Integrated Framework for Predictive and Collaborative Security

of Financial Infrastructures

Start Date of Project: 2018-05-01

Duration: 36 months

D3.4 Predictive Security Analytics Infrastructure

Deliverable Details	
Deliverable Number	D3.4
Deliverable Title	Predictive Security Analytics Infrastructure
Revision Number	1.0
Author(s)	ORT
Due Date	28/02/2019
Delivered Date	28/02/2019
Reviewed by	INNOV and FBK
Dissemination Level	PU
EC Project Officer	Christoph CASTEX

Contributing Partners	
1.	IBM (contributor)
2.	AS (contributor)
3.	INNOV (contributor)
4.	Z&P (contributor)
5.	NRS (contributor)
6.	CNR (contributor)
7.	CINI (contributor)

This project has received funding from the European Union's Horizon 2020 research and innovation programme 2014-2020 under grant agreement No 786727

Document Status

☒ draft

☒ WP leader accepted

☒ Consortium reviewed

☒ Project coordinator accepted

Revision History

Version	By	Date	Changes
0.1	ORT	11/10/2018	Initial ToC, bullet points
0.1	ORT	18/10/2018	First Draft
0.2	ORT	20/11/2018	V_02
0.3	ORT	29/11/2018	V_03
0.4	CNR, IBM,CINI	18/12/2018	V_04
0.5	NRS, ORT	03/01/2019	V_05
0.6	ORT	12/02/2019	V_06
0.7	NRS, ORT	18/02/2019	V_07
O.8	INNOV, ORT	19/02/2019	V_08
0.9	ORT	25/02/2019	V_09
1.0	GFT	28/02/2019	V_10 – Approved by Coordinator

Abbreviations

ANN	Artificial Neural Network
BSD	Berkeley Ssoftware Distribution (referred to License)
CERT	Computer Emergency Response Team
CI	Critical Infrastructure
CMDB	Configuration Management Database
CNN	Convolutional neural networks
CPS	Cyber Physical Security
CPTI	Cyber Physical Threat Intelligence
CSC	Common and Secure Communication
CSIRT	Computer Security Incident Response Team
CTI	Cyber Threat Intelligence
DaaS	Data as a Service
DPA	Data Protection Authority
DPO	Data Protection Officer
DSP	Digital Service Providers
EBA	European Banking Authority
ECB	European Central Bank
eTS	electronic Trust Services
EU	European Union
EUC	End-User Computing
GDPR	General Data Protection Regulation
IaaS	Infrastructure as a Service
ICT	Information Communication Technologies
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
ISMS	Information Security Management System
ISO	International Organization for Standardization
ISP	Internet Service Provider
KNN	K-Nearest Neighbour
ML	Machine Learning
MOA	Massive Online Analysis
NDA	Non-Disclosure Agreement
NIS	Network and Information Systems
OES	Operators of Essential Services
PaaS	Platform as a Service
PAN	Primary Account Number
PCA	Principal Component Analysis (PCA)
QTSP	Qualified Trust Service Provider
RBM	Boltzmann machine
RNN	Recurrent neural networks
RTS	Regulatory Technical Standard
SA	Supervisory Authority
SCA	Strong Customer Authentication
SECaaS	Security-as-a- Service
SME	Small and Medium-Sized Enterprises
SVM	Support Vector Machine (SVM)

Executive Summary

This deliverable is the outcome of Task 3.3. It describes the different algorithms and technological tools for data collection and predictive analytics for physical and cyber security and defines a new architecture for data collection and analysis when deployed and connected to the Reference Architecture of the FINSEC project.

This deliverable provides an overview of the different steps and necessary layers and tools for data collection and analysis methods for the FINSEC cyber/physical security with respect to expected latency and bandwidth requirements. Moreover, this deliverable addresses predictive analytics, describing the most relevant approaches to analyse the collected data, and finally detects attacks and anomaly patterns. One of the major contribution of this deliverable is the identification of an optimized data collection and analysis architecture meeting latency expectations.

In addition, one more important goal of the current deliverable consists in validating some components of our proposed data collection and analytics architecture. To reach this objective, we propose a demonstrator or a prototype to test some predictive and learning algorithms on an open source security dataset. Hence, we implemented some machine learning algorithms and tested them on this security dataset to show the obtained accuracy (close to 98%) on the training part of data. We aim to improve this result by testing and combining new machine learning algorithms. Our prototype is able to validate the different functionalities of the proposed data collection and analytics architecture.

TABLE OF CONTENTS

1. INTRODUCTION 7

2. PREDICTIVE ANALYTICS ALGORITHMS AND METHODS 8

2.1. STATISTICAL METHODS 8
2.1.1. UNIVARIATE MODELS 8
2.1.2. MULTIVARIATE MODELS 8
2.1.3. TIME SERIES MODELS 8
2.2. MACHINE LEARNING AND DEEP LEARNING METHODS 9
2.2.1. SUPERVISED LEARNING 9
2.2.2. UNSUPERVISED LEARNING 9
2.2.3. SEMI-SUPERVISED LEARNING 9
2.2.4. DEEP LEARNING 10
2.3. KNOWLEDGE-BASED METHODS 10

3. MACHINE LEARNING/DEEP LEARNING FOR FINSEC 10

3.1. MACHINE LEARNING TECHNIQUES FOR FINSEC 10
3.2. MACHINE LEARNING TOOLS: STATE OF THE ART 12
3.3. MACHINE LEARNING ALGORITHMS TO DETECT INNOVATIVE ATTACKS 14

4. DATA COLLECTION AND ANALYSIS MODEL 14

4.1. DATA COLLECTION REQUIREMENTS 14
4.2. ARCHITECTURE DESCRIPTION 15
4.2.1. DATA COLLECTION LEVEL 16
4.2.2. DATA PRE-PROCESSING LAYER 16
4.2.3. DATA ANALYSIS LAYER 17
4.3. DATA COLLECTION AND ANALYSIS: TECHNOLOGICAL VIEW 19
4.4. ADAPTIVE MULTI-LAYER DATA COLLECTION 21

5. DATABASES AND BIG DATA INFRASTRUCTURE FOR FINSEC 22

5.1. ARCHITECTURE DISCUSSION 22
5.2. ITERATIVE DATA MINING METHODOLOGY 23
5.2.1. REQUIRED TOOLS 23

6. PREDICTIVE ALGORITHMS: A DEMONSTRATOR/PROTOTYPE DESCRIPTION 25

6.1. DATASET DESCRIPTION 25
6.2. SOME OPERATIONS ON THE DATASET 26
6.3. PREDICTIVE ALGORITHMS: APPLICATION AND PRELIMINARY RESULTS 26

7. CONCLUSIONS 29

8. REFERENCES 30

List of Tables

Table 1 – Data Fusion Level 19

List of figures

Figure 1 Training and Testing phases for Machine Learning 11
Figure 2 A data collection and analysis model 15
Figure 3 Data Fusion operations 18
Figure 4 Data collection and analysis: a technological view 20
Figure 5 FINSEC's reference architecture: a logical view 21
Figure 6 Data collection and analysis: Big Data processing based on Result Polling 23
Figure 7 UNSW-NB15 Testbed (source: [Mous]) 25
Figure 8 Predictive and learning algorithms: Application on a dataset 27
Figure 9 Confusion matrix 28

1. Introduction

A main set of Cyber Physical Security (CPS) challenge stems from the fact that CPS is designed to interact with the physical world. Perhaps the most obvious of these is that the impact of attacks on a CPS can be physically catastrophic. In addition to threatening intellectual property (a problem that is common to all IT systems), attacks on CPS can adversely impact operational safety, and product performance. When compared with IT systems, this means there may be a different level of tolerance for threats against a CPS, and a different level of urgency in addressing attacks. A denial of service attack against a website produces loss of access to data, loss of revenue, or even damage to a server, but if the attack is addressed in minutes, recovery may not be difficult. By contrast, a denial of service attack against the system that regulates the safe operation of a power generation facility or an industrial plant can lead to irreparable damage to capital equipment that could take months or years to replace. For systems like these, the time scale for addressing the attack cannot be minutes. In the FINSEC Project, the necessary time to detect cyber-attack should be negligible and moreover, the time we need to react should be also close to few milliseconds. To attend this objective, FINSEC necessitates rapid tools and robust architectures to cope with these issues.

In addition, a CPS is sometimes deployed in ways that preclude physically securing all of their components. This increases the likelihood that cybersecurity processes will be operating in a compromised environment.

Because a CPS interacts with the physical world, they are subject to the time constraints of the physical process they are executing and monitoring. These processes are generally delay sensitive. As a result, security processes must fit within the time constraints of the application. Current IT cybersecurity controls may need to be modified significantly, because those solutions cannot meet the timing criteria required by the cyber physical systems. Further, the tight time constraints on addressing attacks largely rule out human-in-the-loop solutions. This drives the need for continuous, autonomous, **real-time monitoring** and **detection**.

In this context, the present deliverable focuses on the financial cyber security issues and addresses details on the predictive analytics presented by different algorithms based on machine learning and deep learning techniques. This necessitates the description of different algorithms and tools that impact the nature of the decision to take after the prediction.

