D4.2

Initial Design and Development of Privacy Aware Analytics for Secure Services

WP4 – Privacy-Aware Analytics for Security and Services

SIFIS-Home

Secure Interoperable Full-Stack Internet of Things for Smart Home

Due date of deliverable: 30/03/2022
Actual submission date: 30/03/2022

Responsible partner: CNR
Editor: Paolo Mori
E-mail address: paolo.mori@iit.cnr.it

28/03/2022
Version 1.2

Project co-funded by the European Commission within the Horizon 2020 Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

The SIFIS-Home Project is supported by funding under the Horizon 2020 Framework Program of the European Commission SU-ICT-02-2020 GA 952652

Authors: Paolo Mori, W. Alabbasi, G. Giorgi, A. Saracino (CNR), A. Monge Roffarello, L. De Russis (POLITO), O. Isohanni, J. Jämsä (CEN), G. Cirillo, S. Robinson, O. Waltari (FSEC), S. Stennud (RIOTS)

Approved by: Marco Tiloca (RISE), G. Selander (ERICSSON)

Revision History

Version	Date	Name	Partner	Section Affected Comments
0.1	11/10/2021	Initial ToC	CNR	All
0.2	01/11/2021	Added Section 2 describing the role of analytics in the SIFIS-Home framework	CNR	All
0.3	02/11/2021	Added Sections 3.1.1 and 3.1.3	CNR	All
0.4	03/11/2021	Added description of the Policy Enforcement Task in Section 3.3	POL	All
0.5	07/11/2021	Added section 3.2.1	CEN	All
0.6	12/01/2022	Added preliminary results of the Policy Enforcement Task in Section 3.3	POL	All
0.7	07/02/2022	Added description of the requirements of the Policy Enforcement Task in Section 3.3	POL	All
0.8	08/02/2022	Added description of Privacy Aware Speech Recognition Analytics in Section 3.4	CNR	All
0.9	10/02/2022	Added Section 3.2.2	FSEC	All
0.10	11/02/2022	Added Section 3.1.2 about device activity monitoring	RIOTS	All
0.11	14/02/2022	Added Introduction and Conclusion Sections	CNR	All
1.0	18/02/2022	Final changes before internal revision	CNR	All
1.1	22/03/2022	Addressed comments from internal reviewers	CNR, POL, CEN, FSEC, RIOTS	All
1.2	28/03/2022	Final changes	CNR	All

Executive Summary

This document is an outcome of Working Package 4 "Privacy-Aware analytics for Security and Services", and provides a description of the analytics that have been designed and developed in the SIFIS-Home project at M18, which are a subset of the analytics that have been described in deliverable D4.1. The analytics have been designed and developed consistently with the requirements presented in the deliverable D1.2 "Final Architecture Requirements Report", in order to be applicable to the use cases and IoT-based Smart Home environment taken into account in the SIFIS-Home project, but they could be exploited also in other scenarios.
The proposed analytics are grouped under four activity areas, following the WP4 task's structure: i) anomalies and misbehavior detection; ii) network intrusion detection; iii) policy enforcement; and iv) privacy aware speech recognition. In particular we have designed and developed the following analytics: Device Fault Detection, Device Activity Monitoring in Centralized Cloud, and Parental Control for the area “anomalies and misbehavior detection”, Network Intrusion Detection and Anomaly Detection for IoT Devices in a Consumer Home Network for the area “network intrusion detection” and, finally, Policy Enforcement and Privacy Aware Speech Recognition for the two homonymous areas. For each of these analytics, this document gives a detailed description of the aim, of the involved requirements, of the input data and the returned results, of the analytic training and testing, along with some consideration about the privacy of the input dataset and of the results. This deliverable also describes the techniques for preserving privacy of sensitive data that will be exploited to perform privacy preserving data analysis with the aforementioned analytics.

Finally, the results obtained in WP4 are contributing to dissemination activities in WP7 "Dissemination, Standardization and Exploitation", especially with academic publications in international venues.

Table of contents

1 Introduction

This document is the second output from WP4 "Privacy-Aware Analytics for Security and Services", and it reports the first research and development results obtained at M18 concerning the privacy aware analytics which will be used for intrusion detection at several levels of the SIFIS-Home framework, from device level to the cyber-physical (sensors) and user level, among others. In particular, these analytics have been organized into four groups, consistently with the WP4 task structure: i) anomalies and misbehavior detection analytics; ii) network intrusion detection analytics; iii) policy enforcement; and iv) privacy aware speech recognition analytics.

The main aim of WP4 is to support the functionality of the SIFIS-Home framework by providing the aforementioned analytics, which will be used for several specific purposes in the framework, while still having the general aim of increasing the security level of the smart home. For instance, the Parental Control analytics could be exploited by the SIFIS-Home framework to detect whether a child has been watching a Smart TV alone, i.e., without the supervision of an adult. In this case, if the TV show is not suitable for children, the SIFIS-Home framework could decide to shut down the Smart TV until an adult is present in front of it. Another example concerns the Network Intrusion Detection analytics that could be exploited to protect the SIFIS-Home framework itself: when an anomalous network behavior of a device is detected, the SIFIS-Home framework could decide to isolate such device from the rest of the network, thus preventing that device from infecting other devices, damaging the smart home or hurting the people within it.
Moreover, the activities of WP4 also include the development of analytics for converting the policies expressed by the SIFIS-Home users in natural language into policies expressed in machine-readable language, i.e., policies that are enforceable by the SIFIS-Home framework.

In deliverable D4.1 we provided a list and a high-level description of the analytics that will be designed and developed within WP4. At the time of writing (M18), we have designed and developed a subset of them, and this deliverable describes the analytics from such a subset. In particular, for the area "Anomalies and misbehavior detection analytics" we have the following analytics: *Device Fault Detection*, *Device Activity Monitoring in Centralized Cloud*, and *Parental Control*; for the area "Network intrusion detection analytics" we have the following analytics: *Network Intrusion Detection*, and *Anomaly Detection for IoT Devices in a Consumer Home Network*; for the area "Policy enforcement" we have the analytic named *Policy Enforcement*, and for the area "Privacy aware speech recognition analytics" we have the analytics named *Privacy Aware Speech Recognition*.

Since these analytics have not been integrated into the SIFIS-Home framework yet, this deliverable describes their functionality as stand-alone analytics, while their integration into the SIFIS-Home framework will be described in D4.3 "Final Development of Privacy Aware Analytics for Secure Services", which will be delivered at M33.

This document gives a detailed description of the analytics we designed and developed from several points of view. In particular, in the following is described for each analytics: the main aim of the analytic, the requirements that the analytic contributes to satisfy or that the analytics have, the data it needs as input and how they are collected in the SIFIS-Home framework, the internal architecture, how the analytic is trained and tested (including the description of the dataset that has been used), the results provided by the analytic, whether the analytic has some relevant hardware requirement, some consideration about the privacy of the input dataset and of the results, some relevant implementation details, and some preliminary results obtained from the analytics operated on the dataset previously described.

2 Role of Analytics in the SIFIS-Home Framework

The security and privacy support of the SIFIS-Home framework is based on data analysis. As a matter of fact, machine learning, deep learning statistics and rule-based techniques are adopted to analyze the behaviour of users and devices, to detect intrusions at cyber-physical level (e.g., face-sound recognition), at network level, and at system level (Multi-Level Intrusion Detection). Such analysis mechanisms exploit features extracted through continuous monitoring of the full smart-home stack, e.g., of device system calls, network events, sensor data, DHT operations, collected sounds, pictures and videos. The results of such analysis are used by the SIFIS-Home framework to detect dangerous or not desirable situations in the smart home, in order to timely react and take proper countermeasures to prevent, avoid or mitigate the impact of the malfunctioning/attack.

For instance, the Parental Control analytics is aimed at detecting the number of children and adults that are present in a scene, and it is executed on the video streams produced by the smart cameras installed in the smart-home. The results can be exploited by the SIFIS-home framework for preventing children from watching inappropriate content on the smart TV if they are alone, i.e., if no adults are present nearby.

The SIFIS-Home architecture includes a specific component, namely Data Analytics Toolbox, which hosts all the software engines implementing the aforementioned analytics. This component provides the tools to analyze textual, tabular, and multimedia data in order to perform predictions, analyze voice and gesture commands, detect intrusions and misbehaviors, as well as for providing advanced smart services to the smart home users. The Data Analytics Toolbox is part of the Application Toolboxes module, shown in Figure 1. Please refer to Deliverable D1.3 for a detailed description of the Application Toolboxes module and of the rest of the SIFIS-Home architecture.

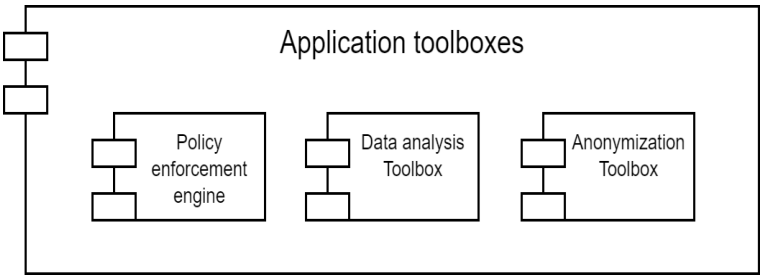

Figure 1: Application Toolboxes module

As we can see in Figure 1, the Application Toolboxes module also includes the Anonymization Toolbox engine, which contains software tools that are used to preserve privacy of data during analysis. Depending on the data type and the desired level of privacy, the most appropriate anonymization tool can be chosen from the toolbox, in order to generalize or suppress data information, as well as for supporting differential privacy for privacy preserving data analysis.

3 Analytics Design

This section gives a detailed description of the analytics that have been designed and developed in Work Package 4 until M18. We recall that these analytics are grouped into four areas, consistently with the tasks of Work Package 4: i) Anomaly and Misbehavior Detection, ii) Network Intrusion Detection, iii) Policy Enforcement, and iv) Privacy Aware Speech Recognition. The description is structured in the same way for all the analytics. First, we describe the main aim of the analytics in the context of the SIFIS-Home scenario and, with reference to deliverable D1.2 "Final Architecture Requirements Report", we list the requirements that the analytics contributes to satisfy, as well as the requirements that the analytics introduces. Then, we describe the data that are given as input to the analytics, how such data are collected from the sensors in the SIFIS-Home framework, how the analytic is trained and tested with such data and the results provided, as well as some privacy consideration on the input data and results. Moreover, we describe the hardware requirements of the analytics, in order to understand on what classes of devices it can be executed. Finally, we present some implementation details of the analytics, along with some preliminary results that have been obtained by executing the analytics on the dataset previously described.

3.1 *Anomaly and Misbehavior Detection*

The analytics of the Anomaly and Misbehavior Detection area have been defined to preserve data correctness and integrity, in addition to preventing anomalous actions and behaviors in smart environments. Data collected from connected devices and network data are processed to provide security as a service, by ensuring multi-level anomaly and misbehavior detection and prevention.

3.1.1 Device Fault Detection

3.1.1.1 Aim of the Analytics

Faulty devices can either affect the whole performance of a system by getting disconnected or by staying connected to the network while not performing as expected and sending wrong/malicious data to intended recipients. The aim of this analytic is to detect devices that are not behaving as expected in order to ensure network reliability, service availability, good performance, and better monitoring of connected devices, as well as service restoring and correction action time minimization in case an anomaly has been detected.

3.1.1.2 Requirements related to the analytics

The Device Fault Detection analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-15: this analytic satisfies the functional requirement F-15 by identifying anomalous behaviours inside the smart home, as it detects anomalous temperature or humidity readings of home devices.
  - F21: once an anomaly has been detected by this analytic, it invokes self-healing algorithms to disconnect devices with anomalous readings.
  - F-41: the analytic provides a presentation of aggregated data and anomalous points from different devices.
- Non-Functional Requirements:
  - PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection. Moreover, the data collection frequency should be scheduled accordingly.

    - DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.
- Security Requirements:
    - SE-17: anomalous behavior is identified in 60 seconds from data collection and analysis time. Since data is collected from sensors based on predefined times and intervals, the identification time depends on these times.
    - SE-21: differential privacy and autoencoder parameters are configurable for data analysis and can be changed.
    - SE-22: this analytic is able to work with anonymized data as well but the accuracy of the results might be decreased.

3.1.1.3 Input Data

Device fault detection uses time series data collected from sensors such as readings of humidity, temperature, noise, vibration, and air flows. Then these analytic processes these data and flags abnormal data deviations based on a specific threshold defined after the training phase leveraging readings during normal device behaviour.

3.1.1.4 Analytic Design

The device fault detection analytic uses an unsupervised learning method since the anomaly that might occur is unknown and not expected. A type of artificial neural network trained on unlabelled dataset, namely Autoencoder [CDM19b], is used in this context. Autoencoders are used for compression purposes and dimensionality reduction through latent representation, and it is one of the most popular anomaly detection methods. The autoencoder network has two parts: an encoder that converts the original dataset into a code referred to as a latent representation, and a decoder which reconstructs the data with the same original dimensions from the latent representation.

Autoencoders are required to be trained on a set of normal data to work appropriately. In particular, they are trained to encode and reconstruct data representing a normal behaviour with an error below a given threshold rate. Thus, once the autoencoder receives anomalous sensor data, the quality of the data reconstructed by the decoder decreases, and the error rate increases above the threshold rate as shown in Figure 2.

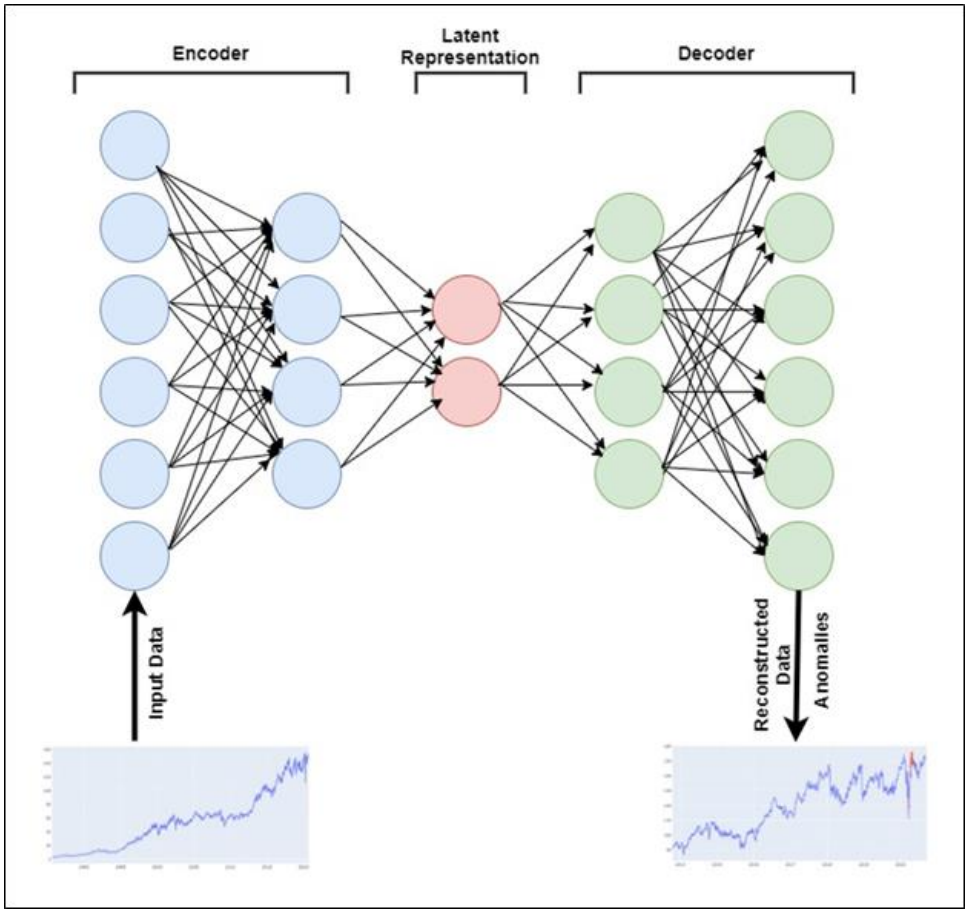

Figure 2: Time Series Data Anomaly Detection using Autoencoders

The analytic receives a set of sequential data with timestamps and the encoder is trained to learn how to produce data and latent representations in both directions: forward and backward. When a data sequence is obtained, the analytic predicts whether it is normal or anomalous by applying the principle of Reconstruction Anomaly Detection; since the used autoencoder has been trained on a set of normal sequences and the reconstruction part works appropriately with an accepted error range below the normal set threshold, once new sequences with different behavior or anomalous nature are fed to the autoencoder, the reconstruction error rate increases beyond the normal error threshold resulting in a high anomaly score.

3.1.1.5 Training and Testing

Available Dataset

The used dataset is Intel Berkeley Research Lab Sensor (IBRL) Dataset [ILSD]. It contains information about data collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004, as shown in Figure 3. This dataset includes a log of about 2.3 million readings collected from the sensors related to temperature, humidity, voltage, and light. Only the data related to temperature and humidity are used in the performed analytics.

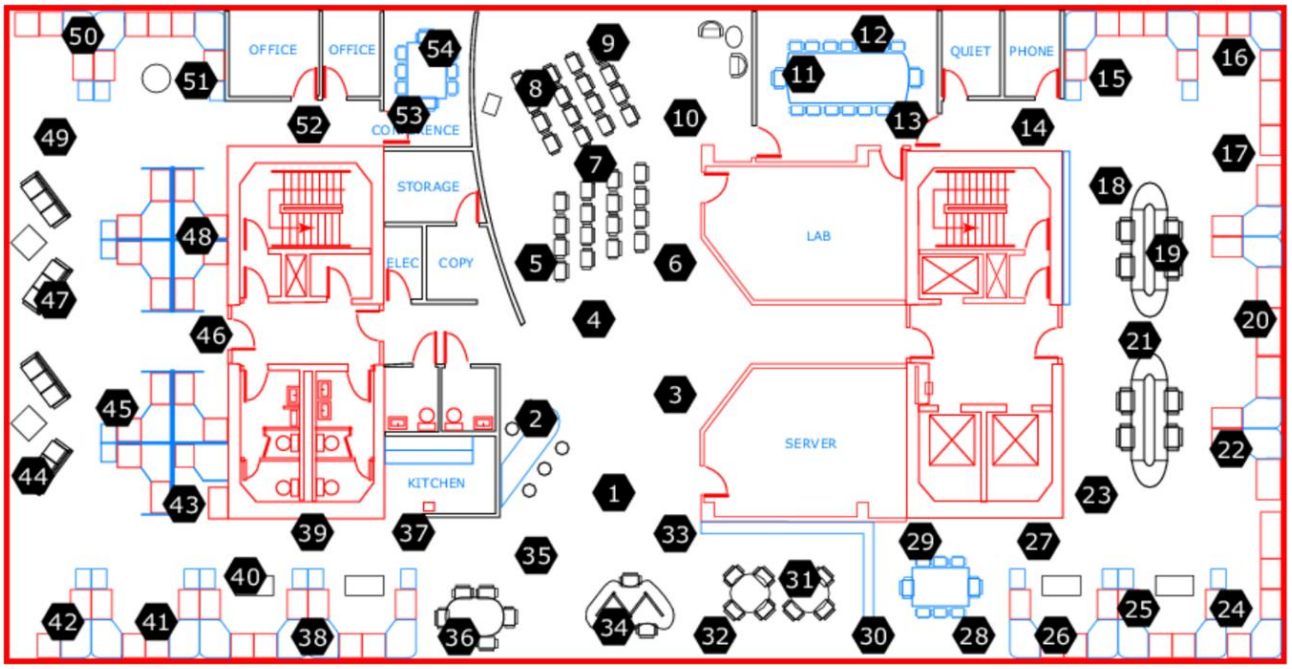

Figure 3: Sensors deployed in the Intel Berkeley Research Lab Sensor (IBRL) [ILSD]

Training strategy

The training is performed using sensor data collected in normal circumstances from up and running devices. Sensor sequences used during the training phase are not labelled and do not have any anomalies, thus minimizing the reconstruction error. Pre-trained models cannot be used in this analytic, since they will not be able to detect anomalies in a dataset that they are not trained on, in addition to the uniqueness of devices' readings that differ among environments and circumstances. The testing is performed on a different dataset from the same devices that might include anomalies resulting in higher reconstruction errors. IBRL Dataset is divided into two parts: the first part consists of the data collected between February 29th and March 9th, that have been used for for training, with aggregated normal temperature ranges between 16 degrees and 30 degrees and aggregated normal humidity ranges between 20 and 50. The second part consists of the data collected between March 17th and March 29th, that have been used for testing. For each sensor, the data are collected every 30 minutes, thus resulting in 48 measurements per day. The sensors with less than 48 measurements are discarded from the dataset since data should be collected periodically in order to be analyzed by the autoencoders model. In the training phase, for each sensor, data sequences are passed to the encoder part of autoencoders to create the latent representation, then the data are reconstructed by using the decoder part and the error ratio is computed. The maximum error ratio obtained in the training phase is saved as the reconstruction error threshold. During the testing phase, any data instance with reconstruction error above the previously computed reconstruction error threshold is marked as an anomalous point.

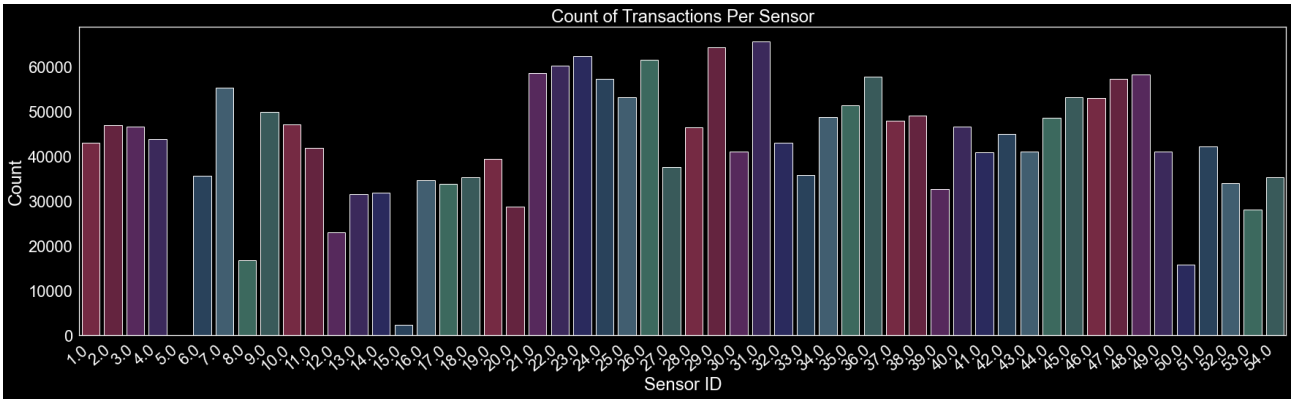

Figure 4: Transactions Per Sensor

3.1.1.6 Output data

The output of this analytic is a score curve of data reconstruction error related to the test data points, in addition to a matrix of anomalous data points corresponding to abnormal values of humidity and temperature. A detailed description of the results is listed in the following:

Sensor ID: the ID of the sensor with the data being analyzed.

