# STAKE: Secure Tracing of Anomalies using previous Knowledge and Extensions

Kevin B. Corrales
Instituto Superior Técnico, Universidade de Lisboa, Portugal

***Abstract*—Internet of Things (IoT) devices have become more present in our households because of an increase in availability and affordability. However, a Smart Home is a challenging environment. It involves people engaged with devices for a variety of purposes, and it is difficult to detect anomalies that can be cyber attacks. In this dissertation we introduce STAKE, a Smart Home gateway for capturing and analysing network traffic, with different levels of detail. The system supports anomaly detection plug-ins to spot attacks in near real-time. We evaluated our system with Machine Learning plug-ins based on the Elliptic Envelope and the Random Forest models. STAKE was able to execute different plug-ins and both detected anomalies.**

**As expected, each model returned different results. Both plug-ins demonstrated promising results when they were created. The Elliptic Envelope model obtained 93,9% accuracy and the Random Forest obtained 96,7%. However, when re-trained in the system the plug-ins did not demonstrate being flexible enough to changes in the Smart Home network. The Elliptic Envelope plug-in demonstrated tendency to produce excessively high false positive rates, which deteriorated the difference between benign and anomaly samples in the training data. The Random Forest plug-in demonstrated a exceedingly tendency for overfitting when re-trained in this kind of environment.**

**Keywords:** Internet of Things, Intrusion Detection System, Anomaly Detection, Machine Learning

## I. Introduction

The Internet of Things (IoT) consists of a network of billions ($10^9$) of interconnected devices, with embedded smart sensors and computational resources, which are connected to the Internet. IoT devices can connect and transfer data over a network without requiring direct human interaction [1]. In many cases this makes the devices autonomous or semi-autonomous. However, IoT devices often have characteristics such as limited network connection, low processing power, low storage resources, reduced dimensions and low energy consumption.

The Smart Home is one of the promises of IoT. A Smart Home refers to a convenient home setup where devices can be remotely controlled, both manually or automatically [2]. The Smart Home is a challenging environment, because it involves people engaged with devices for a variety of purposes that involve real world interactions. Smart Home devices have become more affordable and available, and their presence in consumer households is increasing. The devices can be very diverse, ranging from simple light bulbs, smart plugs and locks, to more powerful devices, like video cameras, digital assistants, tablets and laptops. As a result, the amount of privacy-sensitive data uploaded to the cloud is also increasing. If proper security measures are not taken, these new Internet-connected devices can become entry points for an attacker.

Intrusion Detection System (IDS) is a common approach for network security. An IDS monitors the network and system activities, assesses the integrity of the system and data, recognizes malicious activity patterns, generates reactions to intrusions, and reports the outcome of detection [3]. An important method used by IDS is anomaly detection [4], which consists on finding patterns in data that do not conform to expected behaviour, for example, in Internet traffic [5].

### A. Objectives

In this work, we propose and implement a solution for capturing and storing Smart Home network traffic, and an execution environment for anomaly detection plug-ins. The network traffic is captured in different detail levels: traces, flows and summary features. The captured data is stored in a persistent repository with adequate schema and indexing. The plug-in execution environment allows using Machine Learning models, trained from the captured data, and applied to the near real-time detection of anomalies.

We called our solution STAKE, standing for: Secure Tracing of Anomalies using previous Knowledge and Extensions. It extends SPYKE [6], a previous work that already performs device detection and applies knowledge-based rules, like quota limits per device. We implemented a Supervised Learning plug-in and an Unsupervised Learning plug-in.

## II. Background and Related Work

### A. Network Defence

Cybersecurity solutions can be proactive, like firewalls, and reactive, like IDS (Intrusion Detection Systems) [3].

An *IDS* system monitors and analyses user behaviour and system activities, assesses the integrity of the system and data, recognizes malicious activity patterns, generates reactions to intrusions and reports the outcome of detection [7]. An IDS is composed of several modules: misuse/signature detection, anomaly detection algorithms, hybrid detection, profiling, privacy-preservation data mining, and scan detector [3].

The *misuse/signature detection* module matches malicious patterns with a high detection rate and a low false alarm rate. However, they cannot detect unknown attacks. The *anomaly detection algorithms* build normal patterns in a cyber-infrastructure, such that they can detect the patterns that deviate significantly from the normal model. They can detect new attacks. The *hybrid detection* module is the aggregation

from both mentioned modules above. It normally improves the detection rate and decreases the false alarm rate. The *profiling* module performs clustering algorithms and/or other data mining to group similar network connections and search for dominant behaviours. *Privacy-preservation data mining* focuses in reducing unauthorized access of private information, while retaining the same functions as a normal data-mining method for discovering useful knowledge. The objective of this module is to prevent unauthorized users from accessing private information, such as private data mining or ML results. At last, the *scan detector* module finds vulnerabilities in cyber-infrastructures.

### B. Machine Learning

Machine Learning (ML) is a set of mathematical techniques, implemented on computer systems, to perform information mining, pattern discovery, and inferencing from data [3].