The merge of cybersecurity (logical) and physical security using a data fusion operation is an interesting issue that the FINSEC project needs to address. Indeed, data fusion is often used to prepare the data before the training process, analysis and prediction.

This deliverable also investigates a new architecture for an adaptive data collection and analysis that will be mapped on the reference architecture of FINSEC. This architecture is based on a layered approach to distribute optimally the necessary components (data collector, data processing, etc.) with respect to the expected performance (latency and bandwidth). This architecture is then detailed and discussed and can be improved to reach best performances when deployed within the financial providers' infrastructures.

Nevertheless, one of the goals of this deliverable is to implement machine learning algorithms in order to test but also to validate some services or components of the described data collection and analysis architecture. To attend this objective, we propose an implementation of machine learning algorithms on a Kafka cluster to evaluate the performance of these algorithms.

The rest of this document is organized as follows: The next section (Section 2) is summarizing the existing and different approaches to predict and prevent future attacks and anomalies in cyber-physical systems. These approaches are based on statistical methods, machine learning, and knowledge-based methods. Before addressing the data collection methods, Section 3 is addressing

the different and relevant algorithms and tools of machine learning/deep learning that will be invoked and also used in FINSEC. In Section 4, we provide the requirements we need before collecting the corresponding data. In fact, these requirements allow us to reduce the volume of the data we have to collect according to some objectives or criteria . Moreover, and in the same section, a data collection and analysis architecture is provided clarifying the sequencing and processing steps presented in three layers (Data collection, pre-processing and analysis). The deployment of this architecture is discussed with respect to latency and bandwidth optimization. Section 5 is dedicated to databases and data mining infrastructure for FINSEC. The last Section (Section 6) is dedicated to the description of our first prototype or implementation of some machine learning algorithms operated by a simple Kafka cluster to realize and also to validate some components of our data collection and analysis architecture. The proposed prototype uses Python language on an open security dataset. Our implementation will show the importance of the selected machine learning algorithm to reach higher accuracy.

2. Predictive Analytics algorithms and methods

The methods of security data analytics with the purpose of detecting attacks can be classified into three main categories (statistical methods, machine learning/deep learning methods, and knowledge-based methods) as described in the followings.

2.1. Statistical methods

In these approaches, network traffic activity is captured and a profile representing its normal behaviour is generated. This is done using metrics such as packet level data and flow level data. For instance, a statistical inference is applied to calculate an anomaly score which is generated based on currently observed traffic. If the score is upper than a given threshold, then an alarm of anomaly is generated. We distinguish three relevant models applied in statistical methods: **Univariate models**, **multivariate models** and **time series models.** We describe briefly each of the cited models as follows:

2.1.1. Univariate models

These models need prior knowledge of an underlying distribution of data and estimate parameters (mean and standard deviation) from given data.

2.1.2. Multivariate models

They consider correlations between two or more metrics and do not need prior knowledge of an underlying distribution.

2.1.3. Time series models

These models use an interval time combining with an event counter or a resource measure, and they consider inter-arrivals times of observations as well as their values.

There are several used examples of statistical methods in attack detection. One can cite the information entropy, which consists of summarizing the traffic distribution by capturing the important characteristics of traffic features. Entropy-based methods are suitable for detecting attacks launched by Botnet based on anomalous patterns in networks.

Another statistical method is Cumulative Sum (or CUSUM) algorithm which is a sequential technique used to detect irregular changes in traffic traces.

The statistical methods have some number of advantages. We can cite:

- They do not require prior knowledge of network attacks. Hence, they are capable to detect zero-day attacks
- They use few features to characterize the network traffic leading to considerable reduction of their time and space complexity

These methods have also some drawbacks that we cite below:

- These methods can be trained by an attacker
- An appropriate threshold is difficult to set in order to better balance false positives and false negatives

2.2. Machine learning and deep learning methods

The aim of machine learning is to establish an explicit or implicit model of analysed patterns. Machine learning approach can be divided into three categories:

2.2.1. Supervised learning

In this type of machine learning, the used algorithm will learn knowledge from labelled data and then uses the obtained knowledge to classify the unknown data. There are various supervised learning algorithms such as:

- Support Vector Machine (SVM)
- K-Nearest Neighbour (KNN)
- Artificial Neural Network (ANN)
- Decision Tree (Random Forests)

2.2.2. Unsupervised learning

In this family of learning techniques, the used algorithms can find the underlying data structure without the need for the user to annotate (“label”) the data used in the training process. The most used methods or algorithms in this category are:

- K-means
- K-medoids
- BIRCH
- Chameleon
- DBSCAN
- OPTICS
- STING
- CLIQUE

2.2.3. Semi-supervised learning

In this type of learning, we consider a portion of labelled data mixed into a large amount of unlabelled data to generate training datasets for unsupervised learning.

The machine learning methods have a certain number of advantages and we cite some of them in the following:

- These methods have high detection rate
- They are adaptive: they are capable of updating their execution processes according to the new traffic

Nevertheless, they have also some disadvantages such as:

- The supervised learning cannot detect unknown attacks until relevant information is fed for retraining
- These methods consume more resources in both training and updating processes

2.2.4. Deep learning

Deep Learning is a subfield of machine learning whose algorithm is inspired by the structure and function of the brain called artificial neural networks. The algorithm takes metadata as an input and processes the data through a number of layers of the non-linear transformation of the input data to compute the output. It has a unique feature i.e. automatic feature extraction, which enables it to automatically grasp the relevant features required for the solution of the problem, reducing the burden on the programmer to select the features explicitly. The algorithm can be used to solve supervised, unsupervised or semi-supervised type of problems. Deep learning-based systems using self-taught learning have proved promising in detecting unknown network intrusions.

2.3. Knowledge-based methods

In these approaches, network or host events are matched with predefined attack rules or signatures to examine them for the presence of known attack instances. The most used knowledge-based methods is the **Expert System.** This extracts the specific features from training data and builds a rule classifying new coming data. There exists also another approach noted by **ontology analysis** and consists of expressing the relationships between collected data and using them to infer particular attack types. Another approach is noted by **logic analysis** and consists of expressing logic structure and using this structure to determine whether network events are legal.

As we discussed the advantages and drawbacks of the cited methods, the knowledge-based methods have also some advantages such as:

- They are simple and robust
- They have a high detection rate

Unfortunately, these methods have also some drawbacks such as:

- They cannot detect unknown attacks
- These methods may trigger some false alarms due to non-availability of attack datasets.

3. Machine Learning/Deep Learning for FINSEC

3.1. Machine Learning Techniques for FINSEC

One of the most relevant components in the data collection and analysis architecture consists of analysing and processing the data in the Data Processing Layer. This component noted by “Machine

Learning" in the "Global Analysis System" will be used to analyse the collected data provided and normalized by the "Data Collector" on the cyber physical system (see Figure 2 in Section 4).

A big picture, and without loss of generality, on how machine learning algorithms are working on a given data set, is depicted in Figure 1.

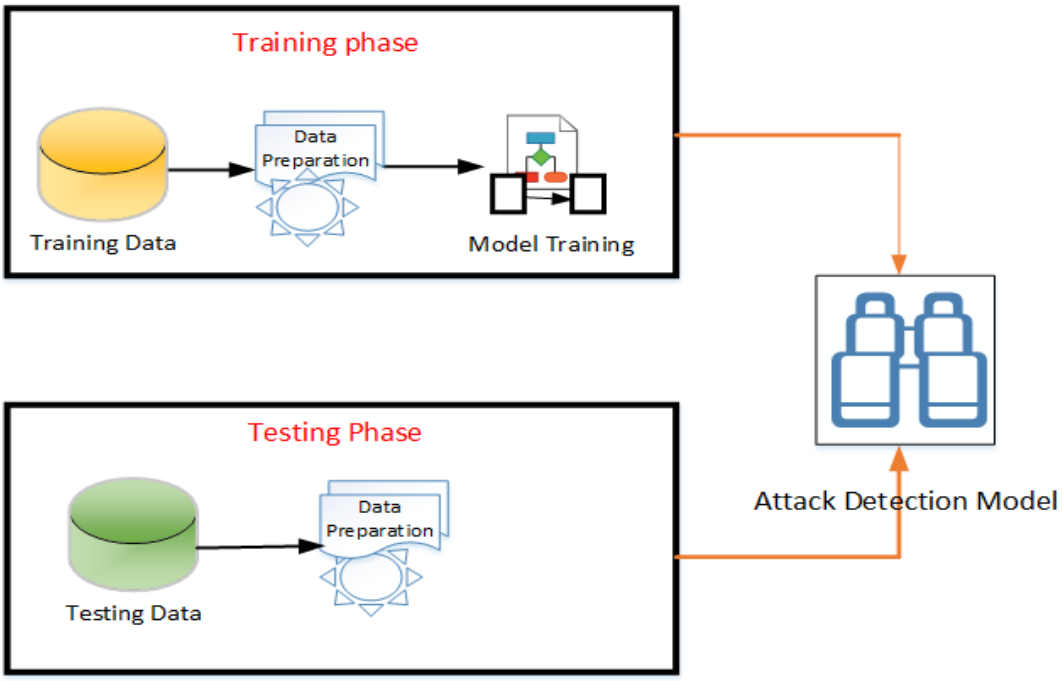

Figure 1 Training and Testing phases for Machine Learning

Indeed, and after data normalization and filtering, one can observe two essential steps that should be used before proposing an attack detection model:

- **Training phase**: this phase consists of collecting security event data for training our model. Our ML/DL algorithms and their combination will be applied on the prepared training data to detect eventual anomalies or attacks according to some added rules. This phase will result in an attack detection model that will be validated and approved by the data testing set.
- **Testing phase**: This phase proposes a data set to test the detection model found in the Training phase. When the data preparation is complete, then we apply the learning algorithms adjusted in the previous phase, to validate and improve the accuracy and the message/decision to be highlighted.