Matrix of anomalous data points: a list of the data instances from the original dataset with abnormal values of temperature or humidity. These data points consist of a time stamp and the value of temperature or humidity.

Chart of reconstruction error values: the values in this chart are related to the reconstruction error values generated during the training phase like in Figure 5

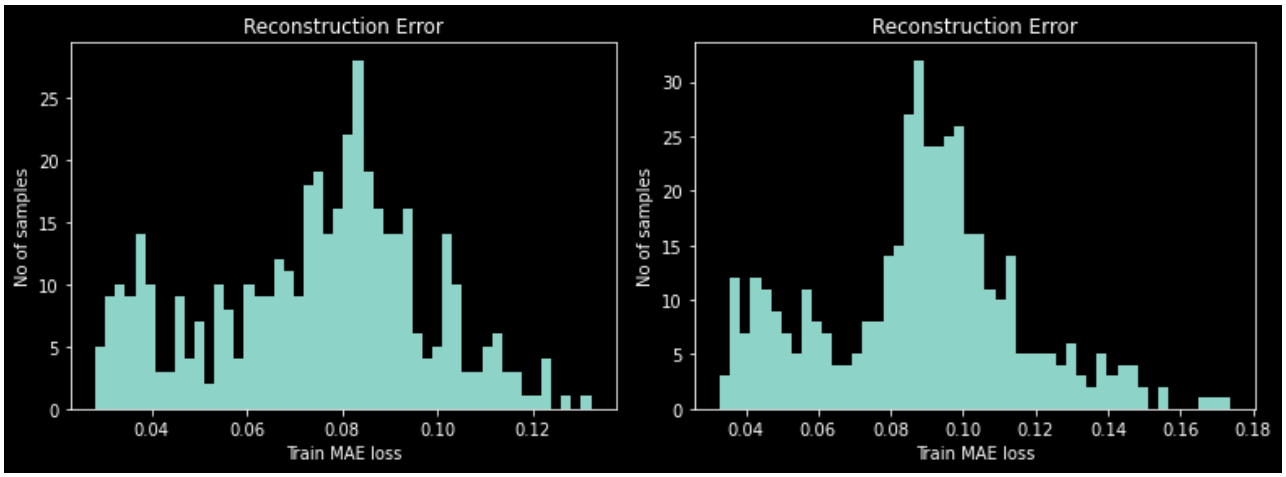

Figure 5: Sample reconstruction error of humidity and temperature respectively

Chart of all data points with anomalous points marked with red colour: These charts include the anomalous data points marked in red associated with the normal values in blue colour.

Such results can be used in the smart home context to detect abnormal device behavior. In such a way,

the SIFIS-Home framework can adapt the context and react based on the current conditions, by changing the behavior of the controlled device (through a modification of the connection) and the surrounding devices.

3.1.1.7 Hardware Requirements

To run this analytic, a minimum of 8GB RAM, 16GB RAM and above, and a minimum of 7th generation (Intel Core i7 processor) CPU are recommended.

Performance is analyzed based on the applied model performance and the time overhead to perform the analytic function by each component as below:

- Anomaly Detection Performance: the time required to classify collected data into normal or anomaly classes.
- Network Time (in case data needs to be shared with a server to be processed): time required to share data with the server in case data processing is done at server side.
- Policy evaluation time: time when an attribute is changed.

Accuracy measures the precision of the anomaly detection component in terms of how many anomalies were detected correctly out of the total number of anomalies.

3.1.1.8 Privacy Considerations

Data collected from environmental sensors, such as humidity, temperature, noise, vibration, and air flows, could raise privacy issues because they could be used to predict the presence and some actions of room occupants. [MMR17]. Hence, to preserve the privacy of such data, their analysis is performed locally, by also taking into consideration the following aspects:

- Input data
  Autoencoders are used for anomaly detection [ZP17], and they are also used for data reconstruction. At the same time, the real data are kept private and only the reconstructed data are shared. Moreover, differential privacy is used with autoencoders during the analysis phase to minimize data memorization by the analytics model and protect individual data instances privacy [Dwo08].
- Results
  Shared results are reconstructed data items, which reveal the original dataset's behavior but not the real dataset instances.

3.1.1.9 Execution of the Analytics by the Analytics Toolbox

This analytic is invoked through physical analysis and multi-level analysis APIs to analyze time-series collected data. The data are passed to be pre-processed for feature extraction and format adaption, then privacy-preserving techniques are applied to the dataset to be forwarded to the analytics engine, which performs data processing using machine learning and deep learning toolboxes. Afterwards, the analysis results are interpreted, aggregated, and presented to the user.

3.1.1.10 Implementation Details

This analytic is implemented using the TensorFlow, Pandas, Numpy, and Keras libraries, by building an autoencoder network and training it on data with normal patterns, and then computing the reconstruction error of these data by using a loss function such as the mean squared error function. After

that, the same autoencoder network is used to reconstruct a new dataset that might contain anomalous entries. If the new dataset contains data entries with abnormal patterns, the autoencoder fails to reconstruct these data points, thus generating a higher reconstruction error since it has not been trained on such data before. Then, the data entries with high reconstruction error are labelled as anomalies. The analytic results are presented using Matplotlib and Seaborn libraries.

Data are collected from the sensors every 30 minutes, hence producing 48 sequences per day. These sequences are fed to the model to be trained in normal circumstances. Then, the trained model is used to analyze other data to check whether anomalies are present or not. The number of sequences can be either increased or decreased and the model training can be performed on a larger or smaller number of sequences based on the user preferences.

3.1.1.11 Preliminary Results

Table 1 and Table 2 show the results obtained for anomaly detection on the IBRL testing set of 11 Sensors having the complete data sequence. The network is trained with the procedure described in Section 4.1.1.3 without anomalies. The results show how autoencoders can distinguish the two classes of the testing set for both temperature and humidity data with a high level of accuracy. The tables provide the results obtained when analyzing time series of temperature and humidity data, respectively, of IBRL dataset performed by autoencoders in terms of True Positives, True Negatives, False Positives, False Negatives, and Accuracy.

Table 1: Results for anomaly detection on IBRL testing set

Temperature							
Sensor ID	Degree Range (Training)	# Actual Anomalies (Testing)	TP	TN	FP	FN	Accuracy
1	17.257 - 28.556	382	375	463	0	7	99.17%
3	17.625 - 28.098	328	310	517	0	18	97.87%
4	18.09 - 27.567	452	437	361	0	15	98.15%
11	15.9 - 27.174	425	407	420	0	18	97.87%
26	14.997- 32.863	391	386	454	0	5	99.41%
28	15.117 - 33.748	279	276	262	0	3	99.45%
30	15.752 - 33.117	383	379	362	0	4	99.46%
32	16.242 - 33.465	317	303	443	0	14	98.16%
33	16.964 - 29.161	398	385	447	0	13	98.46%
34	16.163 - 31.449	430	429	415	0	1	99.88%
35	17.227 - 30.704	362	348	251	0	14	97.72%

Table 2: Results for anomaly detection on IBRL testing set

Humidity							
Sensor ID	Humidity Range (Training)	# Actual Anomalies (Testing)	TP	TN	FP	FN	Accuracy
1	22.737 - 47.142	293	245	549	3	48	93.96%
3	23.536 - 46.547	313	200	532	0	113	86.63%
4	23.923 - 47.961	368	363	445	5	0	99.38%
11	24.754 - 50.186	337	311	508	0	26	96.92%
26	19.327 - 53.779	348	332	443	54	16	91.72%
28	17.71 - 54.134	186	115	654	1	71	91.44%
30	18.922 - 52.429	312	279	527	6	33	95.38%
32	18.242 - 56.086	236	208	524	0	28	96.32%
33	21.849 - 47.477	345	283	500	0	62	92.66%
34	25.2 - 52.859	331	295	514	0	36	95.74%
35	26.516 - 49.797	280	278	263	70	2	88.25%

3.1.2 Device Activity Monitoring in Centralized Cloud

3.1.2.1 Aim of the Analytics

The activity monitoring analytics aims to detect anomalies in IoT sensor behavior. In the scope of this analytics there are IoT sensors that are connected to the centralized cloud and there is no edge processing capability available in the building.

This kind of implementation model produces a continuous data stream addressed to a centralized cloud. The analytics implementation performs processing of the data stream and detects anomalies in real-time. The aim of the analytics is to detect if a connected device is behaving normally based on the outcome of activity monitoring. Additionally, for the considered kind of data flow, high accuracy of the analytics outcome is required, with negligible chances of producing false positives.

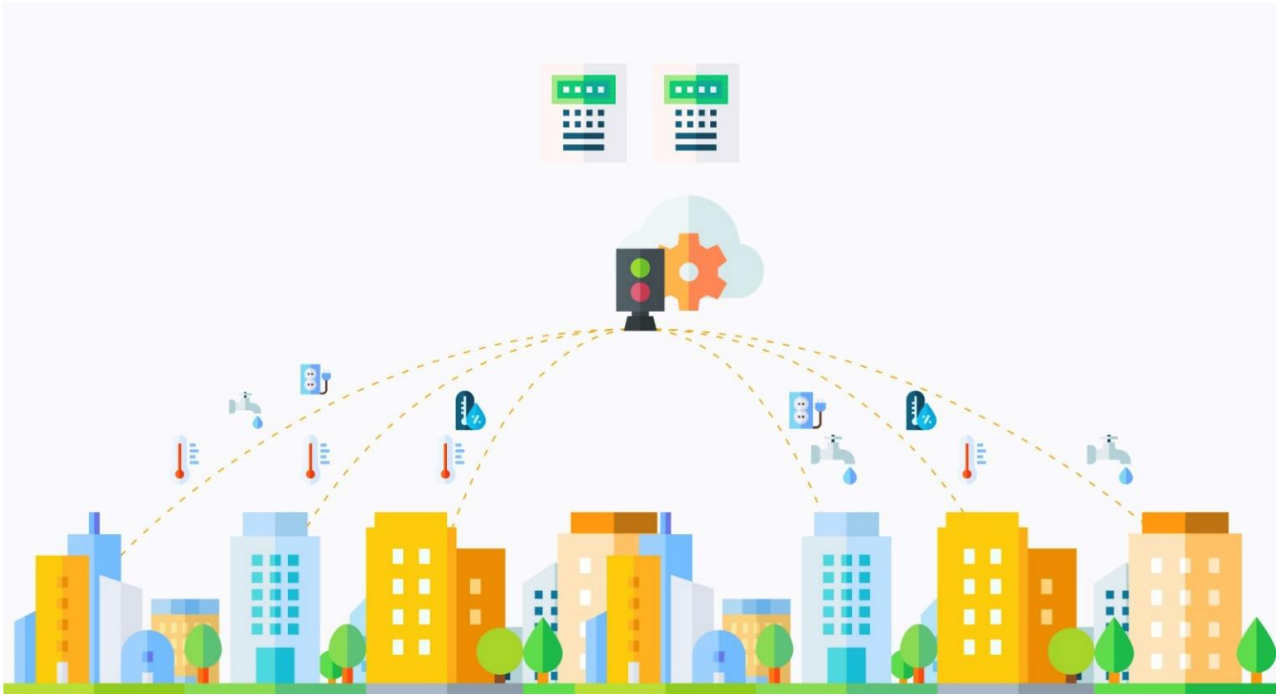

Figure 6: High-level picture of implementation

3.1.2.2 Requirements related to the analytics

The Activity Monitoring Analytics is related to the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-19: The SIFIS-Home system shall detect, identify and disconnect infected devices.
  - F-22; The SIFIS-Home system should allow means of verifying that the malware has not spread to other devices.
- Non-Functional Requirements:
  - PE-10: An abnormal (suspicious) behavior caused by a malware shall be identified and notified within 60 seconds
  - PE-28: The used solutions for communication and system security shall be as much as possible lightweight to enforce in terms of performance, and especially feasible also for resource-constrained devices.
  - PE-29: The performance impact due to communication and system security shall not result in unacceptable impact on the user experience.
  - AV-01: The SIFIS-Home system services and devices shall be available 99% of the time
  - AV-03: Support should be ensured for devices to dynamically react to (D)DoS attacks, by gradually adapting their availability. This includes relying on communication intermediaries for traffic offloading during intense (D)DoS attacks.
- Security Requirements:
  - SE-09 The information about the registered devices, their characteristics and their configurations should be stored in a protected database.
  - SE-13 Data confidentiality shall be ensured all the time.
  - SE-14 The system should not be affected by MITM attacks.
  - SE-17 Anomalous device behaviours should be identified and signalled in less than 60 seconds.
  - SE-22 Analytics shall be able to work with anonymized data when possible.
  - SE-29 Devices must have unique identifiers.

- SE-30 Unless thoroughly assessed and acceptable for the specific application, communications in the networked environment shall be secured, by ensuring confidentiality/integrity/authenticity of messages, as well as protecting from replay protection.
- SE-31 It shall be possible and feasible to provide devices with the necessary key material to establish their security associations and to communicate securely, with preference for automatic procedures.

3.1.2.3 Input Data

The activity monitoring analytics implementation uses continuous data stream received from IoT sensors. The data stream consists of network packets received from the devices. The network packets are processed, and relevant information is parsed from them.

Privacy considerations are important to consider for the analytics implementation, which uses only data that are not privacy sensitive.

As an example of real-world scenario, Riots Global as a relatively small company has a collection of 13000 devices connected to the cloud. The devices practically provide 90 000 things (such as sensors of temperature and humidity, air flow and fan control) and leverage those for collecting data and send them to a cloud storage. Data flow from these devices occasionally causes high peaks in terms of bandwidth utilization. With this number of devices, the data flow is something between 1000 - 3500 data packets per minute, thus not only accurate analytics outcome but also keeping high throughput with given hardware resources is a crucial measure.

3.1.2.4 Analytic Design

Various Machine Learning libraries were considered during the design phase but performance was always an issue and therefore a pure mathematical solution was selected to solve the problem.

The activity monitoring analytics has been enforced through a software library implemented with Python, utilising NumPy library. The goal is to process data flows in real time, thus preserving good performance is the most critical requirement of the implementation. The enforcement of this analytics is based on statistical computing and it aims to automatically detect when something deviates from the current common baseline. The implementation uses metadata related to each IoT device and the actual data collected from the sensors. The implementation monitors the data flow rate and detects data frequency variability between data blobs.

The implementation uses the Interquartile Range (IQR) method to detect anomalies. The Interquartile Range is a measure of variability, based on dividing a data set into quartiles and finding the outliers based on dynamic data processing. In statistics, an outlier is a data point that differs significantly from other observations. A common usage of IQR method is the representation of probability distribution, and a commonly used rule says that a data point is an outlier if it is more than 1.5 * IQR above Q3 or below Q1.

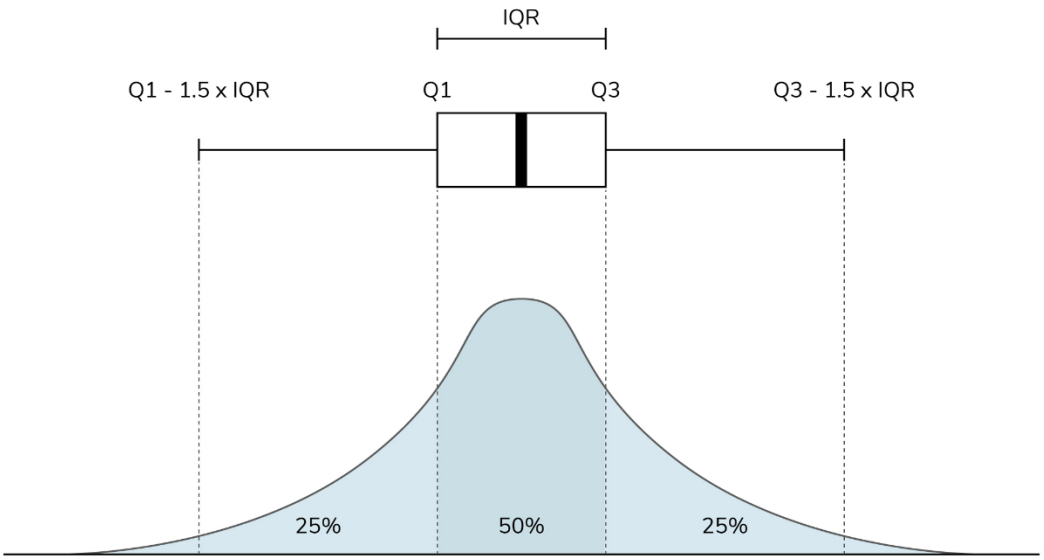

Figure 7: Interquartile Range (IQR) method

3.1.2.5 Training and Testing

The implementation is utilising a passive learning method and is utilizing statistical computing methods. The implementation does not require specific training and it will calculate the limit values for each device by itself.

The implementation admits two configurable values, i.e., time interval and buffer size, and convenient values for both have been found for testing. The time interval depends on the system - too short time interval produces binary results and too long interval make the system react slowly.

The best buffer size depends on the use case. A big buffer size uses more resources, while a too small one yields inaccurate analytics outcomes.

Analytics testing was performed with customized IoT devices that provide valid data (system messages, temperature, humidity, fan speed, on/off data, LED light control, electricity meter), but with higher sample and data transfer rate compared to the normal. Additionally, some of the devices are equipped with touch panel and have a special functionality that can be activated on request for demonstrating the misbehaviour scenarios.

Figure 8: IoT devices used in the analytics training and testing

3.1.2.6 Output data

The analytics implementation is a software library, which relies on an array of data representing all the devices. The status of each device considers packet transmission interval, byte rate for outgoing traffic and calculated IQR values.

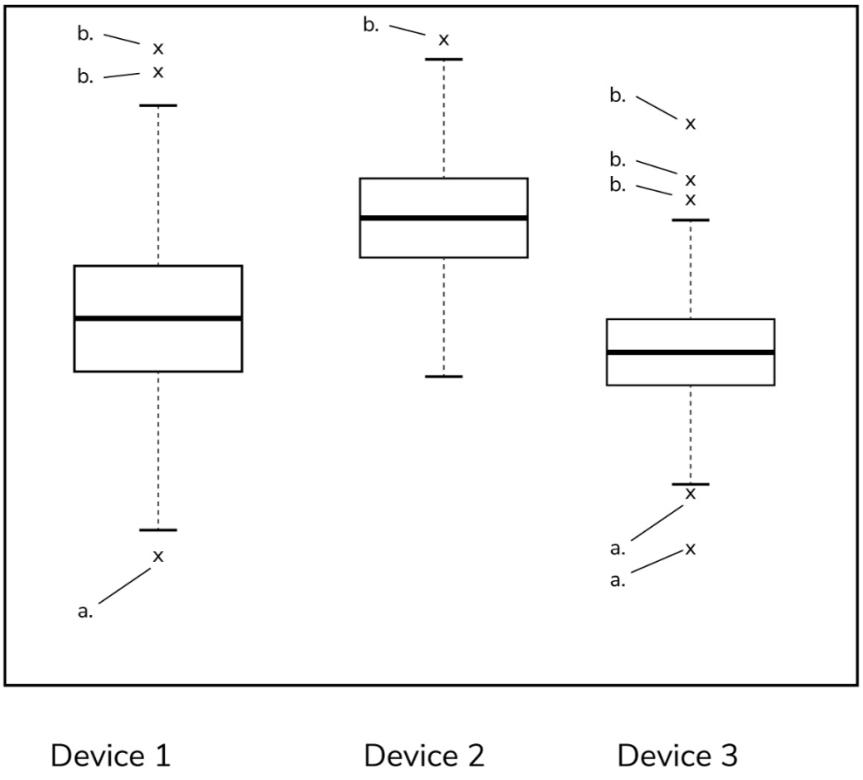

Figure 9: Visual representation of output data points

The implementation uses the IQR method to find the statistical limits for each device and it tells in real-

time if there are anomalies detected in the data flow. The result is based on device activities. The library keeps the array up to date for each device, indicating whether the device is inactive, behaving normally or overactive. Additionally, the library reports the possible outlier difference compared to the limit of 1.5 IQR.

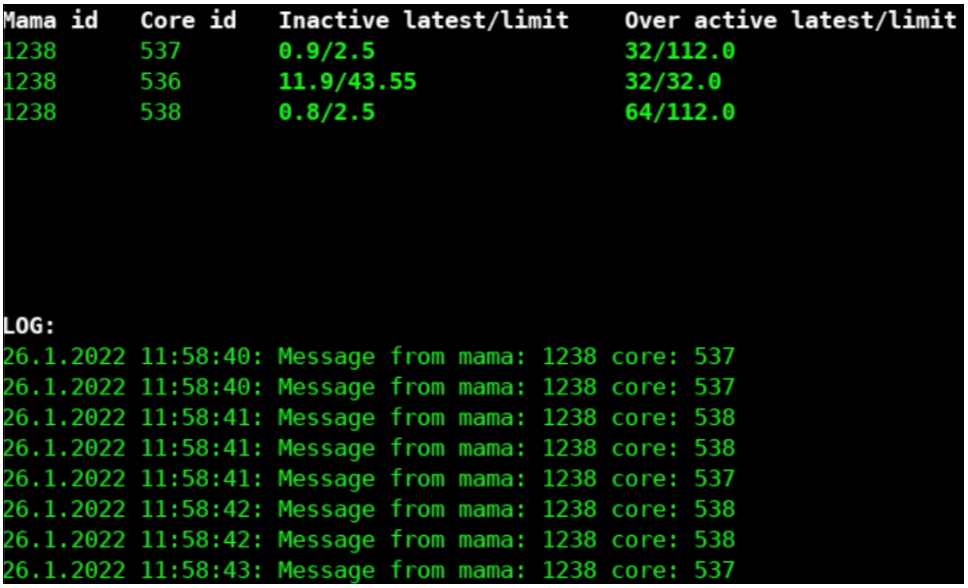

Figure 10: Terminal screenshot of the running analytics

3.1.2.7 Hardware Requirements

The library was written in Python using Python librariesand ran in a Linux environment. The actual hardware requirements are not tight. An example of hardware configuration capable to run the analytics is as follows:

- CPU: 4 Cores
- RAM: 8GB

As part of the analytics demo, a dedicated server has been set up, together with a couple of IoT sensors with customized software to simulate various kinds of anomalies.

3.1.2.8 Privacy Considerations

The data is collected from home environment and sent to the external third-party cloud, where it is processed.

The software library performs analytics for the data collected by sensors. Most of the privacy considerations are part of the actual cloud service using this library.

The library needs unique identifier for each sensor. Sensor data that is collected may be combined with data from other sources to get privacy sensitive data and thus makes it important to consider privacy topics when using this library.

