer*, vol. 51, no. 4. pp. 16–19, 2018 [Online]. Available: http://dx.doi.org/10.1109/mc.2018.2141025

[3] P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. Springer, 2017.

[4] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[5] C. Thompson and D. Wagner, "A Large-Scale Study of Modern Code Review and Security in Open Source Projects," *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE*. 2017 [Online]. Available: http://dx.doi.org/10.1145/3127005.3127014

[6] X. Franch and A. Susi, "Risk assessment in open source systems," *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*. 2016 [Online]. Available: http://dx.doi.org/10.1145/2889160.2891052

[7] A. Salamai, O. Hussain, and M. Saberi, "Decision Support System for Risk Assessment Using Fuzzy Inference in Supply Chain Big Data," *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/hpbdis.2019.8735465

[8] National Standards Authority of Ireland, *Risk Management: Risk Assessment Techniques (IEC/ISO 31010:2009 (EQV)*. 2013.

[9] N. Uddin Sheikh, H. Rahman, S. Vikram, and H. AlQahtani, "A Lightweight Signature-Based IDS for IoT Environment," *arXiv e-prints*, p. arXiv:1811.04582, Nov. 2018.

[10] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems," *Appl. Soft Comput.*, vol. 92, p. 106301, 2020.

[11] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1. 2019 [Online]. Available: http://dx.doi.org/10.1186/s42400-019-0038-7

[12] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, "Machine Learning for industrial applications: A comprehensive literature review," *Expert Syst. Appl.*, vol. 175, p. 114820, Aug. 2021.

[13] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017.

[14] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, "Cybersecurity for Industrial Control Systems: A Survey," *arXiv [cs.CR]*. 2020 [Online]. Available: http://arxiv.org/abs/2002.04124

[15] G. Nguyen *et al.*, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 77–124, Jun. 2019.

[16] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1. pp. 23–45, 2016 [Online]. Available: http://dx.doi.org/10.1080/21693277.2016.1192517

[17] M. A. Latino, R. J. Latino, and K. C. Latino, *Root cause analysis: improving performance for bottom-line results*. CRC press, 2019.
[18] I. S. O. Vim, "International vocabulary of basic and general terms in metrology (VIM)," *Int. Organ.*, vol. 2004, pp. 09–14, 2004.
[19] D. Powers, "Evaluation: From precision, recall and F-factor to ROC, informedness, markedness & correlation (Tech. Rep.)," *Adelaide, Australia*, 2007.
[20] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[21] B. Wehbi, E. M. de Oca, and M. Bourdelles, "Events-Based Security Monitoring Using MMT Tool," *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. 2012.
[22] M. Roesch and Others, "Snort: Lightweight intrusion detection for networks," in *Lisa*, 1999, vol. 99, pp. 229–238.
[23] V. Paxson, S. Campbell, J. Lee, and Others, "Bro intrusion detection system," Lawrence Berkeley National Laboratory, 2006 [Online]. Available: https://www.osti.gov/biblio/1245188
[24] M. Pelillo, Ed., *Similarity-Based Pattern Analysis and Recognition*. Springer, London, 2013.
[25] R. Bollapragada, J. Nocedal, D. Mudigere, H.-J. Shi, and P. T. P. Tang, "A Progressive Batching L-BFGS Method for Machine Learning," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, vol. 80, pp. 620–629.