Nevertheless, and despite the performance that can be guaranteed by the attack detection model illustrated in Figure 1, some applied machine learning or deep learning algorithms can be inconsistent for different problems or for some data sets. Therefore, a machine learning algorithm may perform well for one type of cybersecurity analysis but may not perform well in another type of problems or data sets. Then, the algorithms selection is challenging in this context and has the merit to be addressed carefully and in a dynamic manner to improve the attack detection model performance and accuracy. In other words, the algorithms selection consists of identifying a good trade-off among various system's qualities (performance, accuracy …).

Thus, we can compare various types of machine learning algorithms, but also combine some of them to improve the quality of the expected result.

3.2. Machine learning tools: state of the art

To operate and run machine learning algorithms, different frameworks exist in the literature, and each framework may be performant for some data analytics objectives but this can change when addressing different data sets.

In the following, we summarize the most well known frameworks for machine learning in general and for Deep Learning in particular. These frameworks are described as follows:

1. **Apache Singa**: is a general distributed deep learning platform for training big deep learning models over large datasets. It is designed with an intuitive programming model based on the layer abstraction. A variety of popular deep learning models are supported, namely feed-forward models including convolutional neural networks (CNN), energy models like restricted Boltzmann machine (RBM), and recurrent neural networks (RNN)
2. **Amazon Machine Learning**: is a service that makes it easy for developers of all skill levels to use machine learning technology. Amazon Machine Learning provides visualization tools and wizards that guide you through the process of creating machine learning (ML) models without having to learn complex ML algorithms and technology. It connects to data stored in Amazon S3, Redshift, or RDS, and can run binary classification, multiclass categorization, or regression on said data to create a model.
3. **Azure ML Studio**: allows Microsoft Azure users to create and train models, then turn them into APIs that can be consumed by other services. Users get up to 10GB of storage per account for model data, although you can also connect your own Azure storage to the service for larger models. A wide range of algorithms are available, courtesy of both Microsoft and third parties.
4. **Caffe**: is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Models and optimization are defined by configuration without hard-coding & user can switch between CPU and GPU. Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process over 60M images per day with a single NVIDIA K40 GPU.
5. **Massive Online Analysis (MOA)**: is the most popular open source framework for data stream mining, with a very active growing community. It includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, concept drift detection and recommender systems) and tools for evaluation.
6. **Spark:** is Apache Spark's machine learning library. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs.
7. **MLpack**: a C++-based machine learning library originally rolled out in 2011 and designed for "scalability, speed, and ease-of-use," according to the library's creators. Implementing mlpack can be done through a cache of command-line executables for quick-and-dirty, "black box" operations, or with a C++ API for more sophisticated work. Mlpack provides these algorithms as simple command-line programs and C++ classes which can then be integrated into larger-scale machine learning solutions.
8. **Scikit-Learn**: leverages Python's breadth by building on top of several existing Python packages — NumPy, SciPy, and matplotlib — for math and science work. The resulting libraries can be used either for interactive "workbench" applications or be embedded into other software and reused. The kit is available under a BSD license, so it's fully open and reusable.

Scikit-learn includes tools for many of the standard machine-learning *tasks* (such as clustering, classification, regression, etc.).

9. **Theano**: is a Python library that lets you to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays (numpy.ndarray). Using Theano it is possible to attain speeds rivalling hand-crafted C implementations for problems involving large amounts of data. It was written at the LISA lab to support rapid development of efficient machine learning algorithms. Theano is named after the Greek mathematician, who may have been Pythagoras' wife. Theano is released under a BSD license.
10. **TensorFlow**: is an open source software library for numerical computation using data flow graphs. TensorFlow implements what are called data flow graphs, where batches of data ("tensors") can be processed by a series of algorithms described by a graph. The movements of the data through the system are called "flows" — hence, the name. Graphs can be assembled with C++ or Python and can be processed on CPUs or GPUs.
11. **TensorBoard** is a set of tools that allows graphical representation of different aspects and stages of machine learning in TensorFlow. The graphical interface facilitates the model training monitoring. A graph visualizer shows the model structure with graphs, allowing to ensure that the model components are located and connected properly. This approach simplifies the user experience during the performance evaluation of the model, especially for models of complex structures. In addition, TensorBoard allows to monitor how a model performs when its hyperparameters slightly change, making it possible to choose the hyperparameters that make the model perform best. [https://www.tensorflow.org/guide/summaries_and_tensorboard]
12. **H20.ai**: makes it possible for anyone to easily apply mathematics and predictive analytics to solve today's most challenging business problems. It intelligently combines unique features not currently found in other machine learning platforms including: Best of Breed Open Source Technology, Easy-to-use WebUI and Familiar Interfaces, Data Agnostic Support for all Common Database and File Types. With H2O, you can work with your existing languages and tools. Further, you can extend the platform seamlessly into your Hadoop environments.
13. **Keras:** is an open-source Python deep learning library which can run on top off Theano and TensorFlow. It was developed with the objective of increasing the execution speed of machine learning experiments. Its user-friendly interface and the division of networks into sequences of separate modules simplifies the user experience during the design of the prototype. Modules are easy to create and add to the networks models. [https://keras.io/]
14. PyTorch: is an open-source machine learning framework for deep neural networks built on Torch that supports GPUs acceleration and Python language. Unlike other tools (such as TensorFlow, Theano and Caffe) PyTorch models are built as dynamic computational graphs, thus supporting the change of the neural network behaviour at runtime without the need of static rebuilding of the whole model. This choice allows reduced lags and computational overhead. [https://pytorch.org/]
15. **Shogun:** is one of the oldest tool for machine learning, written in C++, although it supports a plethora of programming languages such as C#, Octave, Python, Java, Ruby, R. Shogun is open-source and provides data structures and algorithms for regression (such as Kernel Ridge Regression), pre-processing, visualization, model selection strategies (such as forward selection, Least Angle Regression), clustering algorithms (such as k-means and Gaussian Mixture Model), one-time classification and multi-class classification (such as Support Vector Machines and K-Nearest Neighbor). [http://www.shogun-toolbox.org/]

In FINSEC project, we propose to focus on two ML/DL frameworks (H2O.ai and TensorFlow) for data analytics and prediction.

3.3. Machine learning algorithms to detect innovative attacks

Machine learning algorithms are used in many contexts where statistical based methods are required to progressively improve performance and efficiency. In the intrusion detection system area, machine learning is usually adopted for anomaly-based detection. Machine learning algorithms may be adopted indeed to identify running Cyber and Physical Attacks, although detection in real-time may not be possible in some cases. In particular, two algorithms have been studied that can be used to identify attacks in progress by analyzing network traffic. These algorithms are called Principal Component Analysis and Mutual Information.

The Principal Component Analysis (PCA) is a statistical function already known in the intrusion detection field. It maps a coordinate space into a new coordinate system with axes commonly known as principal components (PCs). Such axes point in the direction of maximum variance of the original data. In particular, the first PC identifies the greatest data variance in a single direction, the second one is relative to the second greatest degree of variance, and so on. The retrieved PCs are ordered by the amount of data variance they identify. Typically, the first PCs contribute most of the variance in the original data set so that we can describe them with only these PCs, neglecting the others, with minimal loss of variance. Once the PCA is computed, given a set of data and its associated coordinate space, it is possible to perform a data transformation by projecting them onto the new axes.

Also Mutual information is not new in intrusion detection applications. This metric may help the traffic profiling in the presence of anomalies. Mutual Information could be adopted to understand the dependence between the analyzed variables. By combining these two approaches, it may be possible to implement intrusion detection systems able to identify innovative attacks.

4. Data Collection and analysis model

4.1. Data Collection requirements

Before going through data collection in a cyber and physical system, one may verify a set of data requirements that are identified and summarized below:

- **Efficiency:** On one hand, the collected data should be compact, in other words, the unnecessary data that are useless in attack detection should not be collected. On the other hand, the needed data should be collected or calculated in a real-time and high-speed manner to decrease the time delay of attack detection.
- **Privacy:** In the data collection process, the sensitive information of some particular data should be protected.
- **Resource consumption:** The consumption of resources including power, memory, and network bandwidth in the process of data collection and data communication should be well considered

In the following, we propose a data collection and analysis architecture clarifying the data collection in tandem with the data analysis and modules for security predictions. This is depicted in Figure 2.