3.1.2.9 Execution of the Analytics by the Analytics Toolbox

The implementation is a software library to be integrated to part of wider system. It collects data regarding statistics and alerts as environment variables and makes them available to be handled by the higher layers.

3.1.2.10 Implementation Details

The implementation works as follows:

1. Data from a device are received at the cloud server
   - The packet sender is identified from the data
   - The server provides the packet metadata (sender, length and timestamp) to the dataflow analytics service
2. The Dataflow Analytics Service calculates:
   - The time since the previous packet was received (packet interval)
   - The amount of data received at the given, configurable, time interval (byterate).
3. The Dataflow Analytics Service stores the resulting values to a buffer
4. When the buffer has enough data related to the device, the Dataflow Analytics Service calculates the IQR for the observed packet interval and byterate
   - If the packet interval is over 1.5 IQR, then the device is too inactive
   - If the byterate is over 1.5 IQR, then the device is too active
5. The Dataflow Analytics Service keeps updating the IQR values upon receiving a new packet and recalculates alarm limits

3.1.2.11 Preliminary Results

An IoT test network with several sensors was set up to test the analytics implementation. The sensors were connected to the cloud server running the Activity Monitoring Analytics software library. At first, the sensors generated normal traffic for some time, in order to let the system determine alarm limits for each device. After that, the simulated anomalies were triggered for each devices.

An internal test case for the system is that it can be used to monitor a set of 100.000 devices that produce 1000 request per second, so that RAM and CPU requirements are in the limit of economically feasible server configuration with minimum performance impact.

Preliminary tests indicate that all the changes in the monitored devices were detected by the analytics implementation. We also evaluated the performance impact and noted that the implementation is very efficient.

3.1.3 Parental Control

3.1.3.1 Aim of the Analytics

The parental control analytic aims to recognize children acting in a surveilled environment, in order to restrict the actions they can perform, and to monitor their behavior for ensuring that they do not experience any dangers.

3.1.3.2 Requirements related to the analytics

The Device Fault Detection analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
    - F-01: The analytic with a biometric analysis manages to identify the ages of the user in front of the camera.
- Non-Functional Requirements:
    - PE-02: The recognition of the ages of the person happens in less than 5 seconds;
    - US-26: The analytic needs the use of GPU in order to perform its actions on time;
- Security Requirements:
    - SE-02: The API used for interacting with this analytics are protected with a secure communication protocol.

3.1.3.3 Input Data

The parental control analytic takes as input data the video frames produced by the surveillance cameras distributed into the controlled environment, such as the camera of the controlled device or the surveillance cameras deployed in the environment, and the metadata provided by the controlled devices. The combination of multiple cameras allows to create a global view of the monitored environment, thus improving detection accuracy. The metadata about devices allows to understand the policy associated to such devices, which could express conditions to be met in order to be allowed to use the device, e.g., the function *f* of the device can be performed only by an adult user.

3.1.3.4 Analytic Design

The SIFIS-Home parental control analytic is composed by a pipeline of two components described in the following:

- **Face Extraction Layer (FEL)**: It is used to analyze each video frame to detect human faces in front of the controlled smart device. Such a component is designed to run in real time and perform face detection by using a lightweight Deep Learning algorithm. The detection result is a list of bounding boxes representing the location of the detected face in the analyzed frame. An example is given in Figure 11.

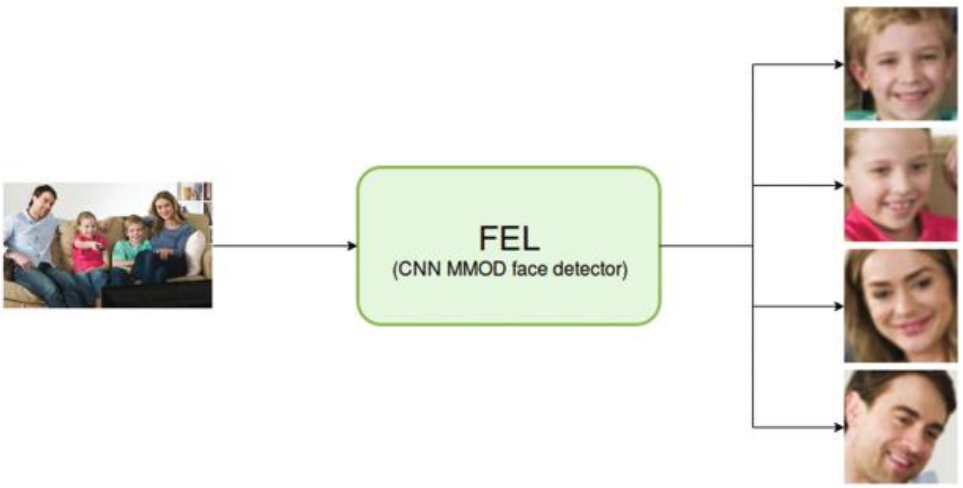

Figure 11: Face Extraction Level

- **Age Classification Layer (ACL)**: Such a component takes in input a single face extracted in the previous layer and analyzes it to estimate the age of the detected person. It is based on a Deep Learning algorithm that take as input a single human face image and maps it into one of the following age ranges [0–2], [4–6], [8–12], [15–20], [25–32], [38–43], [48–53], [60–100].

Figure 12 shows the entire pipeline in an example scenario, where the controlled device is a smart-TV. In this case, the FEL takes as input the frames extracted from the smart-TV camera, performs the analysis for detecting the human faces frame by frame, and forwards the results to the ACL, which estimates the person's age for each face.

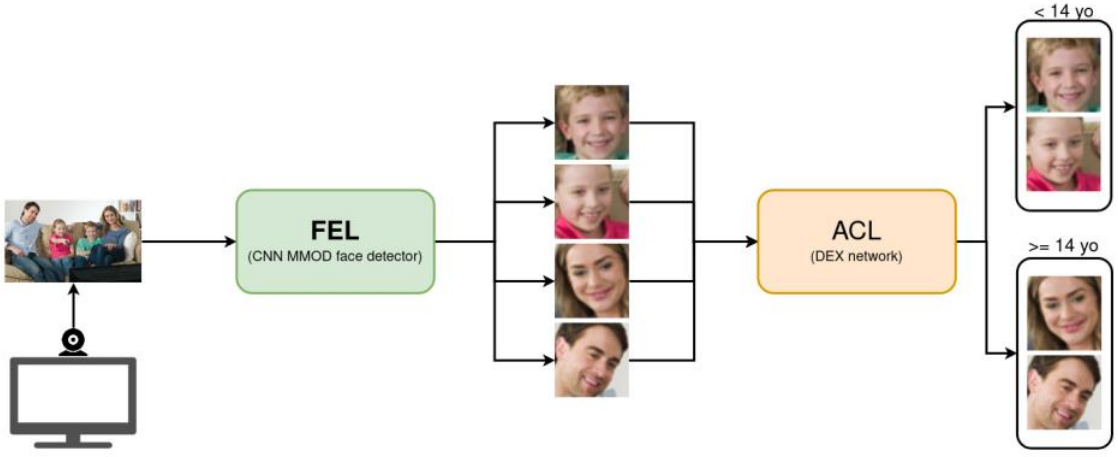

Figure 12: Pipeline of the parental control policy enforcement involving the parental control analytics

3.1.3.5 Training and Testing

Available datasets

Labelled Faces in the Wild
It is a database of face photographs designed for studying the problem of unconstrained face recognition [HRB08]. The data set contains more than 13,000 images of faces collected from the web. Each face has been labelled with the name of the pictured person. 1,680 of the pictured persons have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector.

ImageNet dataset
ImageNet is an image dataset organized according to the WordNet hierarchy. Each concept in WordNet, possibly described by multiple words or word phrases, is called a synonym set or synset. ImageNet populates 21,841 synsets of WordNet with an average of 650 manually verified and full resolution images. As a result, ImageNet contains 14,197,122 annotated images organized by the semantic hierarchy of WordNet. ImageNet is larger in scale and diversity than the other image classification datasets [RDS15].

IMdb-WIKI dataset
IMDB-WIKI dataset is the largest public dataset of face images with age and gender labels. It is gathered using the most popular 100,000 celebrity images from IMDb and Wikipedia websites [RTV08]. In total the dataset is composed of 460,723 face images from 20,284 celebrities from IMD and 62,328 from Wikipedia, thus 523,051 in total. Such a dataset is unbalanced with missing data in the range [0-15] and [70-100] ages (see Figure 13).

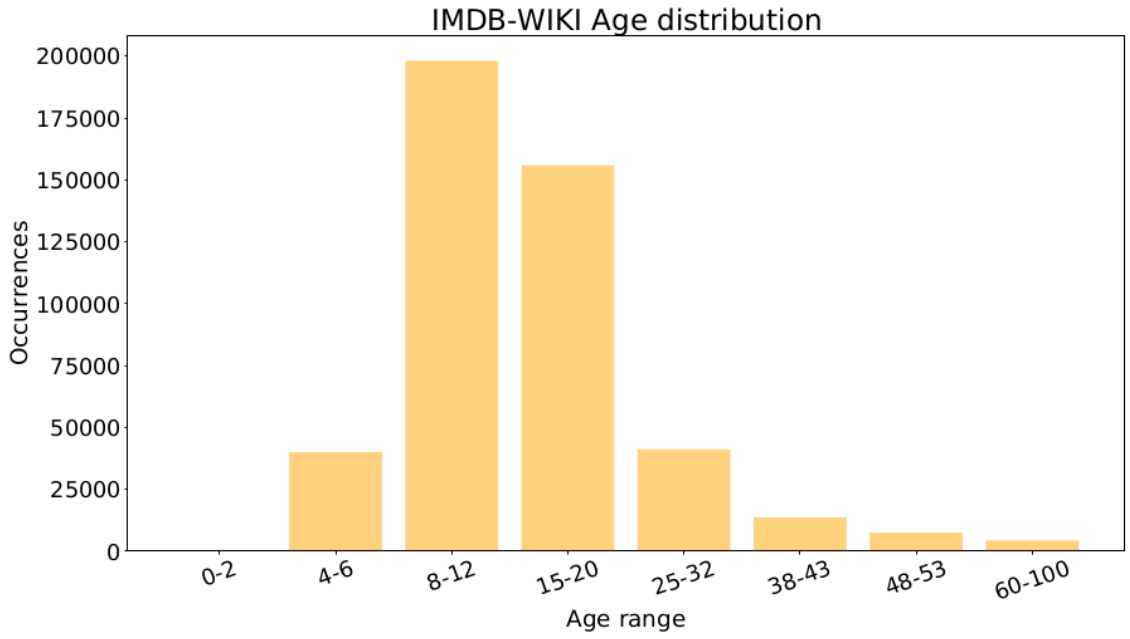

Figure 13: Distribution of ages in the IMDB-WIKI dataset

Adience DB dataset
As a balanced dataset, we exploited the Adience DB dataset, which contains 26,580 face images from

2,284 subjects divided in 8 different age ranges, equally distributed as shown in Figure 9. The dataset has been filtered, by removing images with multiple faces belonging to different age ranges or images wrongly labeled, thus obtaining a dataset consisting of 5,949 faces in the age range [0–14] and 5,937 faces in the age range [15–100] (see Figure 14).

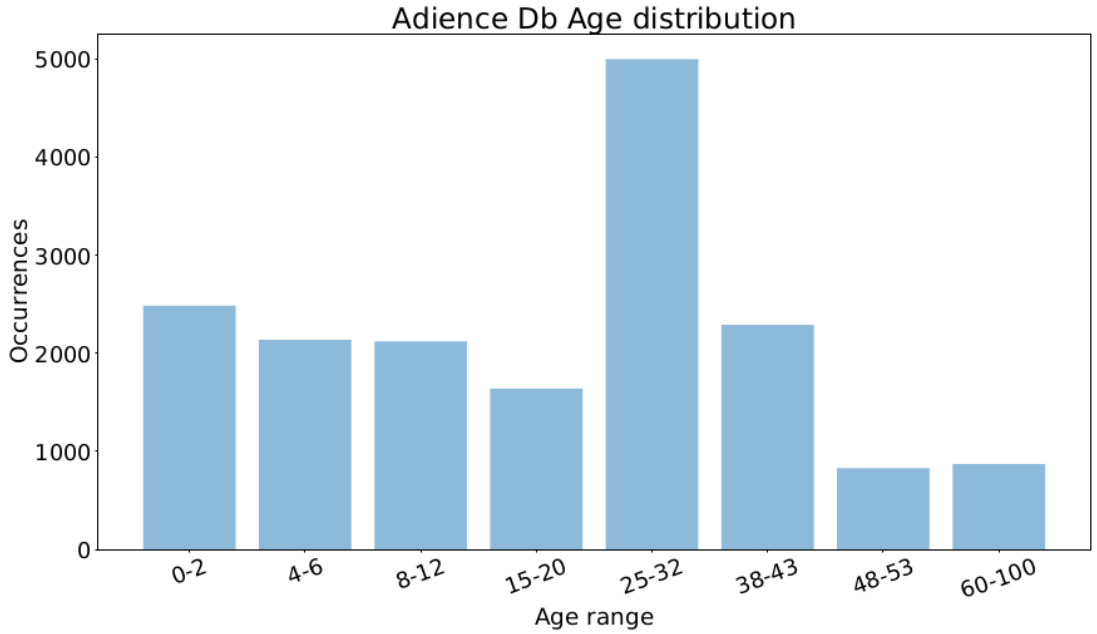

Figure 14: Distribution of ages in the Audience DB dataset

Training strategy

FEL training

The FEL is designed to use an already trained network on Labeled Faces in the Wild dataset, namely Dlib, and no training phase is necessary. Thus, the FEL can be used as a pre-trained network.

ACL training

The deep learning ACL network we used is pre-trained on the ImageNet dataset on 1000 categories. Such pre-training is aimed at learning the high-level object features and should be used as a starting point for age estimation. Since the first layers of the network (convolutional layers) learn low-level image features and the last fully connected layers evaluate the features to determine to which class they belong, the main idea is to start with some knowledge about object detection (ImageNet dataset), and then retrain the last fully connected layer using the specific dataset suitable for estimating ages and detect faces. As transfer learning dataset, we exploited the IMDb-WIKI dataset, which contains a large amount of data useful to learn low-level features in order to detect faces in an image frame and estimate ages. By doing so, we adapt the network to accomplish the new tasks (i.e., face detection and age estimation of 101 classes). In order to attain a fine-tuning of the obtained model on a smaller but balanced dataset, Adience DB should be used, to get higher accuracy on the minority classes of the IMDb-WIKI dataset and train the last layer to distinguish “n” age ranges, e.g., [0–14] and [15–100] age ranges. Figure 15 shows the training pipeline.

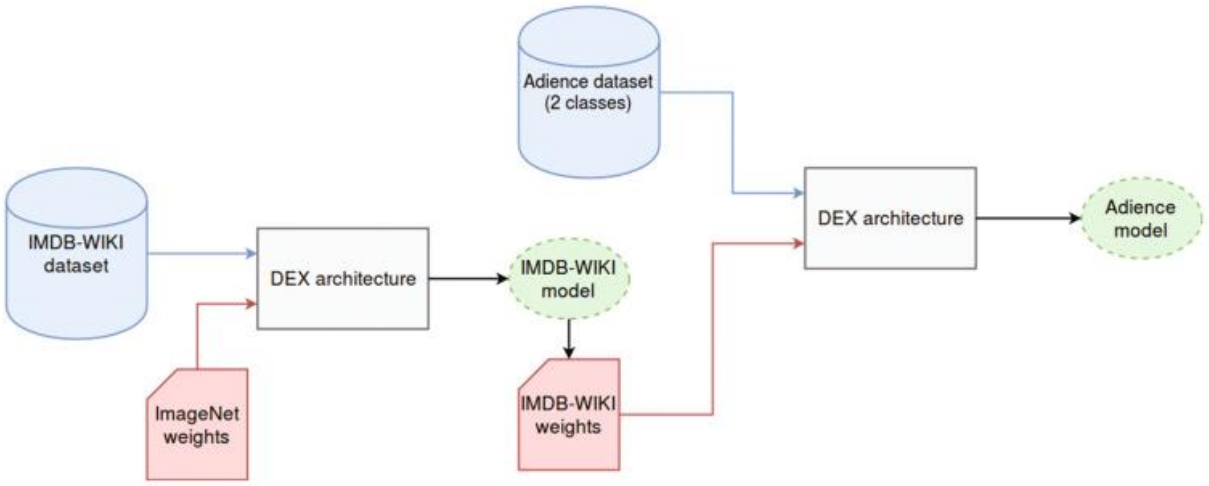

Figure 15: ACL training pipeline

3.1.3.6 Output data

For each analyzed frame, the SIFIS-Home parental control analytic provides information about the human faces that have been detected, including the corresponding age estimation. For each face that is detected, the related record is structured as follows:

- **ID frame:** the identifier of the analyzed frame.
- **ID human face detected:** the identifier of the human face detected in the analyzed frame.
- **Bounding box position:** 4 elements indicating the (x, y) coordinates of the top-left corner, the width and the height of the bounding box surrounding the detected face. Figure 16 shows an example of bounding box coordinates in comparison to the frame size. For instance, the green bounding box in Figure 16 is identified by the following elements:
  - **(x, y) coordinates of the top-left corner**: (101,80)
  - **width**: 171 pixels
  - **height**: 331 pexels

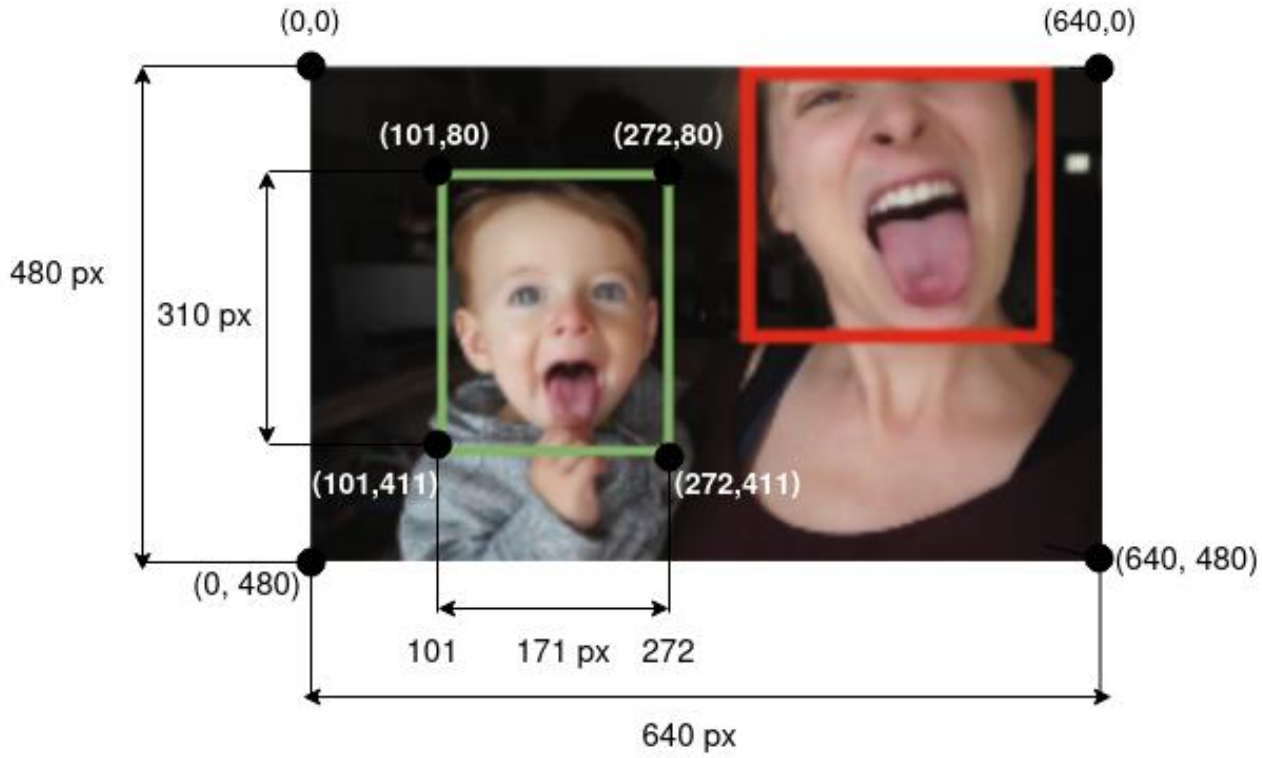

Figure 16: Example of bounding boxes with coordinates in an image

- **Probability human face:** Probability that the human face detected is a face.
- **Age range #i probability:** Probability that the human face detected belong to the age range #i.

Table 3 shows an example of results produced by the analytic.

Table 3: Results produced by the Face Extraction Layer and Age Classification Layer

Face Extraction Layer				Age Classification Layer		
ID frame	**ID human face detected**	**Bounding box position**	**Probability human face**	**Age range #1 probability**	**Age range #2 probability**	**Age range #n probability**
1	1	(101,80,171,310)	98%	85%	7%	9%
1	2	(300,0, 198,300)	99%	1%	97%	2%

Such results can be used by the SIFIS-Home framework to detect the presence of a person in front of a controlled device and to understand whether such person is an adult or a child in order to allow the usage of such device or not. Going back to the example of the smart-TV, exploiting the Parental Control analytics the SIFIS-Home framework can detect whether there is a child alone in front of the smart-TV in order to prevent adult-only videos from being played.

3.1.3.7 Hardware Requirements

The recommended Hardware Requirements for running this analytic are as follows:

- GPU: for better cost and performance, TX 2070 or an RTX 2080 Ti is recommended, with 16-bit models and memory >=11 GB. A blower-style fan for cooling is also recommended.
- RAM: 16 GB RAM or above is recommended, as appropriate in the presence of high-performance GPUs.
- CPU: CPU PCIe lanes and motherboard PCIe lanes must support the desired number of GPUs. A minimum of 2 threads per GPU has to be supported, i.e., usually one core per GPU.

Performance is assessed based on the applied model performance and on the time required to perform the analytic function by each component as below:

- Face Detection Performance: the time needed to extract the faces from the frames.
- Age Estimation Performance: the time required to classify the extracted faces to a specific age range.
- Network Time (in case data needs to be shared with a server to be processed): for surveillance cameras and some smart devices, the data might be shared with a central server to perform the analytics due to HW requirements limitations and performing the analytics on the smart device might cause a considerable overhead and result in poor performance.
- Policy evaluation time when an attribute is changed.

Accuracy measures the accuracy of the face detection component in terms of how many faces were extracted correctly out of the total number of faces, and the accuracy of the age estimation component in terms of how many ages were estimated correctly out of the total number of extracted faces.

3.1.3.8 Privacy Considerations

Input data

In order to preserve the privacy of data, the mechanisms of Autoencoders [ZP17] and Differential privacy [Dwo08] are used (see Section 4). Autoencoders act as a data compressor, it reduces the size of data to be analyzed by keeping only the most important features only, thus improving performance and privacy. Instead, Differential Privacy is used for privacy preservation and minimizing memorization by the learning model. Therefore, the original data are protected, and the results cannot be used to reconstruct the datasets.