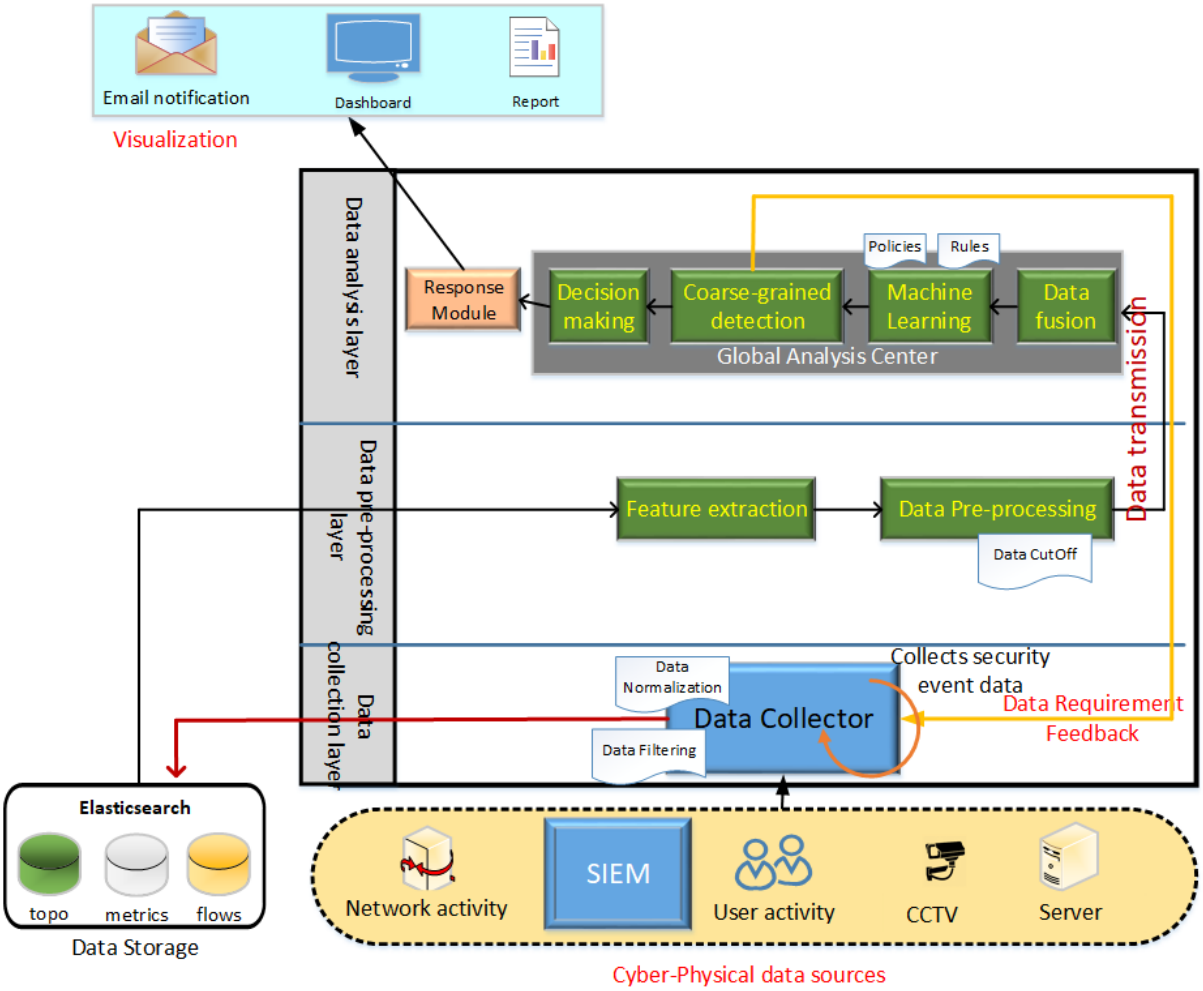

Figure 2 A data collection and analysis model

It is important to notice that this architecture (Figure 2) will be detailed in Figure 4 (see below) to provide a technological point of view on the data collection and analysis architecture. For instance, the "Global Analysis Center" will contain two types of detection:

- Coarse-grained detection: This a data analysis using, for example, a threshold to detect the existence of anomalies in real time. Hence, the processing is rapidly done.
- Fine-grained detection: This step is executed in case of anomaly detection in the previous step. Then, we need to take more time to better explore more data volumes.

4.2. Architecture Description

In the following we describe our architecture (depicted in Figure 2) and propose to discuss the data collection and analysis infrastructure when considering big data issues.

Our data model or architecture for a data collection and analysis (see Figure 2) is composed by three levels given as follows:

4.2.1. Data Collection Level

This layer composed of a Data Collector is fed by different cyber and physical probes (Network protocols, CCTV, servers…). The Data Collector consists of collecting security event data that will be stored in data storage server or in a dedicated storage data base. This data base is updated from time to time to be used by the other two layers, and this will be described in the sequel.

Before sending data from the collector to the storage data base, the data collector has to prepare the data by using normalization and filtering operations (that can be hosted by a Kafka cluster, for example) in order to facilitate the data exploitation by other components of our architecture.

Thus, the data sent from the Data Collector to this storage data base, should be secured using encryption for instance. This induces a communication latency to be considered in this operation as this can impact the performance of our architecture.

4.2.1.1. Data Collection Probes

Various cyber and physical probes take part in forming a comprehensive, holistic and accurate security image. To accommodate an adaptive approach, these probes should allow for different granularity and resolution of monitored data collection.

4.2.1.1.1. Skydive

Skydive agents act as data collectors employing efficient mechanisms to control the granularity of data collected and collection intrusiveness, in terms of CPU, memory and network overheads. These mechanisms allow for extra flexibility in capturing network topology and network flows data, as compared to other existing methods. At the same time, they enable capturing rich sets of metadata attributes from network entities, both in the physical and in the virtual domains.

Skydive architecture allows to capture information in various locations of the infrastructure, both horizontally (network entities, e.g. interfaces, switches etc., on the same layer of the stack) and vertically (network entities on multiple layers of the stack, e.g. application layer, virtual networking layer, physical networking layer, etc.). Having data collected with flexible granularity on the one hand and with high redundancy on the other allows us to correlate information between locations and layers and to use various algorithms (see Creating multivariate models as described in [2.1]**Error! Reference source not found.**) to produce insights.

Skydive can also be used to validate that security rules in the network policy are working as expected.

4.2.2. Data Pre-processing Layer

The objective of this layer is to prepare the processing of the data provided by the dedicated storage data base. This layer is composed of various operations with the goal of preparing the data processing. One can cite feature extraction which consists of selecting the most relevant features that should be identified and then will be considered in the learning analytics of the processing Layer. This feature extraction will help to reduce the volume and the size of the data to be analyzed. Another operation in the pre-processing layer consists in applying a certain number of rules. These rules can be statically defined but also can be generated dynamically and according to the objectives we would like to attend. For instance, Data CutOff operation (this can be triggered by the Global Analysis System when running a coarse grained detection) can be applied to eliminate the part of the unnecessary data we do not need for our analysis. At the end of these operations, the pre-processing layer is then able to

send (with a given latency) the pre-processed data to the processing and analyzing layer. This is detailed in the following.

4.2.3. Data Analysis Layer

This layer is composed of a Global Analysis Center and a Response module. We describe these components by starting with the "Global Analysis Center" composed of:

- ***Data fusion module***: The processing layer is fed by the homogeneous data of the pre-processing layer. This module consists of merging the cybersecurity data with the physical security data provided by the different probes as mentioned earlier. Combining these types of data is a crucial step before starting the analysis and learning processes. Indeed, some important information and messages are hidden and the fusion of the data allow to clarify and distinguish serious issues and patterns that will be discovered by the learning analytics following these operations.
- ***Machine Learning algorithms***: In this module, we deploy several families of learning analytics algorithms based on machine learning. It is important to deploy and use fast algorithms to detect anomalies and attacks very quickly, nevertheless, even if some algorithms can converge slowly, but their combination with other fast approaches can lead to more efficient attacks or anomalies detection in the concerned cyber-physical system. Note that the architecture of deploying and implementing the different servers for machine learning algorithms should take into account an important element which is big-data treatment in few seconds. In fact, this is important when addressing large volumes of data containing relevant information to be depicted. Thus, different architectures of this module will be proposed to better cope with big-data analysis and this can be reached using data split and then analyzed using Map-Reduce processes, or parallel servers dedicated to the analysis of data blocks, before merging the different results. This part, will be described in next sections of this document.
- ***Coarse-grained Detection***: After the convergence of the used multiple learning algorithms, the obtained result can inform us if there is some suspicious attack or anomalies detection (using a threshold for instance). In case of eventual existing attacks, and to reinforce the decision of our algorithms, this module can ask the Data Collector for more data. This will be used and analyzed before confirming or not the existence of attacks or anomalies in our system. The data feedback requested by the Coarse-grained module consists of adjusting the accuracy of the data rate that should be used to collect different data types. Next to this, a new analysis (that is represented by "Machine Learning Analysis in Figure 4 which is also a **Fine Grained Detection**) will be run and deeply explore more collected data before taking a decision.
- ***Decision making module***: This module is showing the result of the smart analysis operated by the previous modules in different steps. A decision aid is then proposed and sent to the Response Module.

The Response Module has the role to show and display in different manners (send email, make an alert, provide a report, …) and an action will be taken accordingly.

For the sake of clarity, we describe the main elements of the architecture of Figure 2 and provide their internal operations for each data collection and analysis run.

Hence, we distinguish four most relevant elements to be described in the sequel:

- **Data pre-processing**:

  The collected data from a cyber physical system is often susceptible to noise, missing values and inconsistency. We already mentioned above, that the quality and the accuracy of the data affect

the quality of the detection model. Hence improving the quality of the concerned collected data will improve the results and the decision' quality after analytics and predictions.

The data pre-processing approaches are divided into the following categories:

- **Data Cleaning**: this operation is useful when the collected data is suffering from missing values, noisy data, incomplete data, …It consists of different solutions (fill in the missing values, clustering to detect outliers, …)
- **Data Integration**: which consists of combining data from multiple storage databases.
- **Data transformation**: this step consists of transforming the data into an appropriate form for analytics.
- **Data reduction**: This is helpful to reduce the data volume or number of attributes that should be analysed.

- **Feature extraction**: The feature extraction problem can be split into two sub-problems as feature construction and feature selection:
  - **Feature construction**:

    Feature construction is one of the key steps in the data analysis process, conditioning the success of any machine learning endeavour. Indeed, one should beware of not losing information at the feature construction stage. To convert 'raw' data into a set of useful features, human expertise is often required. Hence, we would like to complement this by automatic feature construction methods. Thus, feature construction is about how to know when a feature is relevant or informative.

  - **Feature selection**:

    This step is primarily performed to select a subset of relevant and informative features. It can have the following motivations:

    - General data reduction: to limit storage requirements and increase algorithm speed
    - Feature set reduction: **to save resources in the next round of data collection or during utilization**
    - Performance improvement: to gain in predictive accuracy
    - Data understanding: to gain knowledge about the process that generated the data or simply visualize the data
- **Data fusion**:

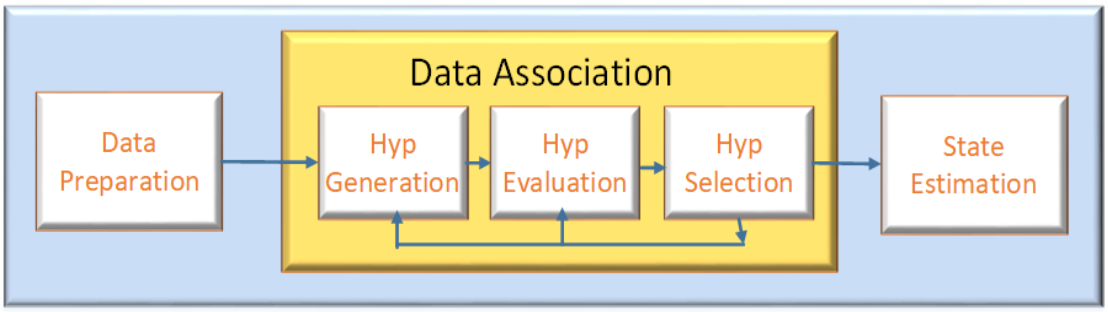

Figure 3 Data Fusion operations

Data fusion process is represented by nodes, and each node can include processes of Data Preparation (for instance, error correction, formatting, …), data association and state estimation.

Data Association in turn involves three sub-functions: hypothesis generation, Evaluation and Selection. We note by 'hypothesis' an association that a particular set of data constitutes all the reports maintained by the system that are relevant to estimating the state of the entity.

State estimation functions differ broadly according to the classes of objects whose states are to be estimated into data fusion "levels" described in the following table:

Table 1 – Data Fusion Level

Level	Data Fusion Level	Description
0	Feature assessment	Estimate patterns: features of an observed region
1	Individual entity assessment	Estimate states of entities considered as individuals
2	Situation assessment	Estimate situations: interactions, other relationships and aggregations
3	Impact assessment	Estimate scenarios and outcomes
4	System assessment	A system's estimation of the state of sensors and other information sources, of other resources and processes of that system

- **ML/DL techniques**:

  After the above-described steps, the new data set is now ready to be analysed in order to predict or analyse a situation according to different identified objectives. Then, different learning or predictive algorithms can be applied using the new data set. These algorithms (Random forest, DNN, KNN, …) are already mentioned in Section 2 of this deliverable, but more details can be found in [MLBook].

  After clarifying the roles of these elements, we propose to define a mapping of technological view of our architecture of data collection and analysis (see Figure 2), to the reference architecture of FINSEC project provided in D2.4 document and also provided in Figure 5 for the sake of clarity.

  This mapping will be assessed using simulation to confirm the expected performance of FINSEC reference architecture when dealing with data collection and analysis operations.

4.3. Data Collection and Analysis: Technological View

Figure 4 depicts the tools and technologies that FINSEC project will utilize to realize and implement the architecture shown in Figure 2. We describe in Figure 4 different components and make a focus in our demonstrator (see Section 6) on the learning and predictive algorithms that will be used.

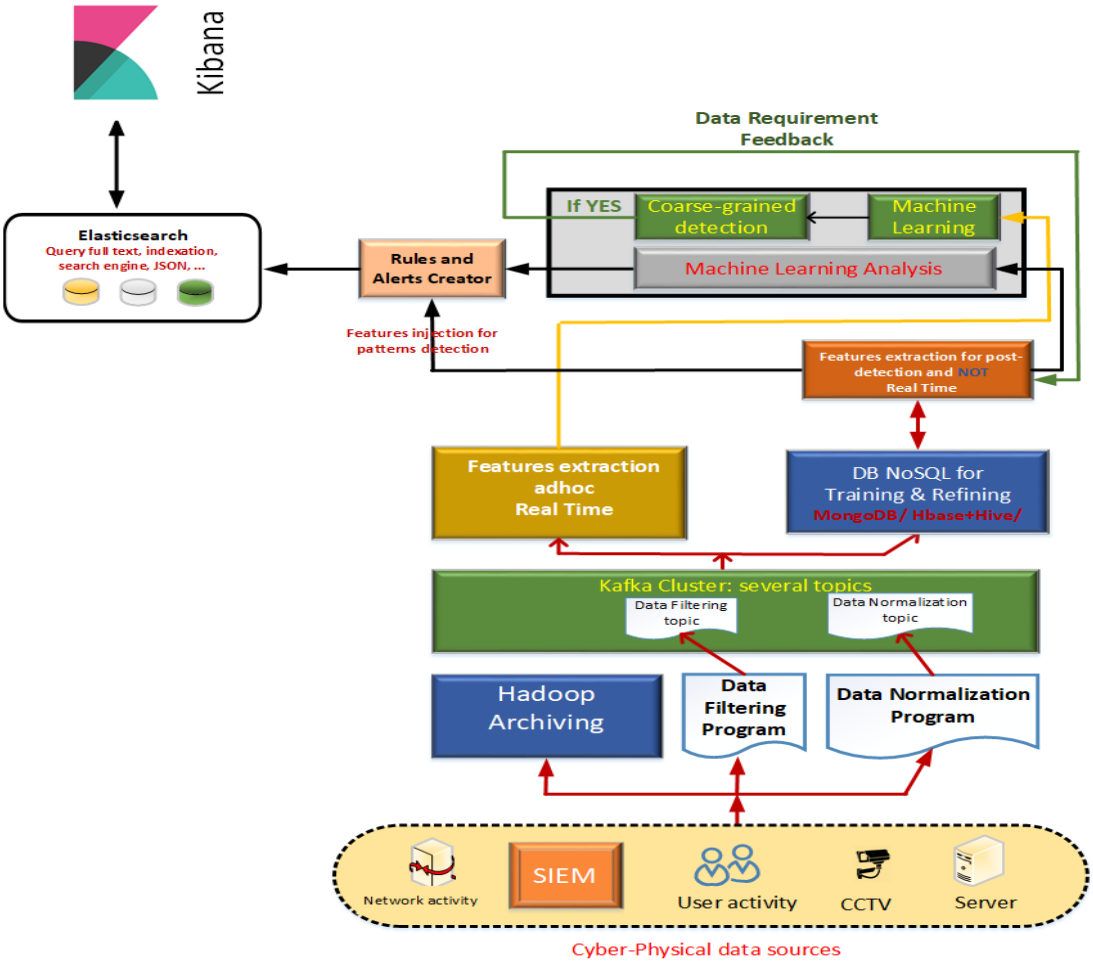

Figure 4 Data collection and analysis: a technological view

In the following, we are going to answer this question: How this architecture (Figure 4) will fit in the FINSEC RA (Figure 5)?

Starting (left side) from the main components or services of Figure 4, we propose a mapping on the already defined components of the RA of FINSEC (Figure 5). This provided as follows:

- **Physical and Cyber data sources (Data collection)**: this part can easily fit in the ”End-Users infrastructure (FINSEC data collection)” as it is actually the same functionality based on data collection from different probes.
- **Normalization, Filtering, … programs** can fit in the service indicated by Normalization/Incident/Events in Figure 5.
- **Hadoop + Spark**: this component that will be used to archive or store the collected data before analysis, can fit in ‘Big Data Infrastructure’ in FINSEC RA.
- **Kafka Pipeline** this pipeline will be used to ensure the Transversal/Vertical connectivity buses.
- We proposed to use NoSQL DB in our architecture and this can fit in “DB” component in FINSEC RA.
- **Feature extraction** that can be realized using Hadoop/**PySpark** approaches: this service can be implemented in the “Big Data Infrastructure” component.