Results

The resulting ages of analysed faces do not make it possible to infer information about the faces in the original dataset.

3.1.3.9 Execution of the Analytics by the Analytics Toolbox

In the parental control analytic, the multimedia analysis API is invoked to analyze both recorded videos as well as streaming content with metadata. These data are forwarded to the pre-processing layer to be formatted and to extract features, and are then anonymized by using the anonymization toolbox. Afterwards, the anonymized datasets are processed using the deep learning and machine learning toolboxes and the analysis results are aggregated, interpreted and presented to the user.

3.1.3.10 Implementation Details

Face Extraction Layer: The tool used for face detection is based on dlib's state-of-the-art face recognition built with deep learning. Dlib's face captures 128 data points per face, resulting in unique

parameters for the hash of each face across a variety of different photos. Subsequently, a Support Vector Machine (SVM) is trained on the derived faces via scikit-learn, resulting in an agile face detection model that can run with minimal latency in the right conditions. The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

Age Classification Layer: The ACL is implemented through deep learning convolutional neural networks (CNNs) that use the VGG-16 architecture, [SZ14], pre-trained on ImageNet for image classification. The training is performed following the procedure described in Section 3.1.1.5, training the classifier with two age ranges: [0-14] and [15-100].

3.1.3.11 Preliminary Results

Face Extraction Layer

Table 4 shows the results obtained by the FEL to detect the faces in the AdienceDB testing set. The table shows the total number of faces in each age range, the number of faces correctly detected (True Positive), the number of not detected faces (False Negative), the number of non-faces detected as faces (False Positive) and the final Face Detection Rate computed as the True positive divided by the total number of faces.

Table 4: Performance of the Face Extraction Layer on the AdienceDB testing set

	[0-2]	*[4-6]*	*[8-12]*	*[15-20]*	*[25-32]*	*[38-43]*	*[48-53]*	*[60-100]*	*Total*
# Faces	*219*	*158*	*129*	*86*	*326*	*147*	*59*	*60*	*1184*
True Positive	*215*	*157*	*125*	*80*	*320*	*144*	*54*	*55*	*1151*
False Negative	*4*	*1*	*4*	*6*	*6*	*3*	*5*	*5*	*34*
False Positive	*2*	*3*	*4*	*0*	*5*	*0*	*0*	*3*	*17*
Face Detection Rate	*98.17%*	*99.36%*	*96.89%*	*93.02%*	*98.15%*	*97.95%*	*91.52%*	*91.66%*	*97.21%*

Age Classification Layer

Table 5 shows the results obtained for the age estimation in the AdienceDb testing set. The network is trained with the procedure described in Section 3.1.1.5 with two age ranges [0-14] and [15-100]. The results show how the network is able to distinguish the two training classes in each age ranges of the testing set. The table provides the total number of faces for each age range, the number of faces detected in the right age range (True Positive), the total number of faces classified in the wrong age class (Errors) and the Age Estimation Rate computed as the total number of faces divided by the number of True Positive.

Table 5: Performance of the Age Classification Layer on the AdienceDB testing set

	[0-2]	*[4-6]*	*[8-12]*	*[15-20]*	*[25-32]*	*[38-43]*	*[48-53]*	*[60-100]*	*Total*
# Faces	*219*	*158*	*129*	*86*	*326*	*147*	*59*	*60*	*1184*
# True Positive	*216*	*139*	*98*	*74*	*324*	*146*	*59*	*60*	*1116*
# Errors	*3*	*19*	*29*	*12*	*2*	*1*	*0*	*0*	*66*
Age Estimation Rate	*98%*	*87%*	*75%*	*86%*	*99%*	*100%*	*100%*	*100%*	*93%*

3.2 *Network Intrusion Detection*

The Network Intrusion Detection analytics exploit network flows and sensor data to detect intrusions in the smart-home network. We defined two distinct Network Intrusion Detection analytics, one performed on the edge and the other one in the cloud.

3.2.1 Network Intrusion Detection

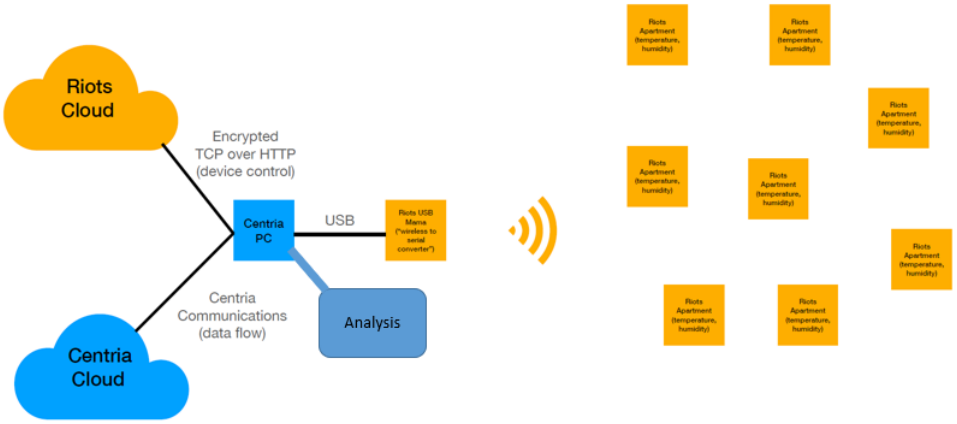

Figure 17: RIOTS test network

First, network packets and sensor values are collected and parsed. Then, statistics are computed and monitored, possibly raising a follow-up alert.

3.2.1.1 Aim of the Analytics

The aim of the analytics is to detect anomalous activities in the network. For example, an exceptionally high ratio of packets conveying a certain flag (state of connection indicator), a sudden increase in the overall traffic within a time window, unusual values from the sensors and a too high packet transmission rate can indicate a malicious or faulty activity which the user should be alerted about.

3.2.1.2 Requirements related to the analytics

The Edge analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:

    - F-20: This system's purpose is to alert users if suspicious network traffic is detected, such as traffic generated by malware.
- Non-Functional Requirements:
    - PE-10: Suspicious network traffic is detected and notified within 60 seconds.
    - PE-11: Alerts are created within 5 seconds after suspicious traffic is detected.
- Security Requirements:
    - SE-22: Anonymized data can be used in this analytic.
    - SE-39: This system identifies denial-of-service attacks based on the activity within the network.

3.2.1.3 Input Data

The input data consists of a network packet exchanged between the connected devices. The packets are captured, and relevant information parsed from them (port numbers, protocol identifiers, timestamps, IP addresses, etc...). Sensor values are collected from the IoT gateway in hex form.

For collecting all the network traffic between devices, the device running the IDS (intrusion detection system) must be connected to a mirror port or a network tap, or it must be a network gateway. If only the host traffic is monitored, then the device could be connected normally to the network.

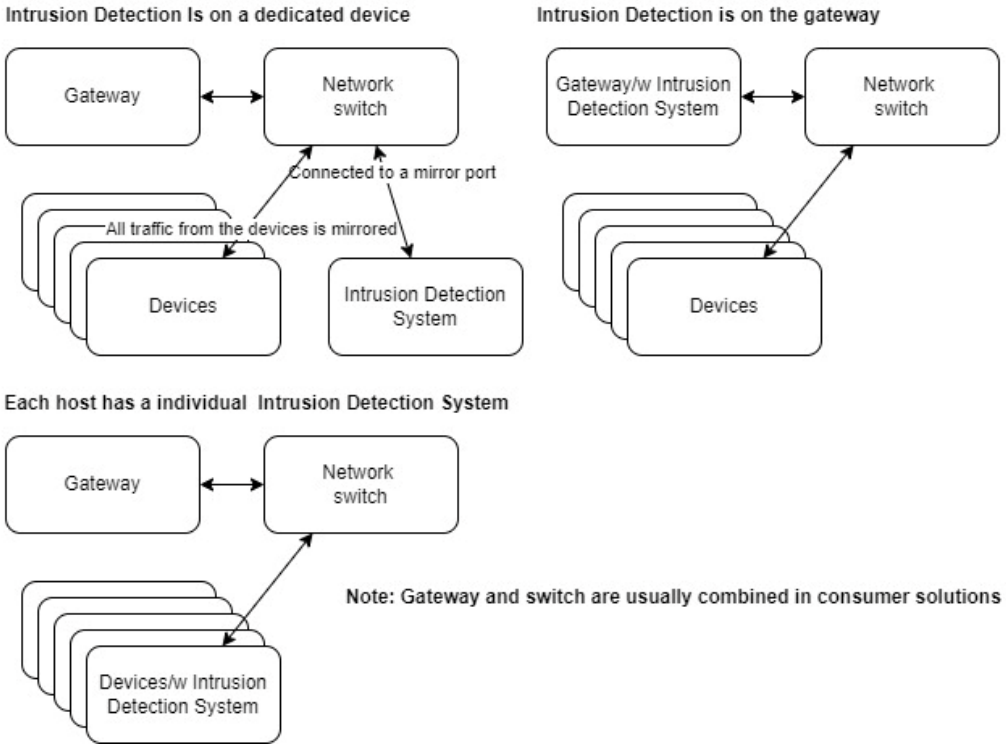

Figure 18: Locations of IDS (intrusion detection system) solutions

Figure 19: Structure of the sensor message

	time	src_addr	src_port	dst_addr	dst_port	length	flags	proto
0	Thu Feb 3 09:42:38 2022	192.168.0.108	8086	192.168.0.107	50000	66	0x010 (ACK)	6
1	Thu Feb 3 09:42:38 2022	192.168.0.107	50000	192.168.0.108	8086	1514	0x010 (ACK)	6
2	Thu Feb 3 09:42:38 2022	192.168.0.107	50000	192.168.0.108	8086	1514	0x010 (ACK)	6
3	Thu Feb 3 09:42:38 2022	192.168.0.107	50000	192.168.0.108	8086	1514	0x010 (ACK)	6

Figure 20: Example of the collected fields from packet capture

3.2.1.4 Analytic Design

The SPOT algorithm (developed by Amossys-team) is a streaming univariate anomaly detector based on Extreme Value Theory (EVT). As a statistical method, its function is to construct a decision threshold zq which is a quantile. It computes zq such that P (X > zq) = q, where X represents the monitored data and q is the user-defined parameter. In practice, q is extremely low (10^{-5}, for example), meaning that observing X higher than zq is very improbable [SPOT17].

Netspot works by monitoring network statistics with the SPOT algorithm, and it increments network counters once a packet is received. For example, looking at flag, size of the packet or protocol. This part of the program is called MINER and it uses the GoPacket library, wrapping around the libpcap library for parsing network packets. The MINER extracts headers when present in the analysed packets, including Ethernet, IP, ICMP, TCP and UDP [NETSPOT21].

Every layer will be used only for specific counters. The MINER dispatches them according to the counter needs. For example, the IP counter can only receive IP layers and it increments every time a packet including such a layer is received. But the ACK counter, which accepts only TCP layers, increments once the received TCP layer has an ACK flag set to 1 [NETSPOT21].

The part called ANALYZER manages the network statistics. The statistics needs the counter values to be computed. For instance, the R_ACK statistic (ratio of ACK packets) divides the ACK counter by the IP counter at certain intervals. For example, every 3 seconds the ANALYZER asks the MINER the counter values and computes the statistics. This statistic corresponds to the ratio of ACK packets during the last 3 seconds. The statistics embed a SPOT instance which monitors what they compute in real time. the ANALYZER has three outputs: Statistics in raw form, the SPOT thresholds, and alerts [NETSPOT21].

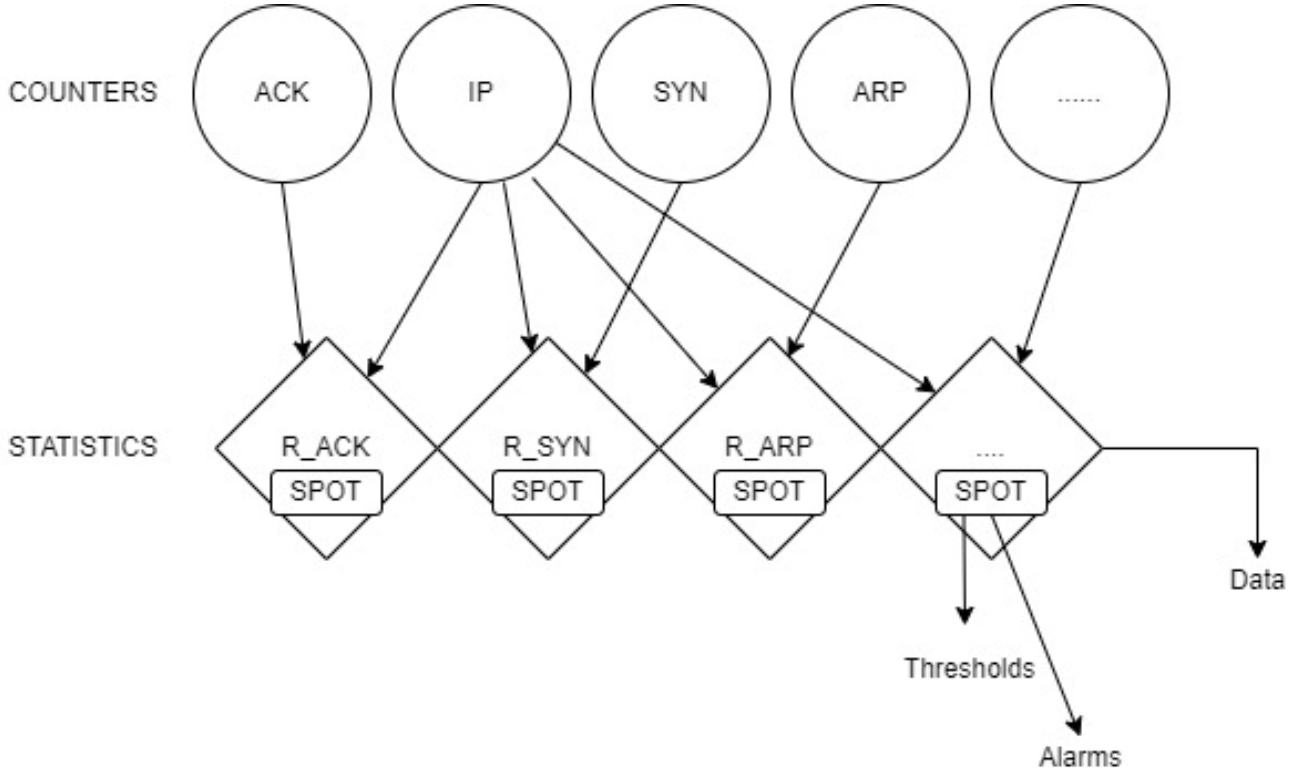

Figure 21: NetSpot components

3.2.1.5 Training and Testing

The SPOT algorithm was tested by utilizing PCAPs (packet capture samples) of malware and Linux tools like Hping3 and Tcpreplay. Tcpreplay is used to modify and replay the malware packet captures and Hping3 is used to generate denial of service attacks.

SPOT algorithm needs the initial batch of observations to form thresholds for the monitored statistics. The thresholds are dynamic, and they adapt to a subtle change in the network by updating them based on chosen number of new observations. However, if the initial thresholds are triggered, the values over or under the threshold are not used to update the thresholds.

For the training and testing purposes, a network consisting of multiple sensors, IoT gateway, database/log server(virtual) and Linux workstation(virtual) was used for these tests. An implementation of SPOT called NetSpot was used to perform the initial tests, as it contains all the components and needed only to be built and configured on the machine which it was intended to be used on. NetSpot was run on a Raspberry Pi4, which was also acting as the IoT gateway for the sensors. NetSpot includes tuning parameters which can be used to tune it to meet specific needs.

For this test, SPOT was initialised by using 2000 observation from the normal traffic. After the thresholds were formed based on the observations, a denial-of-service attack was launched from the Linux workstation. As a results, the statistics that monitor the overall traffic over a certain time window and the ratio of tcp packets with a set synchronize flag were triggered and alarms were raised.

3.2.1.6 Output data

Netspot outputs: raw statistics, thresholds (SPOT will compute one threshold for each monitored statistic) and the alarms (in case of triggered threshold). Netspot can also send these to an influx

database, a time series database for high write and query loads. The influxdata can then be visualized with Grafana. Alternatively, they can be saved locally on a file.

Figure 22: Ratio of the ARP (address resolution protocol) packets observed from Grafana

3.2.1.7 Hardware Requirements

Devices running these analytics are required to have network connectivity and the capability of running minimal Linux distributions. Such devices can be based on either the x86 or ARM architecture. As a reference, the hardware specifications of a Raspberry Pi4 models are presented as a set of minimum requirements:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE.
- Gigabit Ethernet.
- 2 USB 3.0 ports; 2 USB 2.0 ports.

Figure 23: Raspberry Pi4, Riots mama device and apartment sensor

3.2.1.8 Privacy Considerations

Only statistical values are collected and analyzed on the edge. IP packet size and IP address, protocol and port counts are collected. All the data is processed locally in the edge.

3.2.1.9 Execution of the Analytics by the Analytics Toolbox

Information from the network traffic and sensors are parsed and converted to statistics. The statistics and alerts are passed to the algorithm and written to a file or a database to be handled by the Network protection manager.

3.2.1.10 Implementation Details

The detection process is composed of two phases, i.e., scoring and thresholding. The scores indicate how much of an observation is an outlier, and a threshold is used to make the final decision. If the abnormality score is higher than the threshold, the observation is considered as an anomaly.

For implementation purposes, a dedicated virtual machine with the python anaconda distribution was used to create a virtual environment with data processing, math, and visualization libraries necessary for the SPOT algorithm. The algorithm was first tested with csv files that contained aggregated NetFlow's. The test data which was used is available in the example usage of SPOT in GitHub. [NETSPOT2] The csv files contained fields for average packet sizes, number of unique source and destination port numbers and IP addresses, packet rates, ratios of Internet Control Message Protocol and Transmission Control Protocol packets with Synchronization flag set. The field containing the ratios of Transmission Control Protocol packets with Synchronization flag was parsed from the csv files.

An initial batch of data containing high number of synchronisation flag ratios between 0 - 0.1 was used for the calibration/initialization step. SPOT forms bindings between normal and abnormal events, based on that initial batch of data. After the calibration phase, more flows were computed, and the results showed that SPOT was able to flag the peak values over the normal bound as shown in Figure 24.

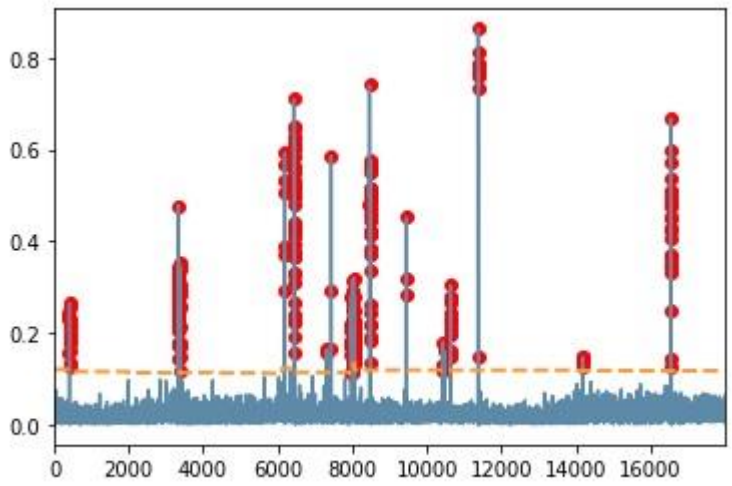

Figure 24: Results of SYN ratio analysis plotted with matplotlib [NETSPOT]

For initial live testing purposes, the Opensource GO-language implementation of SPOT, namely NetSpot, was used to assess the performance in different network situations. NetSpot includes all the

components for extracting network packets, analysing the relevant data, and outputting the statistics and alerts, which makes it suitable for easy testing of this solution.

Practical tests with NetSpot were done by running it on a Raspberry Pi4, which acts as the IoT gateway. In this setup, the sensor nodes send all their data to this gateway which encrypts it and then the gateway sends the encrypted data over a TCP connection to the cloud. Statistics from this traffic are calculated and monitored with the SPOT algorithm.

Normal network traffic for the sensors was generated and the algorithm was tuned according to those normal conditions. After that, anomalies were introduced in the network, in order to observe if any of the monitored statistics were triggered. Initials tests show that NetSpot successfully triggered alerts about Denial of Service (DoS) attacks and the active presence of the Mirai malware.

3.2.1.11 Preliminary Results

NetSpot was tested in a small test network including sensors, an IoT gateway and a Linux (virtual) workstation. A Raspberry4 Pi was used as the IoT gateway with NetSpot running on it. Sensor traffic was generated to be sent regularly to the cloud to simulate normal conditions for this specific network. The monitored statistics were selected based on the attack types against which SPOT was tested (i.e., DoS and Mirai botnet). Preliminary tests indicate that changes in the monitored statistic ratios, e.g., high ratio of Address Resolution Protocol (ARP) packets due to Mirai and excessive traffic (DoS) in the network results in statistics that trigger alerts, according to what was monitored during the tests.

If NetSpot is deployed on a specific host, the host with the anomalous traffic can be identified from the agent id, which is included in its transmitted packets if the device is sending the data to a centralized influx database. If it is deployed to a dedicated device which monitors all the statistic in the network, only the statistical information is shown, and the anomalies cannot be identified to a specific host. Therefore, SPOT is best working alongside some lightweight signature/rule-based intrusion detection system like snort, which is open source and contains several features that can be utilized for a more complete IDS solution.

3.2.2 Anomaly detection for IoT devices in a consumer home network

With the increasing proliferation of IoT devices, the need to specifically address security threats against such devices is becoming more and more important. For example, botnets have gained a lot of attention in recent years, as they can launch large-scale Distributed Denial-of-Service (DDoS) attacks and cause potential damage to companies and institutions. Aiming to drive an increase in security for IoT devices in particular, the Internet Engineering Task Force (IETF) proposed a new standard, called Manufacturer Usage Description (MUD) [LDR19] in 2019.

The MUD standard allows manufacturers of IoT devices to identify and create a secure and well-known operating profile that describes the so-called normal network behavior of a specific IoT device brand and model. The main idea behind this is to have an Access Control List (ACL) enforced as firewall rules on the local gateway. Because the ACL only allows communication that has been defined by the IoT device manufacturer, any unspecified communication can be blocked, which greatly reduces the threat surface against vulnerable IoT devices.

3.2.2.1 Aim of the Analytics

The aim of our (F-Secure) analytic is to detect compromised consumer-class home IoT devices within

a SIFIS-Home network. In a compromised state, an IoT device could unwantedly be enlisted into a botnet consisting of other compromised devices that may be used for e.g., DDoS attacks. The general principle of our analytic is to learn the behaviour of nearby connected devices. These *in situ* learnt communication models can then be used to detect device communication that is different compared to how a particular device typically communicates.