- **Machine Learning Analysis** which one of the most important roles in our proposed architecture can fit in "Big Data Architecture (Hadoop/Spark Cluster)"
- **ElasticSearch+ Kibana** is proposed in our architecture and can fit in the "Predictive Analytics" component of the FINSEC RA.

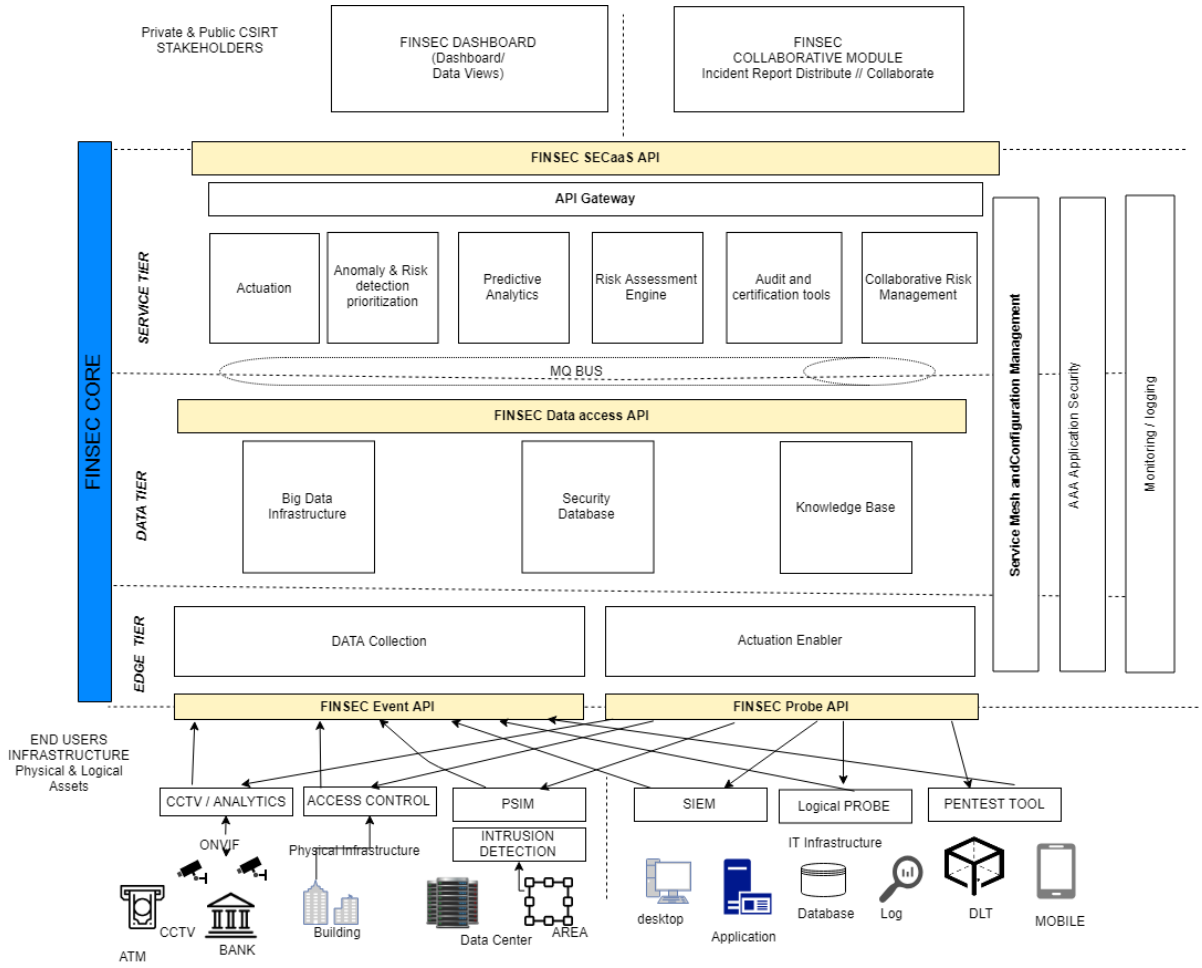

Figure 5 FINSEC's reference architecture: a logical view

4.4. Adaptive Multi-layer data collection

This topic will be elaborated and detailed in D3.1. For the sake of clarity we here give a brief description.

To secure financial critical infrastructures and services, security related data must be collected and analyzed in an intelligent, resilient, reliable, secure and timely manner fulfilling all the communication requirements and standards. The detection capability can greatly be improved by correlating wide-ranging data sources and by multi-layer data collection and analytics. To achieve this, the FINSEC approach is the design and deployment of the concept of adaptive multi-layer data collection by adapting different approaches. Adaptability refers to that a collection mechanism can adjust to different environmental contexts and situations. FINSEC will integrate smart security probes and a set of adaptive strategies for the multi-layer data collection. These include how to render the data collection adaptivity and intelligence, optimize bandwidth and storage of security information, and boost the intelligence of the project's probes by ensuring multi-layer data collection. Security data

analytics methods described in D2.3 will be integrated at appropriate level specific analytics. Predictive/regression algorithms such as linear regression, Support Vector Regression (SVR), logistic regression, KNN regression will be investigated for the lightweight analysis of adaptive strategies. Deep learning mechanisms will be used for the identification of complex risk and attack patterns. A set of rules (both static and adaptive) will be defined for processing, analyzing, configuration, collection, and adaptation. Automated adaptive multi-layer data collection with optimized bandwidth and storage of security information will be achieved using various adaptive collection strategies such as security threats, content variation, collection/sampling rate, bandwidth variation/communication dynamics, application needs, context changes, and storage needs.

In terms of the FINSEC architecture we specifically designed our solution to achieve most of the quality attributes described in D2.3. For example we use a cloud based architecture that supports scalability by design and by implementation. We address the issue of false positives to ensure reliability and accuracy, and so on.

5. Databases and Big Data infrastructure for FINSEC

5.1. Architecture discussion

To cope with Big Data analysis issues in FINSEC project, we may need to address at least a smart and optimized strategy to be able to process and store huge volumes of data. Thus, in the following, we propose to discuss the architecture of Figure 5 based on Result Polling tactic.

This method in a security analytic system (FINSEC Reference Architecture) is composed by various components:

- **Data Collector**: This component (as described before) collects security event data from multiple sources incorporated within physical and logical systems. The collected data is then stored in Data Storage databases after have been normalized and filtered.
- **Parallel Data Processing**: This component (and after reading the data) consists of Mappers and Reducers that depend on the number of available nodes already deployed in a cluster. The mapper read the data in parallel and produce intermediate key-value pairs shown as the result in figure 6. When all the map jobs are completed, the key-value pairs are passed to the reducers. The reducers merge the various values to produce the final results.

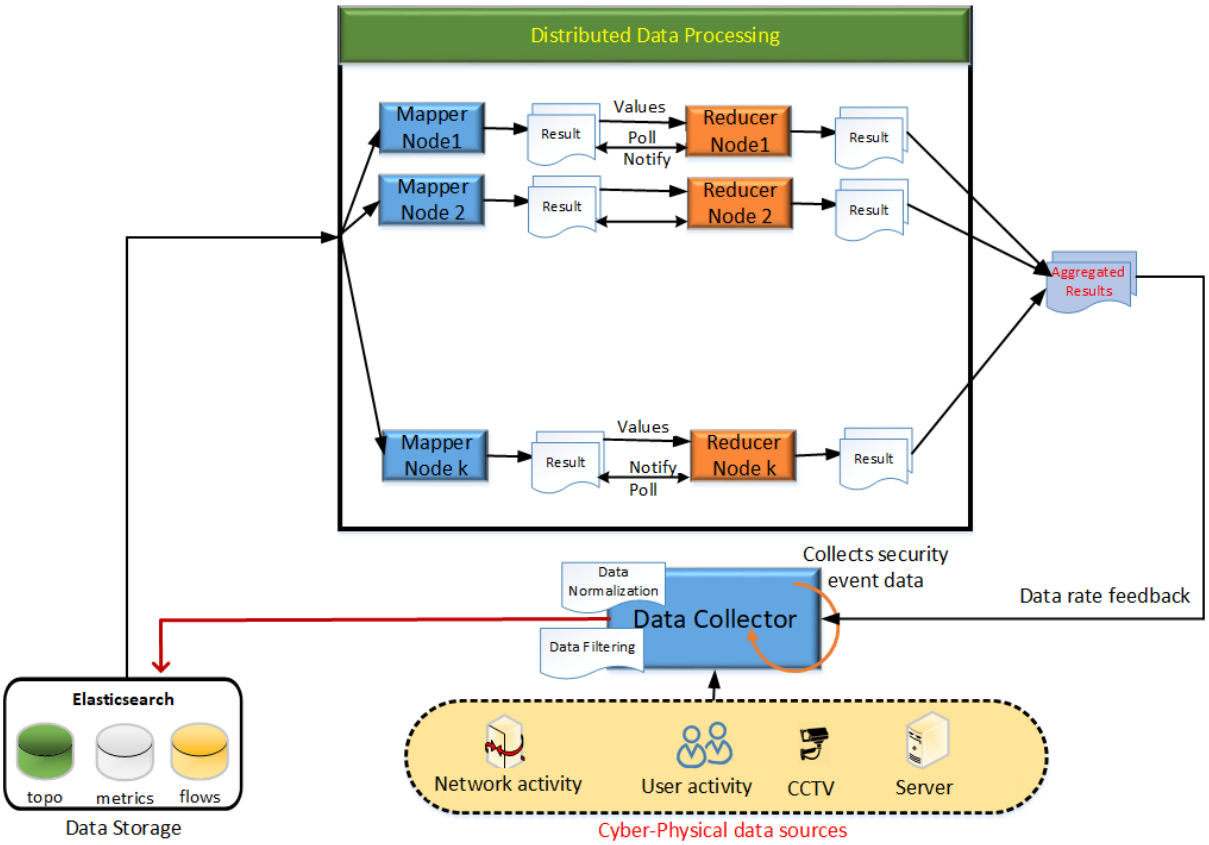

Figure 6 Data collection and analysis: Big Data processing based on Result Polling