3.2.2.2 Requirements involving the analytics

The Anomaly detection for IoT devices in a consumer home network analytics is relevant for satisfying the following requirements defined in deliverable D1.2:

- Functional Requirements:
    - F-15 Requirement is satisfied because the usage description our analytic creates for a device allows us to identify abnormal behaviour.
    - F-19 The usage description our analytic creates shall be enforced as an access control list, through which unwanted traffic will be restricted.
    - F-20 Requirement is fulfilled because if the access control list from our analytic detects something anomalous a notification API is invoked to warn the user.
- Non-Functional Requirements:
    - PE-10 Traffic passes the access control list created by our analytic in real-time, which means detection and reaction can be performed in close to real-time.
- Security Requirements:
    - SE-01, SE-02 The SIFIS-Home APIs use secure protocols.
    - SE-13 In our analytic, data is not stored in any permanent location outside of the entity responsible for running the analytic.
    - SE-17 Similarly to requirement PE-10 above, our analytic operates and reacts in close to real-time.

3.2.2.3 Input Data

The input data is network traffic flows (netflows) as captured by the home router. Hence, visibility is limited to the traffic passing through the router. Given the circumstances, we do not have direct visibility to application layer information about what tasks and communication surrounding active devices are performing. However, netflow data contains crucial communication-describing parameters, such as IP addresses, port numbers, and protocol identifiers of connections in both outbound and inbound directions. These pieces of information are the building blocks used in our analytic for creating MUD files that describe device communication.

3.2.2.4 Analytic Design

The MUD standard is intended to define the expected behavior of a given IoT device by restricting its communications and network functions. For this purpose, the IoT device manufacturer must be able to provide a MUD file. Within the MUD file, the behavioral rules of a device will be expressed in the form of a YANG-based JSON model. The JSON model contains several sections, out of which the most important ones for our analytic are the restrictive policy-defining sections for both directions: “to-device” and “from-device”. These ACL sections define the allowed communication parameters for a particular IoT device. Each ACL consists of individual rules defining either allowed or blocked

communication. Parameters used to describe the ACLs are protocol, port, and IP address or DNS name.

While the MUD standard is getting a lot of attention from industry and academia, it is worth mentioning that real-world implementations are limited [Her21, MSZ21]. To the best of our knowledge, currently no manufacturers provide MUD files for their devices. A popular repository for MUD related resources was created by the IoT Traffic Analytics Research Group, School of EE&T, UNSW Sydney [NOK21]. Furthermore, several tools have been developed for creating and validating MUD files. One of these is MUDgee [HRR18], which can generate an "estimated MUD" file from a pcap capture file. To generate the MUD file, MUDgee must be configured by specifying the path to the pcap file, the default gateway information (IP or MAC address) for the device, and data about the device itself. This tool is interesting and useful for research purposes, but has some limitations for real implementations, as it needs pcap captures that are not always available.

One of our contributions in SIFIS-Home is to implement the creation of MUD files through our home router security product SENSE SDK. The main role of SENSE SDK is to collect and locally store all the information related to network flows from surrounding IoT devices. Each network flow is identified by protocol, port, and IP addresses. We use this data to generate MUD files related to single IoT devices. From here the need arises to understand how to optimize the processes, for example with the identification of a learning period long enough to predict all the possible behaviors of the IoT devices. Moreover, since it is possible to have multiple MUD files for the same IoT device, we are aiming to implement an efficient solution to aggregate and merge MUD files from several origins into one MUD file. Another point on which it is important to pay attention is the lifecycle of these MUD files, as there is the need that these files are periodically updated, in order not to cause blocking problems in case the device presents new legitimate behaviors.

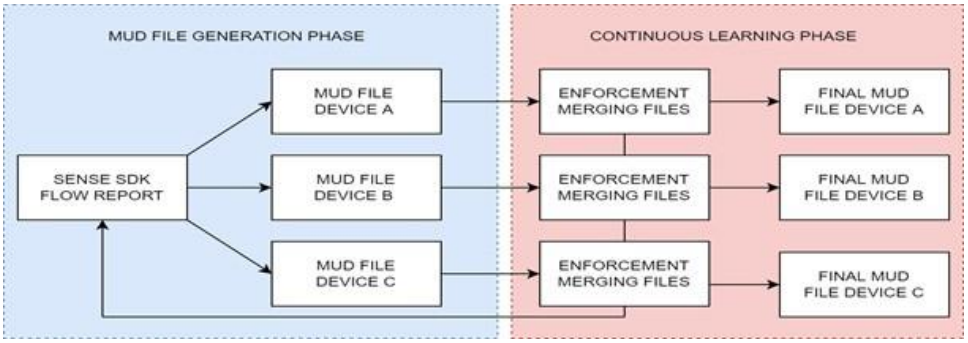

Figure 25: The two distinct stages of our analytic

In Figure 25, we can see the two phases of our proposed MUD file life cycle. The first is the generation of MUD files from the SENSE SDK flow report where a MUD file is generated for each device. The second phase is where the MUD files are merged. The minimum time resolution with which it is possible to extend the MUD files for each device is 3 hours (the time between the generation of a new flow report and the previous one).

Contrary to the abbreviation, the resulting self-generated MUD files are not created by the manufacturer. These generated MUD files should not be considered as an authoritative source of describing device behavior. However, our analytic leverages the structure and semantics of the existing standard. Compatibility with the existing MUD standard is favorable to us also because it allows our analytic to provide MUD-style device protection for devices that do not have MUD support from the manufacturer.

In our architecture the MUD is a first level of control of the flows. If a network flow is allowed by the rules defined in the ACL no actions need to be carried out. On the other hand, if a flow does not pass the ACL, further analysis is necessary. For example, Nakahara et al. [NOK21] showed that the use of a pseudo-MUD based ACL alone generates a high number of false positives, whereas the results improve with the addition of a machine learning algorithm (isolation forest) used to support the ACL.

3.2.2.5 Training and Testing

The training phase consists of the creation of MUD file from flow reports collected by the SENSE SDK. To achieve this, a python script able to take the information from our flow report and convert it into a MUD file is executed. As described in the analytic design section, the MUD file consists of different sections containing information about the file and policies that form a device specific ACL. In Figure 26 it is possible to see an example of a generated MUD file. As far as the testing part is concerned, as it is only relevant for the training of a machine learning process, in this case it is not carried out. However, initial testing of the analytic in practice has been carried out. Malicious network activities, such as a SYN flood attack, has been successfully blocked by deploying a generated MUD file onto the connecting network router. A more detailed description can be found in the Preliminary Results Section.


```
{
  "ietf-mud:mud" : {
    "mud-version" : 1,
    "mud-url" : "https://philipscapture.com/philipscapture",
    "last-update" : "2022-01-09T13:30:12.172-08:00",
    "cache-validity" : 100,
    "is-supported" : true,
    "systeminfo" : "philipsCapture",
    "from-device-policy" : {
      "access-lists" : {
        "access-list" : [ {
          "name" : "from-ipv4-philipscapture"
        }, {
          "name" : "from-ipv6-philipscapture"
        } ]
      }
    },
    "to-device-policy" : {
      "access-lists" : {
        "access-list" : [ {
          "name" : "to-ipv4-philipscapture"
        } ]
      }
    }
  },
  "ietf-access-control-list:access-lists" : {
    "acl" : [ {
      "name" : "from-ipv4-philipscapture",
      "type" : "ipv4-acl-type",
      "aces" : {
        "ace" : [ {
          "name" : "from-ipv4-philipscapture-0",
          "matches" : {
            "ipv4" : {
              "protocol" : 17,
              "destination-ipv4-network" : "216.239.35.0/32"
            },
            "udp" : {
              "destination-port" : {
                "operator" : "eq",
                "port" : 123
              }
            }
          },
          "actions" : {
            "forwarding" : "accept"
          }
```


Figure 26: Sample snippet from JSON MUD file

3.2.2.6 Output data

In principle, the output of the analytic is the ACL itself. Mainly it is a table inside the firewall that allows through rules to block or allow a communication from/to the device. A rule consists of a pair of IP addresses, port, protocol, and the name of the zones (e.g., WAN or LAN).

3.2.2.7 Hardware Requirements

The router responsible for collecting the input data must be able to A) collect flow reports, and B) store flow report data for local, and potentially also off-site processing. The creation of an estimated MUD file from flow report data can be done on the router itself, but can also be performed off-site, e.g., in a cloud backend. The hardware requirements for our analytic do not exceed the hardware specification of a typical consumer grade home router.

3.2.2.8 Privacy Considerations

IP and MAC addresses are generally considered as personally identifiable information. The anomaly detection analytic described in this Section handles both kinds of data. However, it does not store addresses of devices in any permanent location outside the component responsible for the analytic. The proposed analytic only uses IP/MAC addresses of sensor devices to identify a device in the local network, but this information is only used in local processing. Additionally, as proposed by the MUD specification [LDR19] local IPs are to be abstracted out. Output data of our analytic, i.e., the generated MUD files does not contain IP addresses. Instead, the MUD file addresses remote destinations in CIDR network block notation.

3.2.2.9 Execution of the Analytics by the Analytics Toolbox

Once access control rules defined by the MUD file have been enforced on the router, and upcoming communication does not pass the rules, a notification is sent via an API defined by the SIFIS-Home architecture to the Network Protection Manager component.

3.2.2.10 Implementation Details

In our analytic, detection of anomalous communications is achieved through matching upcoming connections from an IoT device to earlier connections that the same type of device has been doing in the past. Whenever an IoT device attempts to establish a connection to an unexpected remote endpoint and/or over an unexpected protocol, it triggers our analytic to consider the communication as anomalous. It does not directly imply that the communication is malicious. However, it raises the need to inspect the communication in more detail.

For our analytic to be able to evaluate upcoming communication attempts, we need to first collect and/or aggregate samples for the device in question. The usage description models we create in our analytic may also be complemented by aggregating usage description models and/or communication samples for the same device brand and model but originating from external sources.

The first stage ("MUD file generation phase" in Figure 25) of our analytic is implemented around a core loop consisting of the following stages:

- Data intake
- Preprocessing and aggregation (with, e.g., earlier data samples)
- Store data for the "Continuous learning phase"

The second stage ("Continuous learning phase" in Figure 25) of our analytic operates at runtime with

the following sequence of operations:

- Collect communication samples from surrounding IoT devices within the SIFIS home
- Match and merge collected samples to the current aggregated usage description
- Output/update usage description files for affected devices

Both stages are implemented in Python. Data processing and aggregation uses pyspark, but not to an extent that would exceed the specifications listed in the Section Hardware Requirements.

3.2.2.11 Preliminary Results

With the aim of providing a preliminary result, we have created a small environment consisting of a router connected to the Internet and a laptop acting as a router on which OpenWRT is installed and where a MUD manager runs. A Philips Hue bulb was used as a test IoT device. In Figure 27, it is possible to see an overview of our test environment.

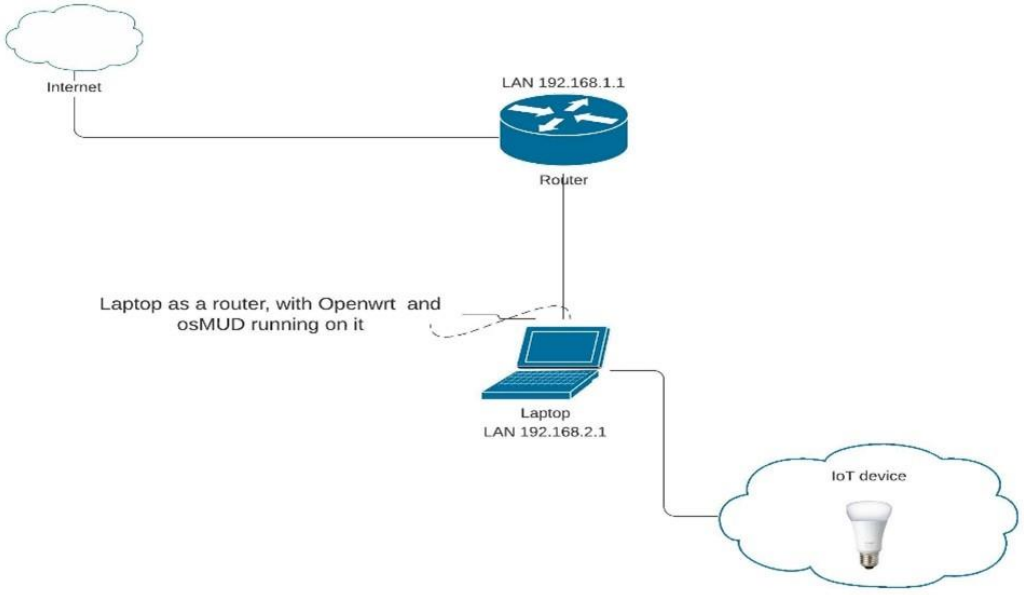

Figure 27: Test environment for preliminary results

The core component in our proof-of-concept testbed is the MUD manager, which takes care of creating and obtaining the behavioral profiles of the IoT devices. This architecture requires that each IoT device inserts its corresponding MUD URL within the DHCP request. After the DHCP invocation, the MUD Manager extracts the MUD URL and then sends a request to the server containing the MUD files. Once the MUD file is obtained, the MUD manager parses the file and extract the rules to be used for the creation of the ACL inside the firewall.

Currently, osMUD [OSM] is to our knowledge the most popular open source implementation of a MUD manager. It is designed to be easily integrated within an OpenWRT router. As shown in Figure 28, the environment is composed of the dnsmasq service that implements the DHCP server and a DNS cache. We also find the MUD manager (osMUD), the firewall and the server containing the MUD file. When the IoT device makes a DHCP request to the DHCP server, a script able to save all request data in a log

file is invoked. The log file is polled by the MUD manager and parsed, with the aim of extracting the MUD URL from which it can contact the MUD server to request the MUD file. Once obtained, the MUD file is parsed, and a script is invoked in order to create the firewall rules.

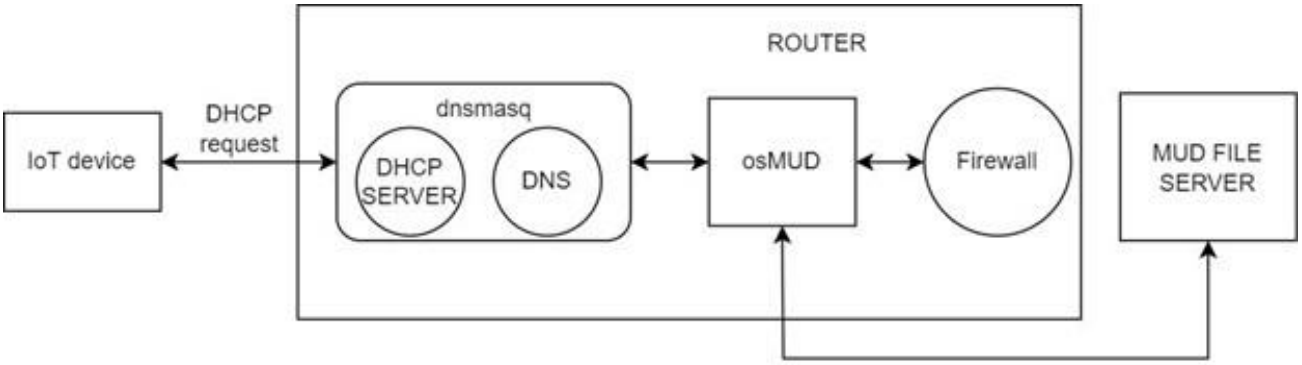

Figure 28: Block diagram of osMUD

Inside the OpenWRT router a MUD file has been saved. For simplicity of implementation and since still in a preliminary stage, we used a MUD file generated with MUDgee from a pcap capture file. The Philips Hue bulb was connected to the network and a pcap capture of the network traffic operation was performed for about 10 minutes using the tcpdump tool. Then the pcap file was used to create the MUD file. Not having yet available an IoT device that sends the MUD URL in the DHCP request, a modified version of osMUD has been implemented in order to use directly the MUD file locally stored in the router. After the device DHCP request, a script that osMUD invokes creates the ACL rules in the firewall using the MUD file specifications. In Figure 29, we can see the rules created by osMUD using the MUD file, all the traffic to/from the device is restricted and all the traffic from the device to outside the private subnet that was not previously allowed is dropped.

Figure 29: A view from the ACL entries in osMUD

A SYN flood attack was subsequently performed with the purpose of creating a behavior that was not included in the MUD file profile and generating a DROP action from the firewall. The SYN flood attack was carried out with the hping3 tool. This tool, given the IP address of the device, is able to send a large number of SYN packets to the victim. It is possible to specify the number of packets to send and the size of each one of those. In Figure 30, it is possible to see the event log file. The SYN packets are

dropped by the firewall.

```
Feb  8 08:02:44 OpenWrt kernel: [  602.439024] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=52.212.78.82 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=57527 DF PROTO=TCP SPT=57992 DPT=80 WINDOW=29
200 RES=0x00 SYN URGP=0
Feb  8 08:02:47 OpenWrt kernel: [  605.335286] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.12 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=49564 DF PROTO=UDP SPT=40750 DPT=123 LEN=56
Feb  8 08:02:47 OpenWrt kernel: [  605.340916] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.8 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=840 DF PROTO=UDP SPT=33336 DPT=123 LEN=56
Feb  8 08:02:47 OpenWrt kernel: [  605.343291] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.4 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=21813 DF PROTO=UDP SPT=57702 DPT=123 LEN=56
Feb  8 08:02:47 OpenWrt kernel: [  605.345680] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.0 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=28167 DF PROTO=UDP SPT=60766 DPT=123 LEN=56
Feb  8 08:02:50 OpenWrt kernel: [  608.708976] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=34.117.13.23 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=782 DF PROTO=TCP SPT=52196 DPT=443 WINDOW=292
00 RES=0x00 SYN URGP=0
Feb  8 08:02:52 OpenWrt kernel: [  610.468953] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=52.212.78.82 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=57528 DF PROTO=TCP SPT=57992 DPT=80 WINDOW=29
200 RES=0x00 SYN URGP=0
Feb  8 08:03:06 OpenWrt kernel: [  624.361399] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.12 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=50590 DF PROTO=UDP SPT=50939 DPT=123 LEN=56
Feb  8 08:03:06 OpenWrt kernel: [  624.364444] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.8 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=2203 DF PROTO=UDP SPT=51259 DPT=123 LEN=56
Feb  8 08:03:06 OpenWrt kernel: [  624.366936] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.4 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=23153 DF PROTO=UDP SPT=54721 DPT=123 LEN=56
Feb  8 08:03:06 OpenWrt kernel: [  624.369424] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.0 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=29648 DF PROTO=UDP SPT=48211 DPT=123 LEN=56
Feb  8 08:03:06 OpenWrt kernel: [  624.748770] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=34.117.13.23 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=783 DF PROTO=TCP SPT=52196 DPT=443 WINDOW=292
00 RES=0x00 SYN URGP=0
Feb  8 08:03:08 OpenWrt kernel: [  626.508743] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=52.212.78.82 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=57529 DF PROTO=TCP SPT=57992 DPT=80 WINDOW=29
200 RES=0x00 SYN URGP=0
Feb  8 08:03:25 OpenWrt kernel: [  643.385896] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.12 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=51653 DF PROTO=UDP SPT=41408 DPT=123 LEN=56
Feb  8 08:03:25 OpenWrt kernel: [  643.389998] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.8 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=3007 DF PROTO=UDP SPT=33120 DPT=123 LEN=56
Feb  8 08:03:25 OpenWrt kernel: [  643.392549] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.4 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=25043 DF PROTO=UDP SPT=56017 DPT=123 LEN=56
Feb  8 08:03:25 OpenWrt kernel: [  643.395030] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=216.239.35.0 LEN=76 TOS=0x10 PREC=0x00 TTL=63 ID=31142 DF PROTO=UDP SPT=56242 DPT=123 LEN=56
Feb  8 08:03:38 OpenWrt kernel: [  656.868368] DROP wan out: IN=br-lan OUT=eth1 MAC=d4:81:d7:9e:3c:e4:00:17:88:70:d3:ca:0
8:00 SRC=192.168.2.236 DST=34.117.13.23 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=784 DF PROTO=TCP SPT=52196 DPT=443 WINDOW=292
00 RES=0x00 SYN URGP=0
```

Figure 30: SYN packets being dropped by the ACL entries defined by our analytic

3.3 *Policy Enforcement*

In the contemporary Internet of Things (IoT) era, people can interact with a multitude of smart devices and applications always connected to the Internet, in the majority of today's environments, including the smart home. The result is a complex network of connected entities, be they physical devices or virtual services, that can communicate with each other, with humans, and with the environment. To preserve the security and the integrity of smart home environments, a smart home system should provide users with mechanisms to define security-based high-level policies on their devices and services, without the need to know (and specify) details that strongly depend on the underlying technology. In this section, we detail the Policy Enforcement task, ranging from its main goals to the implementation and preliminary results.

3.3.1 Aim of Policy Enforcement

The Policy Enforcement task aims to support users to express high-level policies like "*Do not record sound in the living room tonight*". These policies are responsible for ultimately ensuring that the behavior of the devices and applications involved in a given smart home adheres to the latest underlying policy description. The first goal of the Policy Enforcement task is to *translate* high-level policies into device-level policies, when possible. Stemming from a high-level policy, for instance, the system could limit the features of a smart home device, or it could inhibit the operation of a non-reconfigurable device. In addition, it could verify whether a given home configuration is compatible with one or more active (or suggested) policies. The second goal of the Policy Enforcement task is *to detect potential conflicts*

between high-level policies.

3.3.1.1 Requirements related to the analytics

The Policy Enforcement analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-30: the graphical user interface of this analytic shows the list of currently active policies.
  - F-31: the graphical user interface of this analytic shows the list of currently active policies.
  - F-33: through the graphical user interface of this analytic, the administrator can configure the policies to restrict/enable access to functionalities.
  - F-47: this analytic translates high-level policies into device-level policies.
  - F-48: this analytic translates high-level policies into device-level policies by considering the capabilities of the involved devices.
  - F-49: this analytic translates high-level policies into device-level policies by considering the devices that are available in the smart home.
  - F-50: the translated device-level policies include restrictions that specify when the policies must be applied.
  - F-51: this analytic notifies the user when a high-level policy cannot be translated.
  - F-52: this analytic can detect inconsistencies and redundancies in high-level policies.
  - F-53: this analytic exposes an API to retrieve the list of defined policies.
- Non-Functional Requirements:
  - PE-20: this analytic translates high-level policies into device-level policies in less than 60 seconds.
  - PE-21: this analytic provides the list of policies in less than 30 seconds.
  - US-01: the graphical user interface of this analytic can be used by users without any technical backgrounds.
  - US-03: the graphical user interface of this analytic is implemented by following standard accessibility guidelines.
  - US-06: the graphical user interface of this analytic is explorable.
  - US-15: an untrained user can create a high-level policy in less than 5 minutes.
  - US-16: an untrained user can create a high-level policy in less than 5 minutes.
  - TE-01: this analytic uses Java version 8 or higher to interact with the *sifis-home ontology*.
  - TE-02: this analytic exposes HTTP(S) APIs to retrieve/insert information from/into the *sifis-home ontology*.
  - TE-03: the software of this analytic is hosed on a high-availability server.
  - TE-04: internet connectivity should be present to import existing ontologies into the *sifis-home ontology*.
- Security Requirements: SE-10
  - SE-10: High-level policies are stored in a secured database.

3.3.2 Input Data

The Policy Enforcement task needs the following inputs:

- a machine-readable description, e.g., in the JSON format, of the high-level policies to be translated;
- a machine-readable description, e.g., in the JSON format, of the devices that are currently

installed in the smart home, including their capabilities, current statuses, and position;

- a machine-readable description, e.g., in the JSON format, of the applications that operates in the smart home, including their capabilities and current statuses.

3.3.3 Policy Enforcement Design

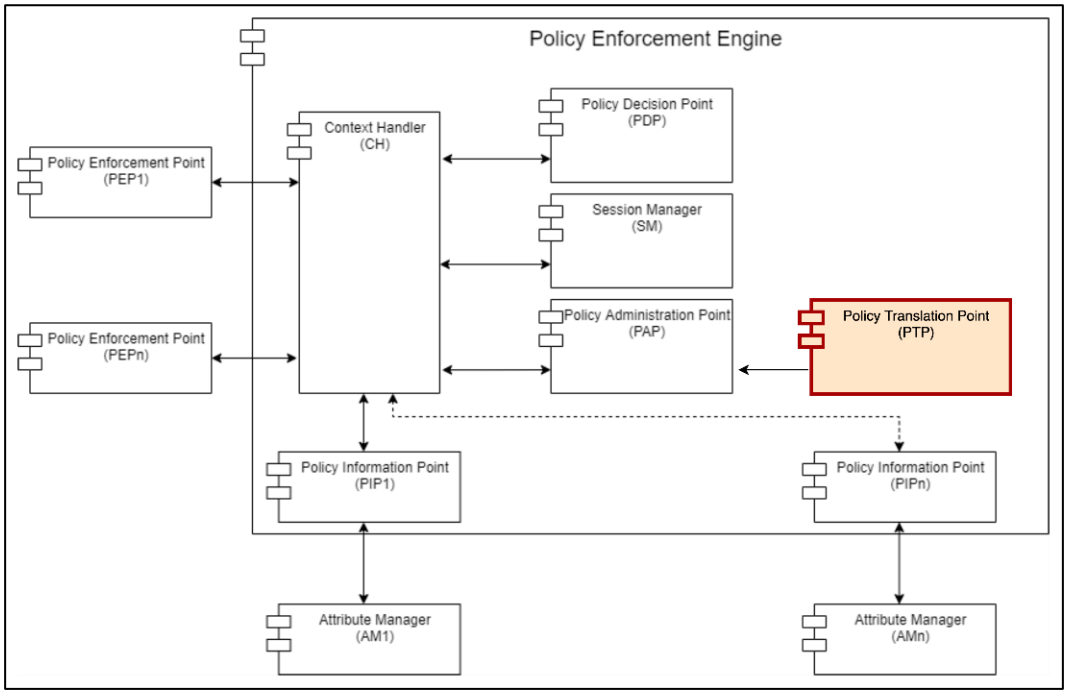

Figure 31: A focused view of the Policy Enforcement Engine with the Policy Translation Point (PTP)

All the software modules responsible to perform the task run in the *Policy Enforcement Engine* module. In particular, the *Policy Enforcement Engine* module contains a specific submodule for this purpose, named *Policy Translation Point (PTP)*, that interacts with the *Policy Administration Point (PAP).* The software running in the *PTP* exploits external modules to:

- get information about the high-level policies (*Settings*), devices (*Node Manager*) and applications (*Application Manager*) installed in the smart home;
- send notifications and alarms to the user in case of conflicts in high-level policies (*Evaluator/Notifier* and *Alarms/Log*).

3.3.4 Output data

The Policy Enforcement task produces two kinds of outputs:

- alarms to notify the presence of conflicts, e.g., inconsistencies and redundancies, among high-level policies.
- device-level policies in the XACML [XACML13] format that specialize the (safe) high-level policies on the devices and applications installed in the smart home.

3.3.5 Hardware Requirements

The *Policy Translation Point (PTP)* is hosted on a smart-home gateway that serves as a high-availability server. The gateway has access to the Internet to import and reuse external ontologies.

3.3.6 Privacy Considerations

The Policy Enforcement task do not present any major privacy risk, as it does not directly interact with real devices and applications in the smart home. The definition, verification, and translation of high-level policies is intended to be performed only by specific authorized users (e.g., an administrator), by means of a web user interface providing a secured login mechanism.

3.3.7 Execution of the Analytics by the Policy Enforcement Engine

The *Policy Enforcement Engine*, and, in particular, the *PTP* submodule continuously checks if there are new high-level policies to be translated from the *Settings* module. Furthermore, it collects and keeps up-to-date information about the devices and applications that are installed in the smart home. To this end, it exploits the *Node Manager* and *Application Manager* modules, respectively. For what concerns the devices, the *PTP* collects capabilities (e.g., capable of audio-recording), current statuses (e.g., on/off), and position in the home (e.g., currently in the living room). For what concerns the applications, the *PTP* collects capabilities and current statuses.

When new high-level policies are specified, the *PTP* first checks if there are conflicts among the currently available high-level policies. Such conflicts can be classified in two main categories:

- redundancies, i.e., a set of two or more policies that produce equal or overlapping results (e.g., *do not record audio* and *do not record anything*) in the same contextual situation (e.g., *in the room from 9 PM to 6 AM*);
- inconsistencies, i.e., a set of two or more policies that produce contradictory actions (e.g., *record audio* and *do not record audio*) in the same contextual situation (e.g., *in the living room from 9 PM to 6 AM*).

When some conflicts are detected, *PTP* exploits the *Evaluator/Notifier* and *Alarms/Logs* modules to send an alarm to the user. Instead, when the available high-level rules are “safe”, the *PTP* uses the information about devices and applications to translate them into a set of device-level policies in the XACML formalism. In particular, the translation process is performed through a semantic reasoning process over a custom SIFIS-Home ontology.

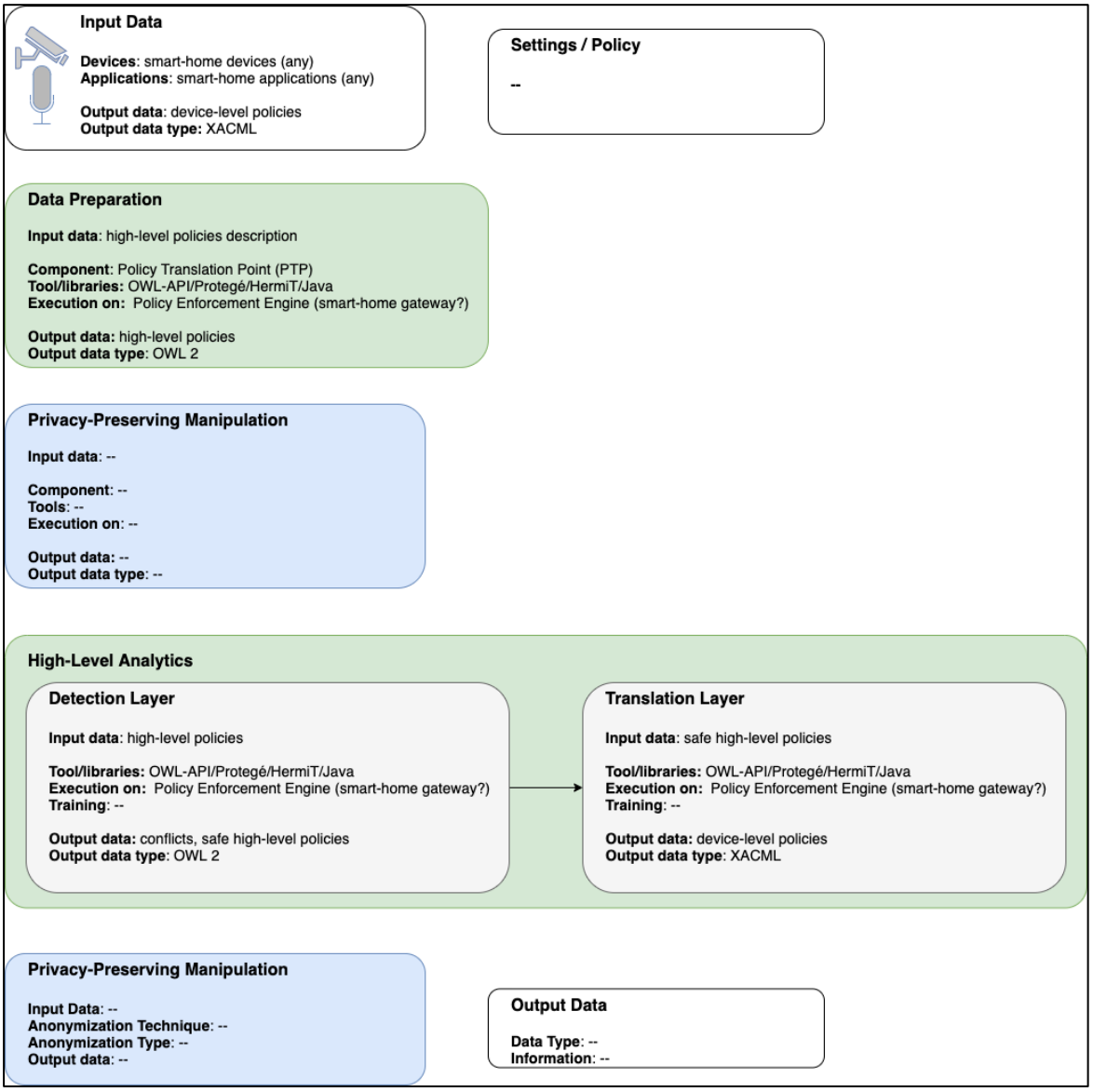

Figure 32: A flowchart that details the different steps of the Policy Enforcement task

Figure 32 details how the Policy Enforcement task is executed, by presenting a flowchart of the main task's steps:

- **Data Preparation.** The first step performed by the *PTP* submodule is the insertion of the available high-level policies into the SIFIS-Home ontology. Such a process decomposes each policy's description into two parts, i.e., the *contextual situation* in which the policy should be executed and the *action* that should be automatically performed. Then, it links the two parts to the proper classes included in the SIFIS-Home ontology. The output of the Data Preparation step is therefore a set of OWL high-level policies.
- **Detection Layer.** The *PTP* submodule analyzes the OWL descriptions of the available high-level policies and finds potential conflicts, i.e., redundancies and inconsistencies, by employing a semantic reasoning process. The output of the Detection Layer step is a set of possible conflicts that can be notified to the user and a set of OWL high-level policies that can be considered as safe.

- **Translation Layer.** The *PTP* submodule uses the safe OWL high-level policies to generate a set of device-level policies in the XACML formalism. As the detection of conflicts, also the translation step is performed through a semantic reasoning process that exploits the information included in the SIFIS-Home ontology. The XACML policies will be then enforced in the devices and applications installed in the smart home by other modules.

3.3.8 Implementation Details

The PTP submodule is a Java web server that has been implemented through the Spring framework. It exploits the following libraries:

- **OWL-API [OWLAPI]:** a Java API and reference implementation for creating, manipulating and serialising OWL Ontologies.
- **HermiT [HERMIT]: a** reasoner for ontologies written using the OWL language.
- **JGraphT [JGRAPHT]:** a Java library of graph theory data structures and algorithms.

3.3.8.1 Translation

To translate high-level policies and detect conflicts among them, PTP exploits *sifis-home ontology*. The ontology models high-level policies, smart home devices/applications, and users by exploiting state-of-the-art vocabularies like foaf [FOAF] (http://xmlns.com/foaf/0.1/) and EupONT [CDM19a] (http://elite.polito.it/ontologies/eupont.owl). In addition, the PTP module has access to a local MySQL database for storing high-level policies. A preliminary version of the PTP submodule has been implemented by taking into account simple high-level policies that are defined by a trigger and action. In the initial version of the *sifis-home ontology* , we included ithe following triggers and actions:

- *Temporal triggers*: events that fire every morning, afternoon, evening, or night, respectively.
- *Video actions*: actions that allow or forbid video recording in a given location, e.g., the bedroom.
- *Audio actions:* actions that allow or forbid audio recording in a given location, e.g., the bedroom.

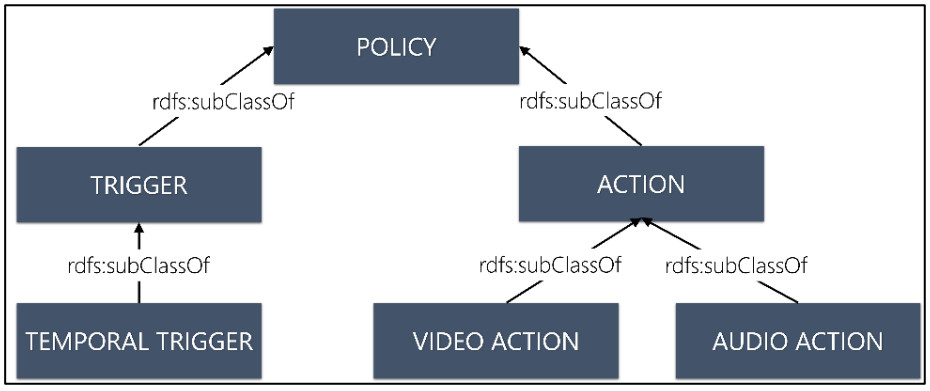

Figure 33: Modeling of a policy inside the sifis-home ontology

Figure 33 shows how a policy is modeled inside the *sifis-home ontology*. The OWL class *"POLICY"* has two subclasses, i.e., *"TRIGGER"* and *"ACTION"*. In the initial version of the *sifis-home ontology*, a set of OWL restrictions have been added to specify that a policy must have a single trigger and a single action. The *"TRIGGER"* and *"ACTION"* classes are in turn specialized in a hierarchy of OWL

subclasses representing events and actions of different categories. These hierarchies of subclasses are expressed at different levels of abstraction: this potentially allows users to specify high-level policies in different ways, by choosing to be more or less specific. Figure 34 exemplifies some video-related actions included in the initial version of the *sifis-home ontology*.

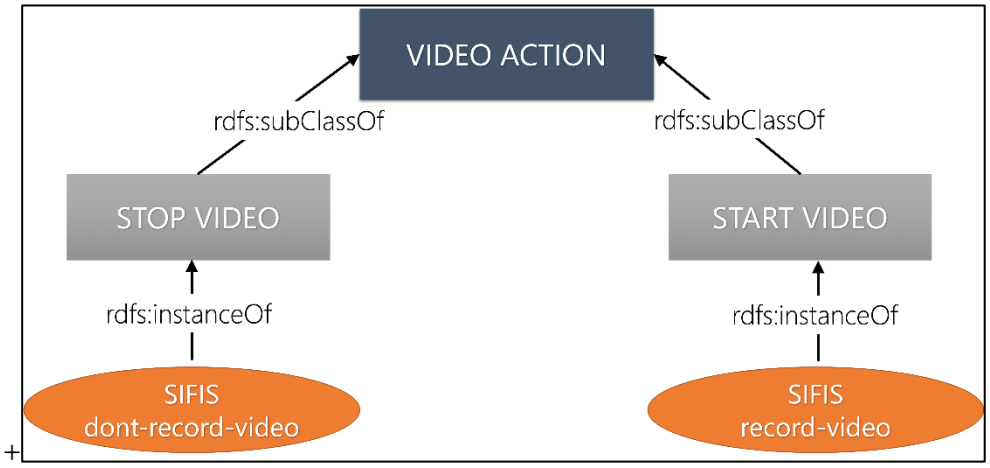

Figure 34: Some video-related actions included in the initial version of the sifis-home ontology

For example, the *"SIFIS dont-record-video"* action is a RDFS instance of the *"STOP VIDEO"* class, while the *"SIFIS record-video"* action is a RDFS instance of the *"START VIDEO"* class.

3.3.8.2 Detection of conflicts

In order to detect inconsistencies and redundancies in high-level policies, the PTP module exploits a Semantic Colored Petri Net (SCPN) formalism [CDM19b]. Petri nets are bipartite directed graphs, in which directed arcs connect places and transitions. Places may hold tokens, which are used to study the dynamic behavior of the net. Petri nets can naturally describe rules like high-level policies as well as their non-deterministic concurrent environment, and they easily allow the step-by-step simulation of the run-time behaviors of the modeled policies: by firing a transition at a time, tokens move in the net by giving the idea of a possible execution flow. As a member of Petri nets family, Colored Petri Net (CPN) combine the strengths of ordinary Petri nets with the strengths of a high-level programming language. In SCPN, each token assumes different semantic "colors" by moving in the net. These colors represent the classes of the *sifis-home ontology*, and allow the inference of more information from the simulation of the net, i.e., to discriminate between problematic and safe high-level policies. Figure *35* summarizes the adopted approach. Specifically:

- High-level policies' triggers and actions are modeled as places in the Petri Net. When a trigger is in common between more than one policy, the associated places are duplicated and connected through a dedicated copy transition (*TCopy*). When a token is in the original place, the copy transition simply replicates the token in each copied place. Instead, action places can be directly reused by policies that have the same action.
- Places can be connected each other through a policy transition (*TPolicy*), i.e., a connection between the trigger and the action of the same policy, or through an activate transition

(*TActivate*), i.e., a connection used when an action of a high-level policy triggers the event of another high-level policy.

- Places, i.e., high-level policies' triggers and actions, are labelled with the corresponding OWL classes extracted from the *sifis-home ontology*.

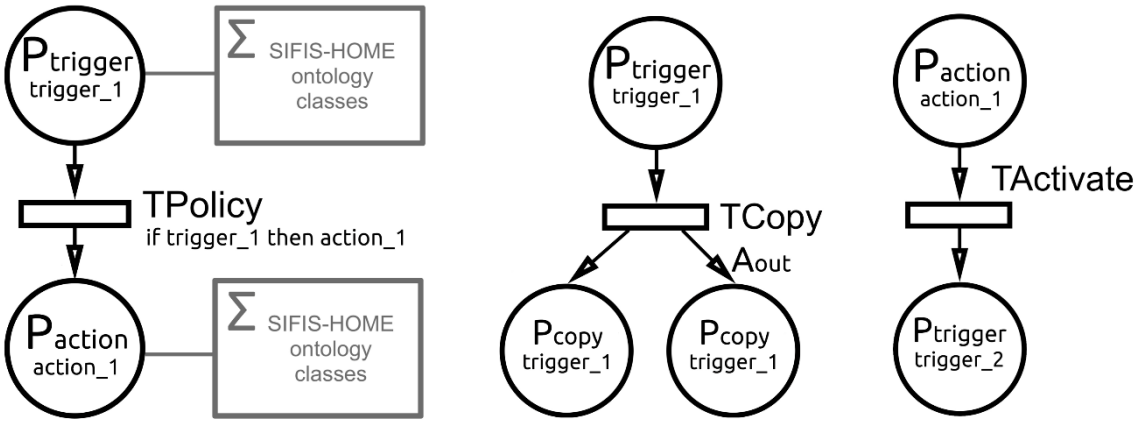

Figure 35: The Semantic Colored Petri Net (SCPN) formalism adopted to model the run-time behavior of high-level policies and detect inconsistencies and redundancies

In order to detect inconsistencies and redundancies, high-level policies are first translated into the corresponding SCPN. Then, the network is executed by placing a token in the trigger places.

3.3.8.3 Instantiation and Graphical User Interface

The *sifis-home ontology* has been instantiated to model a hypothetical smart home that includes the following devices and applications:

- 2 smart cameras in the bedroom;
- 1 smart camera in the living room;
- 1 smart speaker in the bedroom;
- 1 smart speaker in the living room;
- 1 calendar application.