Nevertheless, and normally, the reducers in Figure 6 have to wait until all map jobs are completed, and this can cause a significant delay in responding to an attack. To cope with this issue, the mechanism of poll and notify is then introduced. In this mechanism, the mappers monitor the changes in incoming security event data. As soon as the change in the security data crosses a predefined threshold, the mappers notify the reducers to get ready for taking the required data without waiting for completion of the predefined time interval. Moreover, and to better optimize notifications from mappers, the reducers can poll the mappers for results depending upon the processing capacity of the reducers. After receiving the intermediate key-value pairs from mappers, the reducers process the intermediate results to produce the final results. Immediately after the reducers produce the final results, we may check for possible cyber attack.

5.2. Iterative Data Mining Methodology

5.2.1. Required Tools

The expression Data Mining refers to the automatic techniques used to extract data from huge sources in order to elaborate them. Hadoop and Spark are the most prominent and used tools for distributed storage and BigData processing. Following, Hadoop and Spark will be discussed.

The **Apache Hadoop** software library is a framework mostly written in Java, with some native code in C and command line utilities written as shell scripts. It enables scalable distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. All the modules in Hadoop are designed with

a fundamental assumption that hardware failures are common occurrences and should be automatically detected and handled by the framework, thus delivering a highly-available service on top of a cluster of computers. In fact, Hadoop implements MapReduce, which is a programming model introduced by Google with the aim of processing and generating big data sets with a parallel and distributed algorithm on a cluster, providing redundancy and fault tolerance. MapReduce consists of two methods: Map and Reduce. Map performs data filtering and sorting, while Reduce performs a summary operation, reducing the clusters result into one output. Implementing MapReduce, Hadoop splits files into large blocks and distributes them across nodes in a cluster. The packaged code is transferred into nodes, which can then process the data in parallel. In this way, nodes access and manipulate only local data, therefore avoiding the performance bottlenecks caused by the extra network communication overhead [https://www.ibm.com/analytics/hadoop/hdfs] [http://ieeexplore.ieee.org/document/6877311/]. As a result the dataset processing is faster and more efficient than in a conventional supercomputer architecture, in which computation and data are distributed via high-speed networking.

The base Apache Hadoop framework includes four modules: Hadoop Common, Hadoop Distributed File System (HDFS), Hadoop YARN and Hadoop MapReduce. Hadoop Common contains libraries and utilities needed by the other modules. HDFS is a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster. Hadoop YARN is a platform responsible for managing computing resources in clusters and using them for scheduling users' applications. Hadoop MapReduce is an implementation of the MapReduce programming model for large-scale data processing. In addition to the base modules, nowadays Hadoop refers also to a set of others software packages that can be installed on top of Hadoop, including Apache Spark [https://hadoop.apache.org/].

Apache Spark is an open-source unified analytics engine for large-scale data processing. Apache Spark architecture is based on the Resilient Distributed Dataset (RDD). RDD is a multiset of data items distributed over a cluster of fault-tolerant machines.

Contrary to MapReduce programs (that read data from disk, map a function across the data, reduce the results and then store reduction results on disk), Spark functions as a working set for distributed programs that offers a restricted form of distributed shared memory [https://amplab.cs.berkeley.edu/wp-content/uploads/2011/06/Spark-Cluster-Computing-with-Working-Sets.pdf]. For this reason, Spark's RDD performance can be several order of magnitude better than MapReduce implementations, such as Apache Hadoop [http://or.nsfc.gov.cn/bitstream/00001903-5/423653/1/1000007146729.pdf]. Spark high performances (for both batch and streaming data) are also due to the use of a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine. As reported in [http://or.nsfc.gov.cn/bitstream/00001903-5/423653/1/1000007146729.pdf], Spark speed is to detriment of memory consumption.

Spark needs a cluster manager and a distributed storage system. It can run using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. It can access data in HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive and other data sources. Spark can be used interactively from the Scala, Python, R, and SQL shells. In addition, thanks to over 80 high-level operators, it is easy to write and build parallel apps. Libraries such as SQL and DataFrames, MLlib, GraphX and Spark Streaming can be combined in the same application [https://spark.apache.org/].

6. Predictive algorithms: a demonstrator/prototype description

In the following, we propose a prototype to test some learning and predictive algorithms when applied to an open source security dataset described in the sequel. The objective of this section consists in providing first implementations of learning algorithms already indicated in our proposed architecture in Figure 6. Indeed, in this deliverable, we aim to test and validate some services or components in the data collection and analysis architecture through an implementation of machine learning algorithms using a Kafka cluster. This implementation will check and then validate this architecture (see Figure 4) according to some metrics such as performance and accuracy.

We start our implementation as described above, nevertheless, the ambition consists in implementing the whole components and services of Figure 4 and its mapping (via various APIs) with the FINSEC reference architecture.

6.1. Dataset description

We used a security dataset named by UNSW-NB15 [Mous] generated by configuring an authentic testbed environment with the IXIA tool [IXIA] to simulate current representations of normal and abnormal network traffic. The generated data is a hybrid of real modern normal activities and synthetic contemporary attack behaviours. This is depicted in Figure 7.

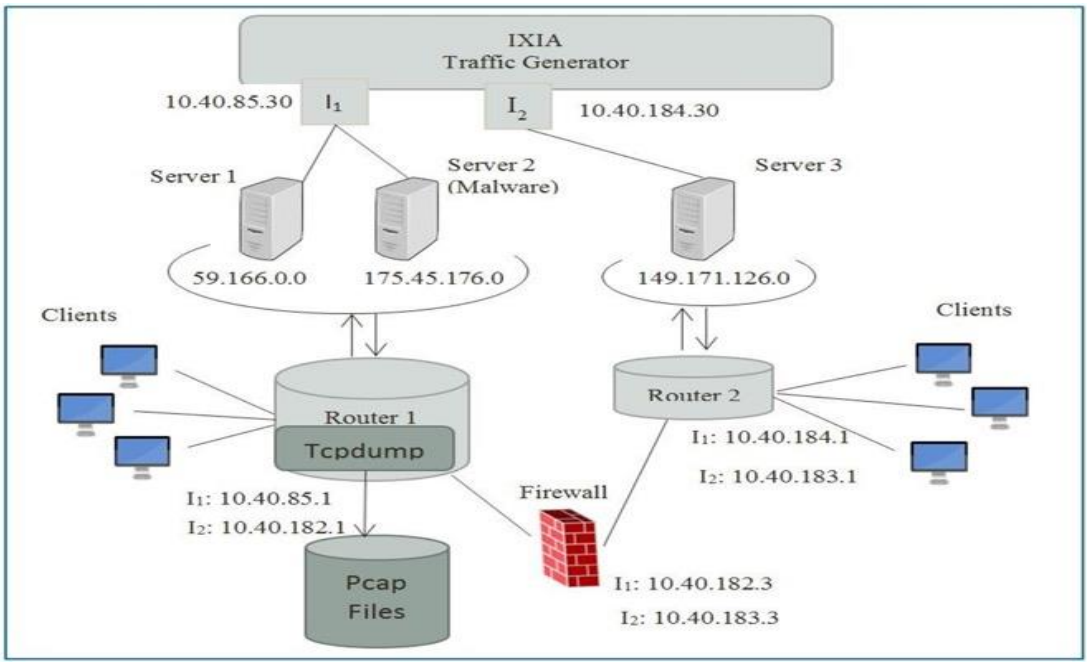

Figure 7 UNSW-NB15 Testbed (source: [Mous])

Tcpdump tool is used to collect and capture a volume of 100 GB of the raw traffic. This dataset has nine types of attacks namely Fuzzers, analysis backdoors, DoS, etc. It also contains more than **2 million observation**s and **49 features**.