To demonstrate how the PTP module works, a custom web user interface has been implemented by exploiting the Angular JS framework. The interface allows:

- The definition of high-level policies (see Figure 36)

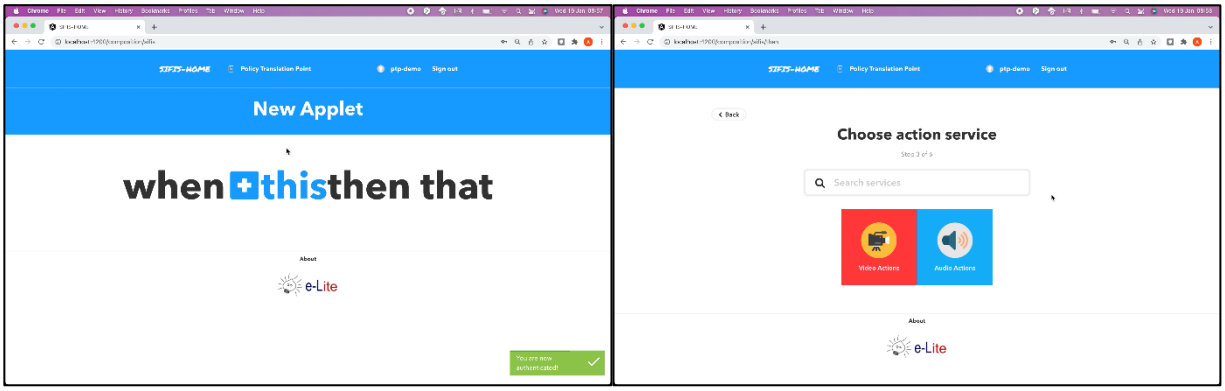

Figure 36: Definition of high-level policies

- The translation of high-level policies into XACML policies (see Figure 37)

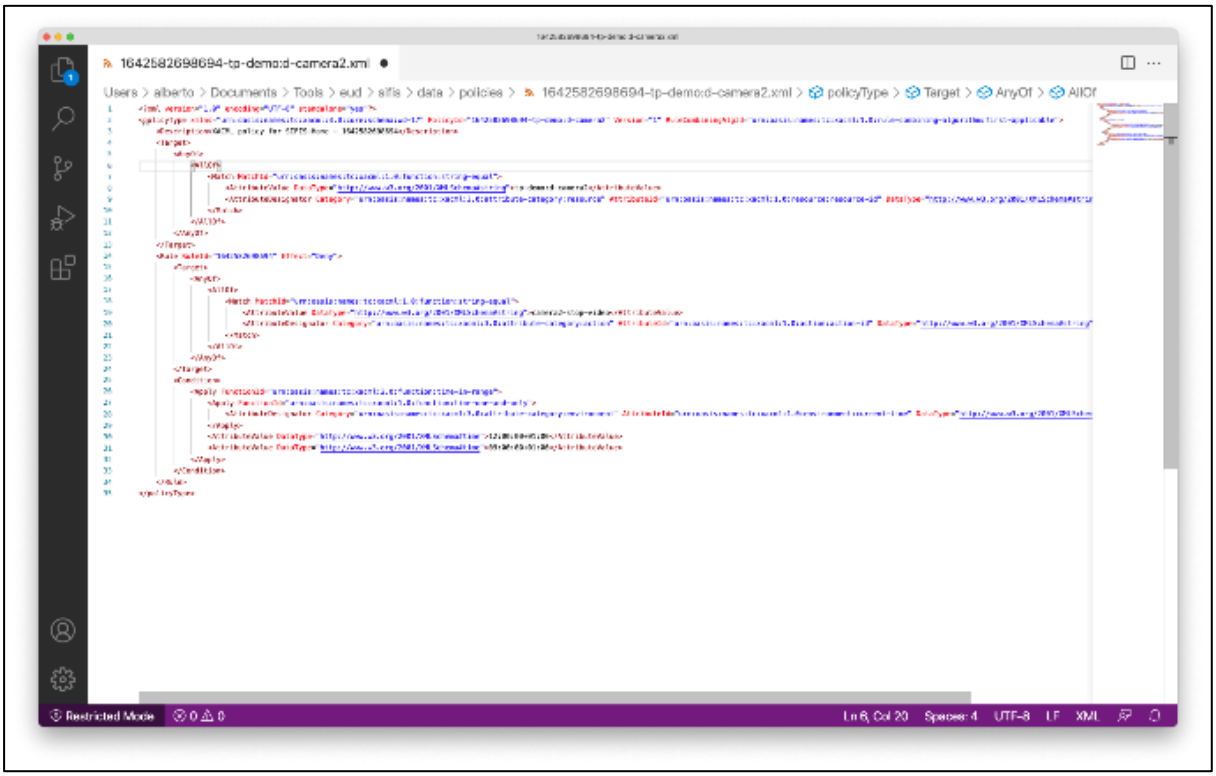

Figure 37: Translation of high-level policies into XACML policies

- The detection of conflicts, i.e., redundancies and inconsistencies, among high-level policies (see Figure 38)

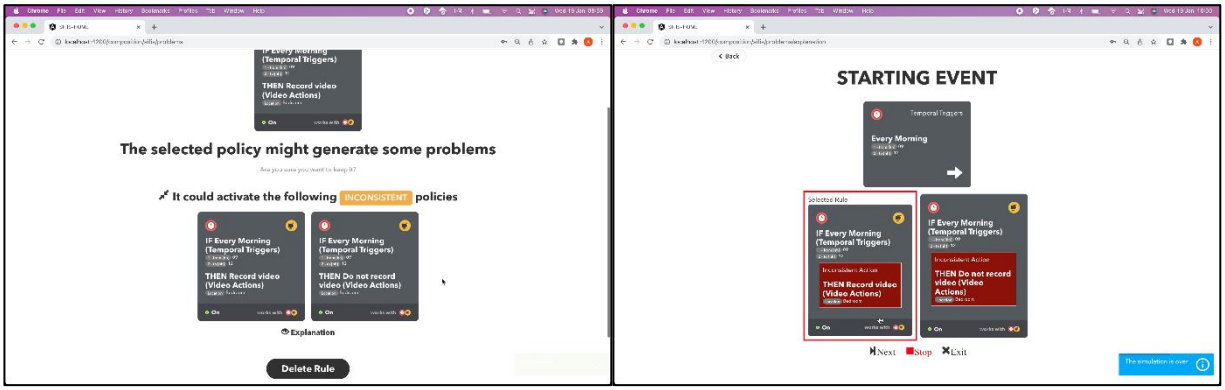

Figure 38: Detection of conflicts

3.3.9 Preliminary Results

The preliminary implementation of the PTP submodule is able to translate high-level policies composed of temporal triggers and video/audio actions into XACML policies for specific devices and applications installed in the hypothetical smart home. As an example, the policy:

Every morning between 9 and 12 AM, do not record any video in the living room

is translated by the PTP submodule into 1 XACML policies, as there is only one smart camera in the living room (*tp-demo:d-camera1*):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<policyType xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="1642586591516-tp-demo:d-camera1" Version="1"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">
    <Description>XACML policy for SIFIS-Home - 1642586591516</Description>
    <Target>
        <AnyOf>
            <AllOf>
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">tp-demo:d-camera1</AttributeValue>
                    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                        DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                </Match>
            </AllOf>
        </AnyOf>
    </Target>
    <Rule RuleId="1642586591516" Effect="Deny">
        <Target>
            <AnyOf>
                <AllOf>
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">camera1-stop-video</AttributeValue>
                        <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>
        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
                    <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
                        DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true" />
                </Apply>
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">12:00:00+01:00</AttributeValue>
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00+01:00</AttributeValue>
            </Apply>
        </Condition>
    </Rule>
</policyType>
```

The preliminary implementation of the PTP submodule is also able to detect redundancies and inconsistencies among high-level policies. The following table reports two conflicts that have been assessed through the custom web interface:

Conflict	**Policy 1**	**Policy 2**
Redundancy	*Every night between 01 and 06 AM, do not record any audio in the bedroom*	*Every night between 01 and 06 AM, do not record any audio in the bedroom*
Inconsistency	*Every morning between 9 and 12 AM, do not record any video in the bedroom*	*Every morning between 9 and 12 AM, record video in the bedroom*

More examples are available in the repositories that contain the source code of the PTP submodule and its web interface.

3.4 *Privacy Aware Speech Recognition*

3.4.1.1 Aim of the Analytic

The Privacy-Aware Speech Recognition (PSR) aims to recognize and translate spoken language into text through computational linguistics in a privacy-preserving way. The analytic performs a speech-to-text process, listening to audio signal and delivering an editable and verbatim transcription of the speeches present in an audio signal where the sensitive information have been anonymized. When required, a privacy preserving version of the original audio can be obtained back as well by simply performing a text-to-speech translation of the anonymized text result. The produced privacy preserving transcriptions and audio can be then sent to external service since they don't reveal any sensitive information.

3.4.1.2 Requirements related to the analytics

The Privacy-Aware Speech Recognition analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-07: The analytic provides an Automatic Speech Recognition that can be used to control the home appliances in the smart home;
- Non-Functional Requirements:
  - PE-06: The analytic performs an audio transcription in less than 2 seconds;
- Security Requirements:
  - S-04: The analytic avoids the disclosure of sensitive information by anonymizing it when processing input data;
  - SE-50: The analytic applies an anonymization of the voice timbre in order to make the identity of the speaker not identifiable;
  - SE-51: The analytic preserves only the foreground audio while removing all the background noise;
  - SE-52: The analytic provides the possibility to anonymize the sensitive information from the audio and textual audio translation.

3.4.1.3 Input Data

The Privacy-Aware Speech Recognition needs as input data an audio sample containing a voice to translate. An audio sample compatible with the current implementation of the analytic should have a sampling rate of 16000Hz, with 1 single channel (mono) represented in 16 bit. Such an audio sample can be captured by any device and, in the case of different audio formats, a pre-processing step should be performed to adapt the audio sample to the mentioned input analytic format.
The other (optional) input taken by the analytic is the list of textual entities to anonymize from the audio. The textual entities managed by the analytic are listed in Table 6.

Table 6: Textual Entities managed by the analytic

Entity	Example
Location	Via G. Moruzzi 1, Pisa, Italy
Date-Time	2019/09/29, let's meet tomorrow at 6pm
Person	Mr. White Jhonson
Organization	Google inc.

3.4.1.4 Analytic Design

The pipeline of the analytic is composed of three different components, as showed in *Figure 39*.

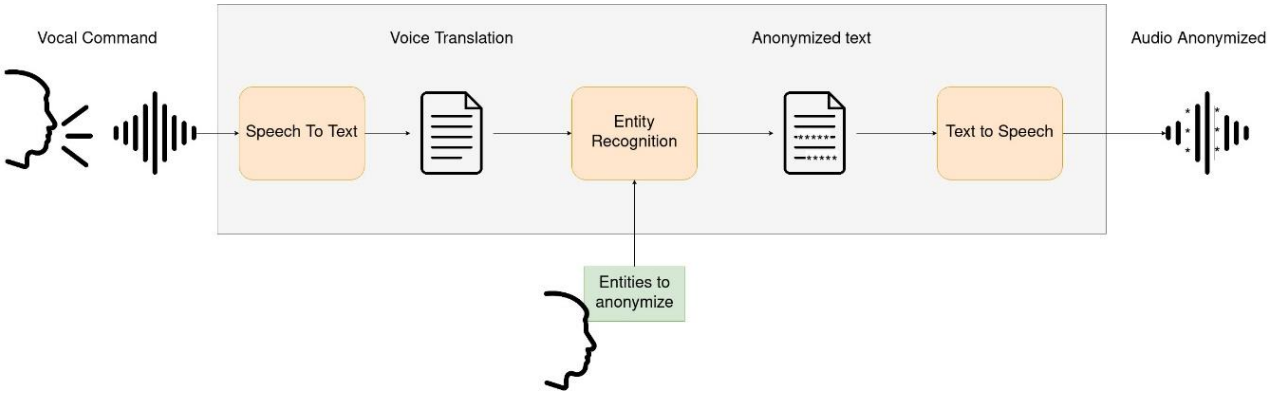

Figure 39: Pipeline of the privacy aware speech recognition analytics

The analytic workflow starts with the user providing the application with the textual entities that to anonymize. Such a setup affects the following entity recognition steps and the results of the analytic. The selectable entities are presented in section 3.4.1.3
After the entity settings, the user provides the analytic with a voice command, in order to starts the processing as described below.

The **Speech To Text (STT)** component translates the voice command into textual information, by using a machine learning algorithm. The obtained voice translation is the input of the **Entity Recognition** component. By considering the entity settings provided by the user, it replaces any detected entity with anonymous information. Finally, the anonymized textual data are passed to the **Text To Speech** component which creates a new equivalent audio sample, as an anonymized version of the original audio sample ready to be sent externally.

Speech To Text (STT)
The used Speech To Text is Mozilla Deep Speech [Mozilla, HCC14]. It is an open-source engine which uses a machine learning mechanism for voice recognition. The core of the architecture is a Recurrent Neural Network (RNN) model that takes an audio spectrogram as input and produces the relative text translation as output. The training set is a set of pair, (x,y), where x represents the audio file and y its translation. Every training element is a temporal frame of the original audio sample. The features are extracted through the Mel Frequency Cepstral Coefficients (MFCC) technique as presented in Figure 40. The main component of the MFCC technique are the following:

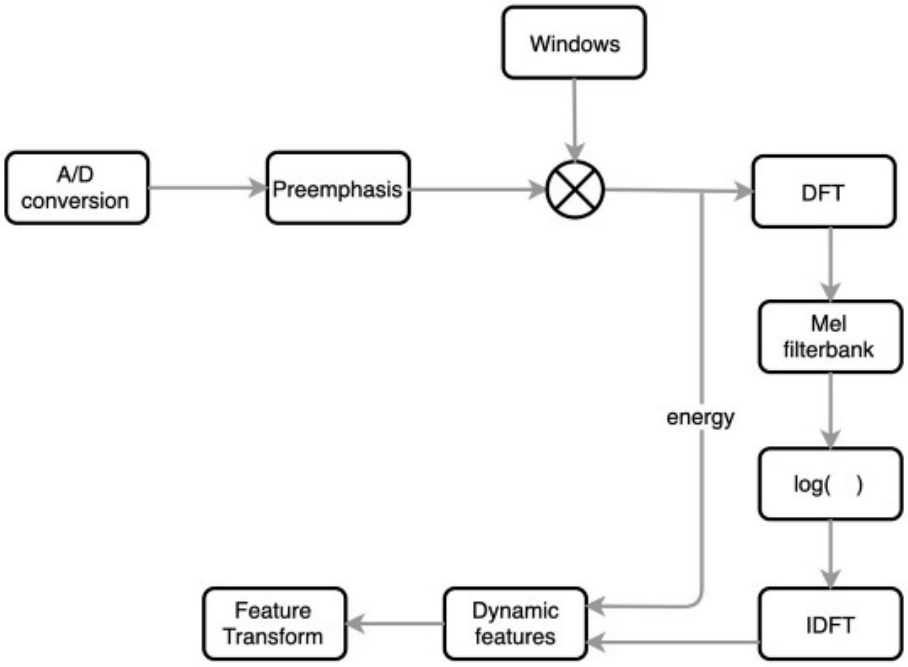

Figure 40: Feature extraction

- ***A/D Conversion***: The analog signal is converted into a digital signal with sampling rate 8 or 16KHz
- ***Preemphasis***: In this phase, the high-level frequency energy is increased to enhance phonetic accuracy.
- ***Windowing***: The audio segment is divided into frames from which the features are extracted.
- ***DFT***: The signal is transformed from time domain to frequency domain through the Discrete Fourier Transform (DFT).
- ***Mel-Filter Bank***: Such a filter creates a link between the sample frequency and the audible frequency.
- ***Applying Log***: The log function is applied to simulate the human perception.
- ***IDFT***: The Inverse Discrete Fourier Transform is applied (IDFT).
- ***Dynamic Features***: From the 13 features extracted from the previous steps, other 13 are added by computing the first and second derivate of the computed features.

The network converts each input x into a sequence of characters with the relative probabilities of the transcription.

$$y_t=(c_t|x)$$

where y_t is the probability to have the character c_t given the audio x. The set of characters depends on the language alphabet:

$C=\{a,b,c,....,z,....,blank\}$ where *blank* is the character transition.

The RNN is composed of 5 layers as showed in Figure 41

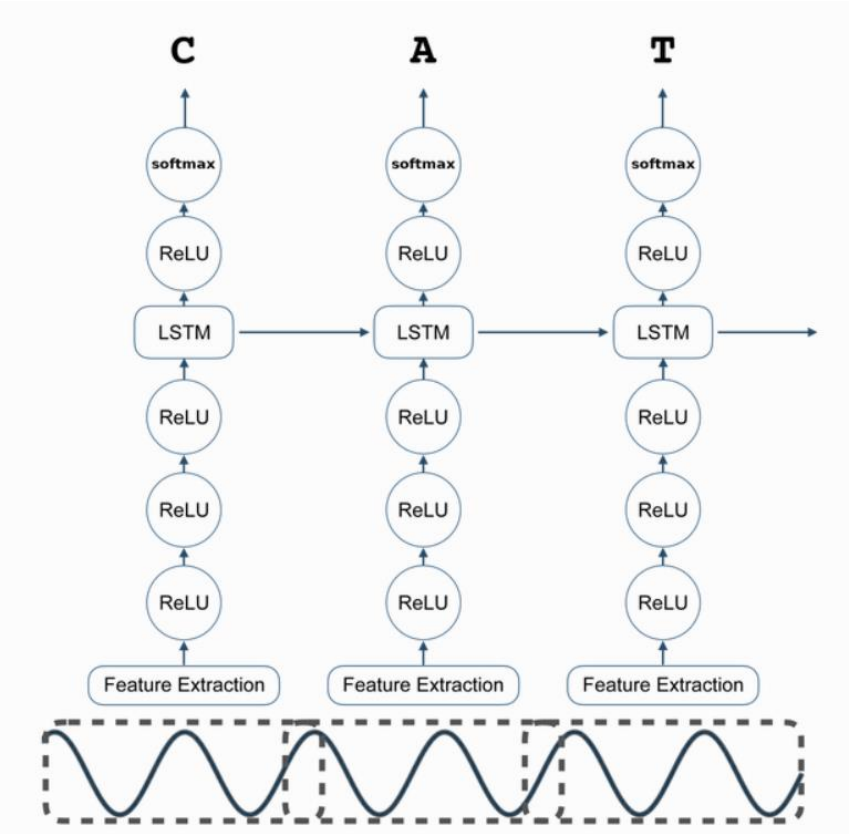

Figure 41: Layers of the RNN [ABD19]

Entity Recognition

The Name Entity Recognition is the process of identifying entities containing key information from a text. They are phrases or words which can be categorized in a specific topic, i.e., names, address, telephone numbers.

The pipeline related to the entity detection is showed in Figure 42.

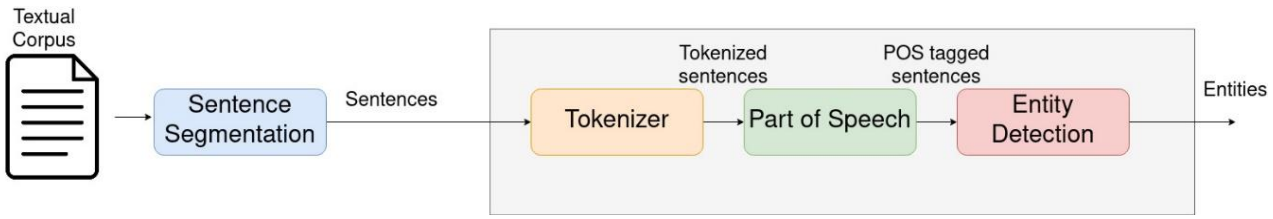

Figure 42: Pipeline of the entity detection

The first step in the entity recognition process is sentence segmentation, i.e., the process of determining the beginning and conclusion of a sentence, which largely depends on the language being used. The tokenization splits a sentence into smaller pieces known as tokens. These tokens are typically words, numbers, or punctuation marks. Each token is passed to the Part of Speech process which identifies the words by a part-of-speech (POS). Specifically, it detects the POS showed in Table 7.

Table 7: Part of Speech Description

POS	Description
NN	singular or plural noun
DT	determiner
VB	verb, base form
VBD	past tense verb
IN	preposition or subordinating conjunction
VBZ	verb, third-person singular present
NNP	singular proper noun
“TO”	Word “TO”
JJ	adjective

The last step is entity detection, that infers the entity of the word (e.g., Location, Name, time) from a token and its relation with the adjacent tokens .

SpaCy network
A state-of-the-art network able to perform Entity recognition is the SpaCy deep learning network [MHH21]. SpaCy is a Python library that takes text as input, applies tokenization, and encodes the result into hash values (i.e., 3197928453018144401 for the word “coffee”). In the embedding stage, the features (prefix, suffix, syntax) are used for the extraction of hash values that reflect word similarities, and a vocabulary is built.
In order to exploit adjacent vectors during the encoding state, values are processed by the Convolutional Neural Network (CNN) and merged with their context. The result of the encoding process is a matrix of vectors of hashes that represents the original text. Before the prediction of an ID, the matrix has to be passed through the Attention Layer of the CNN, by using a query vector to summarize the input. During the prediction state, a Softmax function is used for the prediction of a super tag with part of speech and morphology information. Similarly, for named entities, the available class is predicted. After the training process of the model, the CNN can be used for NLP tasks.

OpenNLP
OpenNLP is a machine learning-based toolbox for processing natural language text written in Java [Apache]. It has many features, including tokenization, lemmatization, and part-of-speech (PoS) tagging. Named Entity Extraction (NER) is one feature that can assist us to comprehend queries.

3.4.1.5 Training and Testing

Dataset
The STT component is trained with the Common Voice Dataset. It is the main multilingual collection of transcribed speech. It is used for Automatic Speech Recognition training, but it is applied in other domains (e.g., language identification,). The dataset is created by the Common Voice project, a Mozilla initiative that consists of a webservice able to gather user's voices. Each user can contribute by recording their voice by reading sentences displayed by the webservice on the screen. The last dataset release contains 38 different languages collected and over 60,000 individuals have participated, resulting in 2,500 hours of collected audio. Table 8 shows the number of individuals and hours as to collected English audio.

Table 8 Dataset Description

Language	***Voices***	***Hours***	
		Total	***Validated***
English	*39,577*	*1,087*	*780*

The STT component exploits the pre-trained model trained with the English language of the Common Voice dataset using audio signals in "WAV" format with a sampling frequency of 16KHz on a single channel.

3.4.1.6 Output data

The Privacy Aware Speech Recognition can provide two different outputs described as following:

- **Anonymized textual translation**: It represents the audio input translation in textual format after the anonymization of the sensitive information specified in the user preferences. In the output translation, all the sensitive textual entities detected by the analytic are replaced by default keywords which preserve the essence of the word without revealing the exact word.
- **Anonymized voice information**: It represents the audio reconstruction of the anonymized textual translation.

3.4.1.7 Hardware Requirements

The Privacy Aware Speech To Text analytic can be deployed in three different platform:

- Mobile
  - Android version 10
  - Wi-Fi, 4/5G connection
  - RAM: 8GB
  - Processor: 8-Core 2GHz
- Raspberry Pi
  - OS: Ubuntu 20
  - Version 4
  - Processor: Quad core Cortex-A72 (ARM v8) 64 bit 2.5Ghz
  - RAM: 8GB
  - Wi-Fi connection
- PC
  - OS: Ubuntu 20
  - RAM: 8GB
  - GPU with compute capability greater than 6

3.4.1.8 Privacy Considerations

The Privacy-Preserving Speech analytic elaborates voice samples taken as input data. In the SIFIS-home scenario, such voice samples have been recorded by the sensors deployed in the Smart home. Consequently, such data can contain sensitive and personal information, and can be used for identifying the speaker. For this reason, that information must be treated in accordance to the GDPR normative, and the transfer of transcriptions to a third-party application could compromise the compliance to such normative. Hence, an additional aim of the analytic is the one to avoid the spreading of personal data outside the SIFIS-Home context. To this end, the analytics also performs the replacement of all the sensitive information with a set of default keywords which represent only the named entity of the original word. For instance, "John" would be replaced by "Person", "Google" would be replaced by

"Organization", and so on.

3.4.1.9 Implementation Details

Speech To Text

Android implementation

In Android, the STT is implemented through the Mozilla Deep Speech network, which is pre-trained with the Common Voice dataset. The network is embedded in an Android application producing a model conversion in Tensorflow Lite.

The acoustic and language model are provided into the apk of the Android application in the asset folder in a hierarchical way as represented in Figure 43.

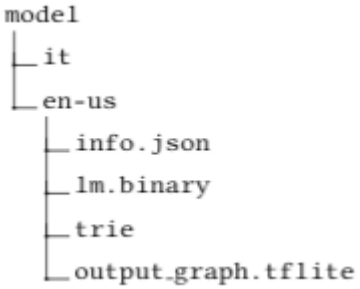

Figure 43: Acoustic and language model

- **output_graph.tflite** is the acoustic model built in tensorflow lite. It provides a lightly version of the model feasible for the mobile uses.
- **lm.binary** is the language model; its aim is to link a probability score to a word sequence.
- **Trie** is the file containing the language vocabulary organized into a tree structure.
- **info.json** is the file containing the hyperparameter of the model language.

PC/Raspberry implementation

The SST in PC/Raspberry version is implemented through the Mozilla Deep Speech network pre-trained with the Common Voice dataset. In such a version, the implementation exploits all the computational capability of the network using the model written in Tensorflow. The model is implemented in Python, by using the Tensorflow library and Scipy to perform the Fourier Transform computation used in the audio feature extraction. The Python script performs the audio segmentation, and, for each segment, it extracts the features and sends them to the LSTM network which provides the textual transcription according to the pre-trained model.

Entity Recognition

Android implementation

The Android version of the entity recognition component is implemented with the OpenNPL library, by using the pre-trained models able to detect the English entities for location, money, organization name, percentage name, person name and time. The trained models are stored in a .bin file, which contains the weights of the entity recognition model. The library is imported through the gradle in the Android project and each model (.bin files) are imported in the assets folder of the project.

PC/Raspberry Implementation

The implementation of the entity recognition component is based on the SpaCy[1] library and uses the "en_core_web_sm", i.e., the small English pipeline trained on written web text (blogs, news, comments) that includes vocabulary, syntax and entities.

3.4.1.10 Preliminary Results

The preliminary results of the Privacy-Aware-Speech recognition analytics are represented in the implementation of an Android application that interacts with the user recording its voice and returning the textual translation with pre-defined sensitive information anonymized. The application is composed by two activities: the Entity Filter activity and the Home activity. The first one is used by the user to set the entities which they want to anonymize from the recording audio (Figure a). The Home activity is used to capture the voice of the user. It is composed of a recording button that, when pressed, makes the application start recording an audio sample (Figure b). At the end of the registration, the audio is passed to the STT (DeepSpeech) component, which translates the voice sample into text, and forwards the result to the Entity Recognition component that detects the entities in the sentence and anonymize them based on the anonymization setting provided by the user through the Entity Filter activity. The final result is displayed in the application screen replacing the defined entities with a set of standard keywords (see *Figure 44*).

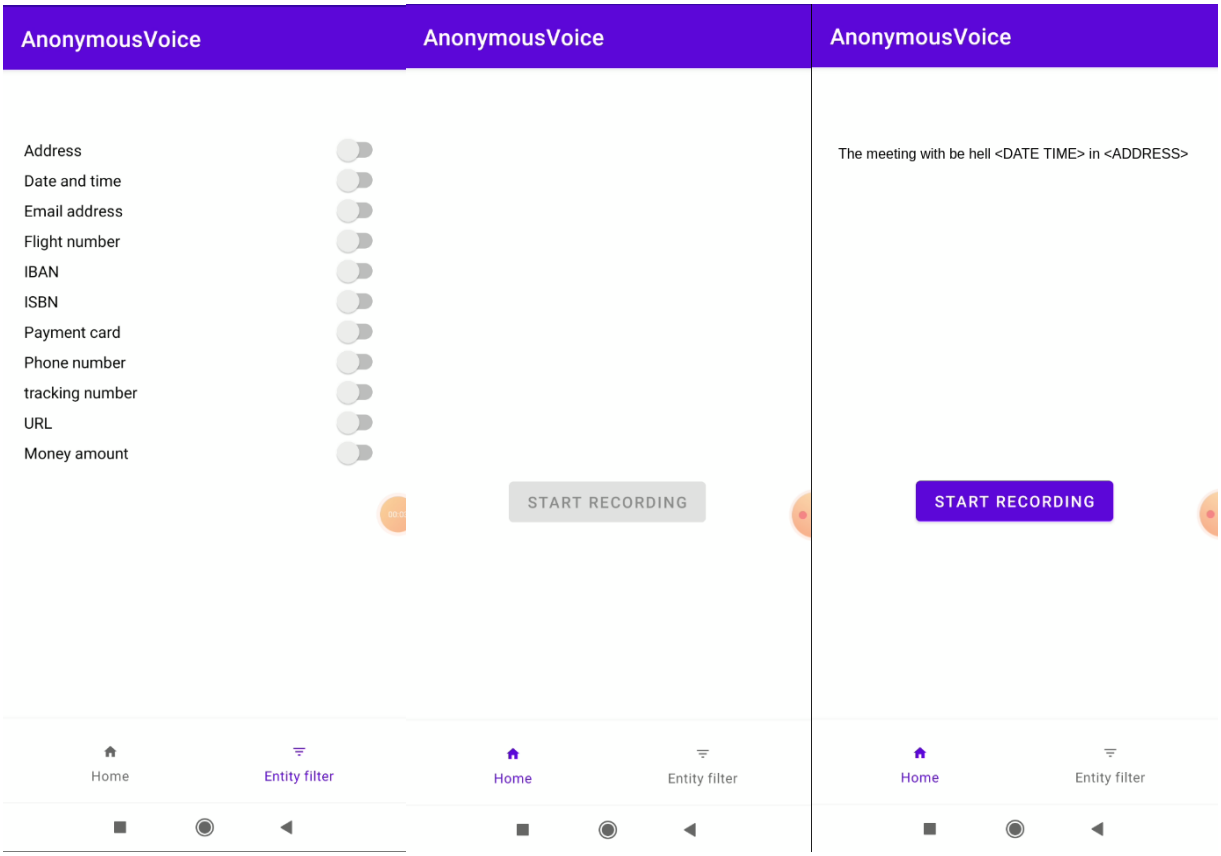

Figure 44: Screenshots of the privacy aware speech recognition app

[1] https://spacy.io/

4 Privacy Preserving Techniques for Analytics

This section presents the research activities of Task 4.4 concerning the mechanisms for preserving privacy of sensitive data and to perform privacy preserving data analysis.
Privacy preserving data analysis includes a set of algorithms able to analyze and gain new knowledge from sets of data, without disclosing sensitive information or minimizing privacy loss. These techniques are, in fact, able to work on anonymized data, i.e., on data where information that uniquely identifies a specific person have been deleted or altered, still maintaining the dataset relevance to the classification problem.
In particular, anonymization techniques alter data in a way that reduces the risk to identify any individual referred to in a specific dataset, by removing, masking or randomizing the pieces of information that can be used to uniquely identify an individual. While this is easily done on textual data, smart home analysis mechanisms have to handle also multimedia data types, such as video, images and sounds. A number of techniques exploiting Differential Privacy and Autoencoders can be used to anonymize and at the same time effectively analyze image-based and sound-based data, in particular concerning physical features identifying people (e.g., faces or voices).
Activities of Task T4.4 are thus focused in implementing solutions to be integrated with the data analysis mechanisms developed in Tasks T4.1 – T4.3 and those related to speech analysis from task T4.4 itself. Moreover, Task T4.4 works on defining mechanisms to optimize the trade-off between privacy gain and data utility, i.e., on instructing the framework about how to configure the analytics and prepare data for them. Furthermore, Task T4.4, together with the activities of Task T1.3 in WP1, defines workflows to identify when it is necessary to enforce privacy and when data can be analyzed without anonymization, e.g., if they shall be analysed in the home premises or shall be sent out to external parties for more complex analysis. The mechanisms for enforcing data privacy will be included in the Data Anonymization Toolbox of the SIFIS-Home framework presented in D1.3.

In the following, we detail the techniques considered in SIFIS-Home that will be used for ensuring data privacy.

4.1 *Generalization*

Generalization techniques enable data analysis, but on a less precise form of data than the original. Generalization mechanisms help to abstract away from sensitive personal data, by replacing dataset attribute values with a more general version of those, thus preserving data utility and preventing re-identification attacks. Generalization is also referred to as “blurring” and can be performed by using several techniques, such as “binning”, and by providing less precise data values. When binning is used, dataset values are replaced with ranges, for instance by replacing dataset age column values with ranges. Table 9 shows an original dataset instance of age, date of birth, and location attributes. Table 10 shows the generalized version of these attributes as an example.

Table 9: Original Dataset Instance

Age	Date of Birth	Location
44	15/8/1978	Pisa

Table 10: Dataset Instance with Generalized Attributes

Age	Date of Birth	Location
40-60	1978	Tuscany

There are different techniques for generalization, such as:

- **Tokenization**: it uses synthetic datasets generation to randomly replace original values with tokens following a specific tokenization pattern that is configurable and keeps the original data format.
- **Redaction**: it enables the masking or removing of all or a part of an attribute value. An example consists of replacing the date of birth field value with the birth year as in table 10, and in hiding part of a credit card number.
- **Pseudonymization**: it replaces attribute values with consistent pseudonyms, meaning that the same pseudonyms are identically applied to the same individuals throughout the whole dataset. Pseudonyms preserve the original dataset structure and format. Table 11 shows an original dataset instance of Name, Account Id, Email, Transaction Value, and Transaction Date attributes. Table 12 shows the pseudonymized version of these attributes as an example.

Table 11: Original Dataset Instances without Pseudonymization

Name	Account ID	Email	Transaction Value	Transaction Date
Sami	AC4481245	sami@gmail.com	45	15/9/2020
Maria	AC1114455	maria@hotmail.com	67	17/6/2020
Tommy	AC1214445	tommy@gmail.com	87	11/11/2020
Sami	AC4481245	sami@gmail.com	34	12/8/2020
Brian	AC4545553	brian@outlook.com	67	15/12/2020
Sami	AC4481245	sami@gmail.com	95	18/6/2020

Table 12: Dataset Instances with Pseudonymization

Name	Account ID	Email	Transaction Value	Transaction Date
DFJFSDF	X321343T	idrshdy@gmail.com	45	15/9/2020
LKGJSHF	C125100C	jfhstey@hotmail.com	67	17/6/2020
LGKKGJD	F454587T	kfjdhsh@gmail.com	87	11/11/2020
FKDHWD	X321343T	idrshdy@gmail.com	34	12/8/2020
FKSJFJD	F454587T	ofhstfj@outlook.com	67	15/12/2020
HSYGJEX	X321343T	idrshdy@gmail.com	95	18/6/2020

4.2 *Data Suppression*

This privacy preserving mechanism is used when generalization is inapplicable and attribute values might cause user re-identification. In data suppression, sensitive attributes or dataset instances are either deleted or replaced with a NULL value. For instance, tabular dataset rows can be either removed or updated with NULL value. There are five kinds of suppression:

- **Attribute suppression**: it involves the removal of a whole part of the dataset as one attribute data or more.

- **Record suppression**: it deletes a whole record or more than one record of the dataset.
- **Value suppression**: it deletes some individual specific values.
- **Cell suppression**: it removes the values of some cells depending on the sensitivity of the cell content.
- **Multidimensional suppression**: it deletes specific records based on some attribute values.

4.3 *Perturbation and Randomization*

Perturbation alters the original data by adding noise to dataset elements using several techniques such as: additive perturbation [MPS99], multiplicative perturbation, matrix multiplicative perturbation [CL08], condensation [AY04], micro aggregation [DM02], and Differential Privacy (DP) [Dwo08]. Due to the privacy guarantees it provides, DP is a powerful privacy preserving technique and it is one of the main privacy preserving methods used in this project.

4.4 *Differential Privacy*

A state-of-the-art and one of the most powerful privacy-preserving mechanisms is differential privacy, which is used to add noise either to the data before the analysis phase or during data analysis based on Laplace or Gaussian distributions, in a way that makes individual data instances indistinguishable when added to or removed from a dataset. This privacy mechanism uses privacy and sensitivity parameters to control the privacy degree applied to the dataset using Equation (1) [Dwo08].

$$\Pr[K(D_1) \in S] \leq \Pr[K(D_2) \in S] \times \exp(\varepsilon)+\alpha \quad (1)$$

Where ε represents the privacy budget, α is the probability of failure, K is the randomized function providing ε-differential privacy for both datasets D1 and D2 which differ in one element only, and S ⊆Range(K). There are three ways to add noise to machine learning models to satisfy the differential privacy property, the first one is to add the noise to the objective function, the second is to add noise to the gradients at each iteration of the training phase, and finally to add noise to the output of the training phase [Dwo08].

4.5 *Autoencoders*

Autoencoders are a data compression mechanism used for dimensionality reduction and feature representation as a latent space. This type of neural networks consists of two parts as shown in Figure 45. The first part is responsible for latent space representation construction using the original input image and is referred to as encoder. The second part is responsible for the reconstruction of the original image from the latent space and is defined as the decoder.
Autoencoders generally result in a loss of utility for the reconstructed image compared to the original image, due to the loss of some features during the compression process. This loss of utility is measured by the distance between the input frame image and the reconstructed frame image. Thus, autoencoder networks are used as a privacy preserving mechanism to protect sensitive data of a dataset. For this reason, even if a third party acquires such data, it should not be able to decode them without the knowledge of the decoder part.

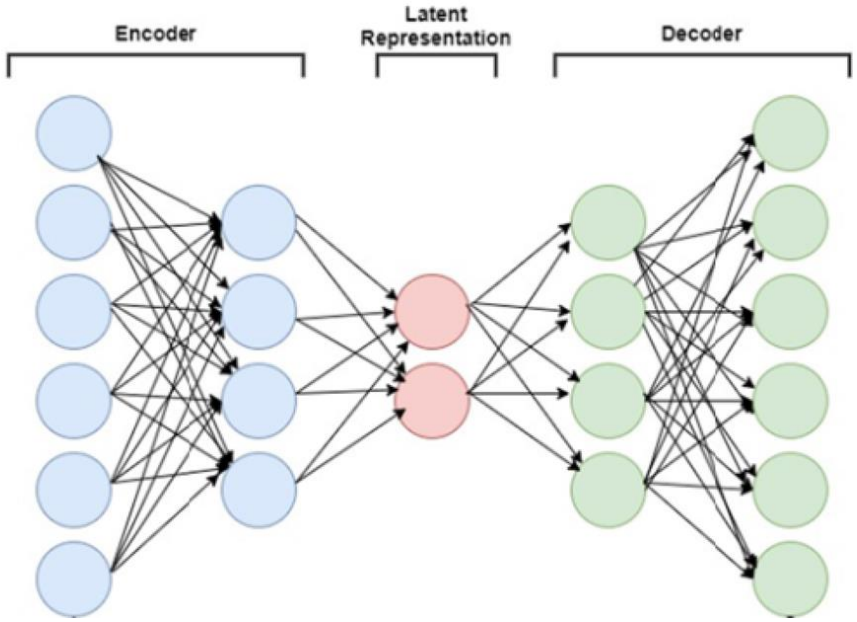

Figure 45 Autoencoder Neural Network Structure

The anonymization degree is controlled by the code size (i.e., the latent representation: the lower the code size, the higher the anonymization). The underlying mechanism can be considered similar to the one used for compression. Autoencoders could be combined with other privacy preserving methods.

4.6 *Trading-off security and privacy with data utility*

With the widespread deployment of smart devices and great advances in data analytics and computer vision fields, privacy and security have become major concerns raising social, ethical, and legal issues. Therefore, adequate implementation of privacy preserving mechanisms has been an active area of research during the last years to help mitigate these concerns.

Privacy preserving mechanisms are used to hide sensitive data and identifying attributes in the datasets that might reveal the identity of a person or other critical information. However, the implementation of these privacy preserving mechanisms might be in contrast with the accuracy of the used data analysis algorithm. Still, privacy preservation is of crucial importance as it is a major requirement of the Trustworthy AI paradigm. Thus, by taking into consideration both privacy and accuracy and by regulating the degree of privacy, it is crucial to attain a solution that provides the best achievable degree of privacy with the least possible harm to accuracy.

The design of the above-mentioned mechanisms into a single organic solution needs to be carefully pondered. This requires to consider all configurations and degrees of privacy to be selected, in order to maximize the degree of privacy gain with the least possible loss in data utility, compared to the original implementation model with no privacy methods applied and akin to a linear optimization problem.

5 Conclusion

This document is the second deliverable from WP4 “Privacy Aware Analytics for Security and Services”, and described the outcome of the WP4 activities carried out during the first half of the project, i.e., up until March 2022. In particular, this deliverable provided a detailed description of a number of analytics designed and developed in the SIFIS-Home project, namely: Device Fault Detection, Device Activity Monitoring in Centralized Cloud, Parental Control, Network Intrusion Detection, Anomaly Detection for IoT Devices in a Consumer Home Network, Policy Enforcement, and Privacy Aware Speech Recognition.

During the second half of the SIFIS-Home project, we will progress the design and development of such analytics, and we will design and implement new ones, consistently with, and in order to address all the requirements described in Deliverable D1.2 "Final Architecture Requirements Report".
Furthermore, all the analytics will be integrated in a specific component of the SIFIS-Home architecture, called Data Analytics Toolbox, which is embedded in the Application Toolboxes module, as shown in Figure 1. This would allow the SIFIS-Home framework to benefit of the results of the analytics for carrying out the smart home protection tasks.

A final description of the privacy aware analytics for security and services designed and developed in WP4 will be provided in deliverable D4.3 "Final Development of Privacy Aware Analytics for Secure Services ". This deliverable will be released in June 2023, and it will update and obsolete the present document, thus acting as final comprehensive description of WP4 activities.

6 References

[SZ14] K. Simonyan and A. Zisserman: Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014

[LDR19] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[Her21] J. L. Hernández-Ramos et al., "Defining the Behavior of IoT Devices Through the MUD Standard: Review, Challenges, and Research Directions," in IEEE Access, vol. 9, pp. 126265-126285, 2021, doi: 10.1109/ACCESS.2021.3111477.

[MSZ21] N. Mazhar, R. Salleh, M. Zeeshan and M. M. Hameed, "Role of Device Identification and Manufacturer Usage Description in IoT Security: A Survey," in IEEE Access, vol. 9, pp. 41757-41786, 2021, doi: 10.1109/ACCESS.2021.3065123.

[NOK21] Nakahara, M., Okui, N., Kobayashi, Y. and Miyake, Y. "Malware Detection for IoT Devices using Automatically Generated White List and Isolation Forest." In Proceedings of the 6th International Conference on Internet of Things, Big Data and Security (IoTBDS 2021), pages 38-47

[HRH18] A. Hamza, D. Ranathunga, H. Habibi Gharakheili, M. Roughan and V. Sivaraman, "Clear as MUD: Generating, Validating, and Applying IoT Behavioural Profiles", ACM Sigcomm Workshop on IoT Security and Privacy (IoT S&P), Budapest, Hungary, Aug 2018.

[MMR17] P. Morgner, C. Müller, M. Ring, B. Eskofier, C. Riess, F. Armknecht, Z. Benenson Privacy, "Implications of Room Climate Data". In: Foley, S., Gollmann, D., Snekkenes, E. (eds) Computer Security – ESORICS 2017. ESORICS 2017. Lecture Notes in Computer Science, vol 10493. Springer, Cham, 2017. https://doi.org/10.1007/978-3-319-66399-9_18

[Dwo08] C. Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer Berlin Heidelberg, 2008.

[MPS99] Muralidhar, K., Parsa, R., & Sarathy, R. (1999). A general additive data perturbation method for database security. management science, 45(10), 1399-1415.

[DM02] Domingo-Ferrer, J., & Mateo-Sanz, J. M. (2002). Practical data-oriented microaggregation for statistical disclosure control. IEEE Transactions on Knowledge and data Engineering, 14(1), 189-201.

[AY04] Aggarwal, C. C., & Yu, P. S. (2004, March). A condensation approach to privacy preserving data mining. In International Conference on Extending Database Technology (pp. 183-199). Springer, Berlin, Heidelberg.

[ZP17] Zhou, C., & Paffenroth, R. C. (2017, August). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 665-674).

[RDS15] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[ILSD] Intel Lab SensorData: http://db.csail.mit.edu/labdata/labdata.html

[Apache] Apache Software Foundation, “OpenNLP Natural Language Processing Library.” 2020

[MUD] MUDgee; https://github.com/ayyoob/mudgee

[OSM] Open-Source MUD manager; https://github.com/osmud/osmud

[XACML13] OASIS. Oasis eXtensible Access Control Markup Language (XACML) version 3, OASIS standard. Technical report, 22 January 2013

[SPOT17] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, Christine Largouët. Anomaly Detection in Streams with Extreme Value Theory. In proceedings of the 23rd ACM SIGKDD International Conference, 2017

[NETSPOT21] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, Christine Largouët. Netspot: a simple Intrusion Detection System with statistical learning. In proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2021

[CL08] Chen, K., & Liu, L. (2008). A survey of multiplicative perturbation for privacy-preserving data mining. In Privacy-Preserving Data Mining (pp. 157-181). Springer, Boston, MA.

[NETSPOT2] https://github.com/Amossys-team/SPOT

[CDM19a] Fulvio Corno, Luigi De Russis, Alberto Monge Roffarello (2019). A High-Level Semantic Approach to End-User Development in the Internet of Things. In: International Journal of Human-Computer Studies (pp. 41-54). Elsevier.

[CDM19b] Fulvio Corno, Luigi De Russis, Alberto Monge Roffarello (2019). Empowering End Users in Debugging Trigger-Action Rules. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (pp. 1-13). ACM, New York, USA.

[BKG20] Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. arXiv preprint arXiv:2003.05991.

[HRB08] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition.

[RTV08] Rothe, R., Timofte, R., & Van Gool, L. (2015). Dex: Deep expectation of apparent age from a single image. In Proceedings of the IEEE international conference on computer vision workshops (pp. 10-15).

[EEH14] Eidinger, E., Enbar, R., & Hassner, T. (2014). Age and gender estimation of unfiltered faces. IEEE Transactions on information forensics and security, 9(12), 2170-2179.

[HCC14] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

[ABD19] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., ... & Weber, G. (2019). Common voice: A massively-multilingual speech corpus. arXiv preprint arXiv:1912.06670.

[MHH21] Montani, I., Honnibal, M., Honnibal, M., Landeghem, S. V., Boyd, A., & Peters, H. (2021). spaCy: Industrial-strength natural language processing in Python.

[Mozilla] Mozilla. Project deepspeech. https://github.com/mozilla/DeepSpeech,2017.

[OWLAPI] The OWL API, http://owlapi.sourceforge.net/

[HERMIT] HermiT OWL Reasoner, http://www.hermit-reasoner.com/

[JGRAPHT] JGraphT - a Java library of graph theory data structures and algorithms, https://jgrapht.org/

[FOAF] FOAF Vocabulary Specification, http://xmlns.com/foaf/spec/

Glossary

Acronym	Definition
ACL	Age Classification Layer
CPN	Colored Petri Net
DFT	Discrete Fourier Transform
DHT	Distributed Hash Table
EVT	Extreme Value Theory
FEL	Face Extraction Layer
FR	Functional Requirements
IDFT	Inverse Discrete Fourier Transform
IQR	InterQuartile Range
JSON	Javascript Object Notation
MUD	Manufacturer Usage Description
NER	Named Entity Extraction
NFR	Non-functional requirement
NSSD	Not So Smart Device
OS	Operative System
OWL	Web Ontology Language
P2P	Peer to Peer
PAP	Policy Administration Point
PSR	Privacy-Aware Speech Recognition
PTP	Policy Translation Point
RNN	Recurrent Neural Networks
SCPN	Semantic Colored Petri Net
SD	Smart Device
SIFIS-Home	Secure Interoperable Full Stack Internet of Things for Smart Home
STT	Speech To Text
UC	Use case
US	User story
XACML	eXtensible Access Control Markup Language