We used Jupyter Notebook, to explore the dataset and provide preliminary statistics before applying learning algorithms. The following figure is showing a restricted representation of features and rows of our dataset:

Out[18]:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt
243	244	0.921987	ospf	-	INT	20	0	1280	0	20.607666	254	0	10551.125000	0.0	0	0	48.525633	0.0
244	245	0.921987	ospf	-	INT	20	0	1280	0	20.607666	254	0	10551.125000	0.0	0	0	48.525633	0.0
245	246	0.921987	ospf	-	INT	20	0	1280	0	20.607666	254	0	10551.125000	0.0	0	0	48.525633	0.0
246	247	0.921987	ospf	-	INT	20	0	1280	0	20.607666	254	0	10551.125000	0.0	0	0	48.525633	0.0
266	267	1.244202	ospf	-	INT	28	0	3024	0	21.700656	254	0	18749.367190	0.0	0	0	46.081555	0.0
267	268	1.244202	ospf	-	INT	28	0	3024	0	21.700656	254	0	18749.367190	0.0	0	0	46.081555	0.0

6.2. Some operations on the dataset

To better understand the content of our security dataset, we propose some manipulations to obtain relevant statistics and information on the behaviour of some features and observations at the same time. For this, we addressed the following operations:

- Feature selection by Principal Component Analysis, Association Rules Mining, …
- Aggregation of flow and protocols feature to accumulate appropriate network observations
- Feature creation to extract and generate the relevant attributes of normal and malicious activities

6.3. Predictive algorithms: Application and preliminary results

In our work and according to the objectives of FINSEC, we would like to investigate various approaches and machine learning algorithms and apply them on this dataset to illustrate the different impacts of these solutions to reach an optimized accuracy.

Thus, first we will use the **Random Forest** algorithm to compute the final accuracy result. Note that this algorithm is already described above (see Section 2).

We implemented a prototype fed by the described dataset and following different steps for predictive and learning analytics as discussed above. The following figure depicts the different training steps on the given dataset. After using the Random Forest approach, one can see in this figure, the obtained accuracy which is estimated to **93.82%.**

```
In [19]: ##Découpons le dataset en un training set et un test set 80/20 :
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

In [20]: X_train.head()

Out[20]:
```

id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt	sjit	
163083	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.0000	0	0	0.009000	0.000000	0.000000	0.00
14838	0.000005	2	0	114	0	200000.005100	254	0	9.120000e+07	0.0000	0	0	0.005000	0.000000	0.000000	0.00
75439	0.064837	4	4	2304	2304	107.963044	62	252	2.132116e+05	213211.5938	0	0	15.984000	14.002667	22.590646	19.78
120344	0.222508	10	10	760	4218	85.390189	62	252	2.459238e+04	136516.4375	2	3	24.723111	23.142667	1281.435985	1766.65
132516	0.000009	2	0	114	0	111111.107200	254	0	5.066666e+07	0.0000	0	0	0.009000	0.000000	0.000000	0.00

```
In [21]: y_train.head()

Out[21]: id
         163083    1
         14838     1
         75439     1
         120344    1
         132516    1
         Name: label, dtype: int64

In [23]: ##Procédons à présent au training :
         from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier()
         classifier.fit(X_train, y_train)

Out[23]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')

In [24]: ##Procédons maintenant à la prédiction relativement au test set :
         y_pred = classifier.predict(X_test)

In [25]: print(y_pred.shape)

         (51535,)

In [26]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test, y_pred)

Out[26]: 0.938294363054235
```

Figure 8 Predictive and learning algorithms: Application on a dataset

To confirm the obtained results (accuracy = 93.82%), we provide below the obtained confusion matrix (see Figure 9) indicating the number or rate of: True Positives, True Negatives, False Positives, and False Negatives depicted in Figure 9 by the couples (1,1), (1,0), (0,1), (0,0), respectively.

One can remark that our algorithm has found good results as the number of True Positives and False Negatives are very high compared to the number or rate of True Negatives and False Positives. In other words, the errors of our algorithm are negligible even if the used approach can be improved to attain higher accuracy scores.

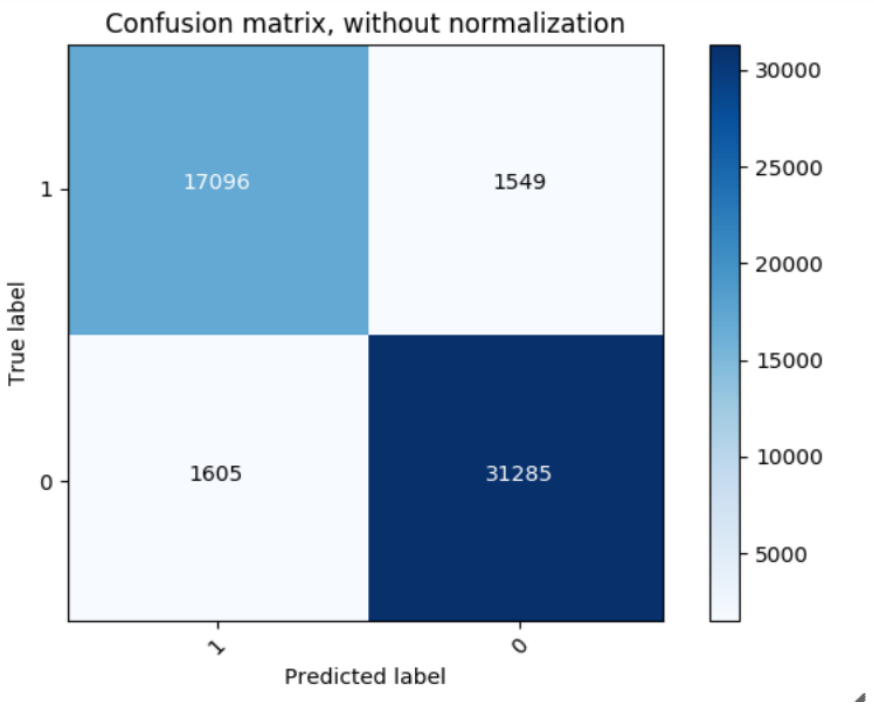

Figure 9 Confusion matrix

As a future work, we propose to use new predictive and learning algorithms in order to improve the obtained accuracy. This can be done using the following algorithms:

- Artificial Neural Networks
- Naïve Bayes
- AdaBoost

The testing step is time consuming and this is due to the dataset volume and the total number of features used to improve the accuracy result.

7. Conclusions

We discussed in this deliverable/document the most important approaches and algorithms used to predict future cyber attacks in a given system, when focusing on heterogeneous data collected from physical and logical parts. This predictive analytics should be applied to a collected set of data. This data has to be collected using various and smart probes according to some metrics already identified in this document. The collected data is realized using smart probes deployed on the logical and physical parts as described previously.

One of the major contributions of this deliverable consists in proposing a data collection logical architecture combined with learning and predictive analytics algorithms to cope with abnormal or attack situations. To validate some services or components of this architecture, we proposed a prototype/demonstrator when using a security dataset from the literature. Our objective using this demonstrator consists in testing our algorithms to improve their accuracy. This accuracy can be improved when using other learning and predictive approaches and especially when combining between different relevant machine learning approaches. Our implementation allowed us to validate some analytic components of Figure 4 using real data that will be provided by FINSEC partners, and we have the ambition to extend our prototype to other services (storage, visualization,…) when meeting the FINSEC reference architecture expectations.

8. References

[Lin 2018] H. Lin, Z. Yan, Y. Chen and L. Zhang, "A Survey on Network Security-Related Data Collection Technologies," in IEEE Access, vol. 6, pp. 18345-18365, 2018. doi: 10.1109/ACCESS.2018.2817921

[NIST 1500-201] NIST Special Publication 1500-201, Framework for Cyber-Physical Systems: Volume 1, Overview, Version 1.0, June 2017

[NIST 1500-202] NIST Special Publication 1500-202, Framework for Cyber-Physical Systems: Volume 2, Working Group Reports, Version 1.0, June 2017

[Bilbao] K. Peciña, A. Bilbao and E. Bilbao, "Physical and logical Security Risk Analysis model," *2011 Carnahan Conference on Security Technology*, Barcelona, 2011, pp. 1-7. doi: 10.1109/CCST.2011.6095895

[He-LTE] L. He, Z. Yan and M. Atiquzzaman, "LTE/LTE-A Network Security Data Collection and Analysis for Security Measurement: A Survey," in *IEEE Access*, vol. 6, pp. 4220-4242, 2018. doi: 10.1109/ACCESS.2018.2792534

[Gao-liu] Gao Liu, Zheng Yan, Witold Pedrycz, “Data collection for attack detection and security measurement in Mobile Ad Hoc Networks: A survey”, Journal of Network and Computer Applications,

Volume 105, Pages 105-122, 2018

[CISSP] https://sybextestbanks.wiley.com/courses/102/data/ebook.pdf

[H2O] https://www.h2o.ai/

[Tensorflow] https://www.tensorflow.org/

[Mous] Moustafa, Nour. Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic. Diss. *University of New South Wales, Canberra, Australia, 2017.*

[LinkDataSet] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[IXIA] https://www.ixiacom.com/products/ixnetwork

［MLBook］ Alpaydin Ethem, Introduction to Machine Learning, 2010.