*1) Approaches: Supervised Learning* adopts a knowledge discovery approach, using probabilities of previously observed events to infer the probabilities of new events. On the other hand, *Unsupervised Learning* methods draw abstractions from unlabeled datasets and apply these to new data. Additionally, there is an hybrid type of learning called *Semi-supervised Learning*, an intermediate between Supervised Learning and Unsupervised Learning. In this hybrid type of learning, the model starts with training data that is labeled with the correct answers and then concludes with a model with a tuned set of weights, which is able to predict results for similar data that have not already been labeled.

ML can aid security by performing pattern recognition and anomaly detection [7]. Pattern recognition is usually more adequate for cases in which the threat model is clearly known. For unknown threats, anomaly detection is a more suitable option.

*2) Evaluation Metrics:* Even though Unsupervised Learning models do not use labels to train, when labels are available, they can be used to evaluate this type of models. Labeled data allows us to obtain True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) values that let us calculate the following equations.

In general, using only *accuracy* (as defined in equation 1) to measure model prediction performance is not enough, since it is abstract and only provides an approximate measure of a model performance. The accuracy is a simple way of measuring the effectiveness of your model, but it can be misleading. Therefore, other measures are necessary for us to be able to obtain a concrete understanding of the models behaviour and efficiency. Additionally, accuracy is sensitive to any change in the data set and is mostly effective when data are not balanced.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} , \tag{1}$$

Therefore, it was also be considered other type of evaluation. In an anomaly detection context, to comprehensively evaluate imbalanced learning, especially for minority classification, it is commonly used methods such as [3]: precision, recall, f-score and ROC curve (Receiver Operating Characteristics) plots.

In short, *precision* (defined in equation 2) is the fraction of relevant instances among the retrieved instances. On the other hand, *recall* (equation 3) represents the fraction of the total amount of relevant instances that were actually retrieved. In other words, *precision* measures the degree to which repeated measurements under unchanged conditions show the same results and *recall* measures the proportion of positives that are correctly identified.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} , \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} , \tag{3}$$

Possessing these values, we are able to calculate *f-score* (equation 4), which allows us to measure the accuracy of the test. The f-score represents a weighted harmonic mean of the precision and recall of the test.

$$\text{F-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Furthermore, the ROC curve is generated by plotting the True Positive Rate, also known as recall (equation 3), against the False Positive Rate (equation 5) at various threshold settings.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{Specificity} , \tag{5}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6}$$

The ROC measurement is a probability curve and AUC (Area Under The Curve) represents degree or measure of separability. AUC measurement represents how much model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting.

## III. Related Work

### A. Smart Home Security Monitors

The baseline of this project is SPYKE [6], an open source network intermediary for Smart Home networks, that provides communication monitoring between devices and remote servers, and also the ability to block and limit unwanted connections. With the objective to accept new devices to the Smart Home network, the authentication is performed on SPYKE via WPA2 protocol and using a password, the user is then able to enforce whitelisting-based *user policy* which provides privacy protection. The traffic in SPYKE is filtered via iptables[1], a very widely used firewall implementation engine in the Linux kernel, with rules defined by the user. This solution was shown to handle a large number of devices rules,

[1] iptables: https://linux.die.net/man/8/iptables, accessed on March 12, 2020

which filters outgoing packets with no significant performance degradation.

Furthermore, Bitdefender Box [8] is a commercial security monitor product. Bitdefender Box objective is to secure a Smart Home. In terms of security, it offers vulnerability assessment to detect network security flaws, exploitation prevention to block attempts to exploit vulnerabilities in connected devices, local device security to protect connected devices in place of a locally installed antivirus, as well as anomaly detection, brute force detection, and data protection. This security monitor also provides an application that can be used outside of the house through the Internet.

### B. Machine Learning for NIDS

Recent works on NIDS using ML include Li et al. and Aung et al. *Li et al.* [9] present two datasets collected from programs involving 126 types of vulnerabilities. The authors conducted a comparative study to quantitatively evaluate the impact of different factors on the effectiveness of vulnerability detection, involving more semantic information, imbalanced data processing, and different neural networks. The experimental results show that control dependency can increase the overall effectiveness of vulnerability detection *f-measure* by 20.3%. However, the imbalanced data processing methods were not effective for the dataset created for the study.

*Aung et al.* [10] show the *accuracy* of intrusion detection with the complexity of time when comparing the hybrid algorithm detection method and the single algorithm detection method. The k-Means and Random Tree algorithms were used and each of the methods showed advantages and disadvantages. The experimental results show that the accuracy of the Random Tree algorithm based on k-Means is good in classification of normal and attacks in 10-fold cross-validation but not good in validation.

### C. Smart Home IDS Supervised Learning approaches

*Kamaraj et al.* [11] analysed the potential overhead-savings of ML-based anomaly detection models on the edge. The testbed included three Raspberry Pi 3 boards, which were used as the edge devices, the cloud, and the interface to connect to an energy measurement platform. Then, it was shown how each anomaly detection model performed on each dataset (scenario) with respect to both overhead and anomaly detection accuracy. From the many tested anomaly detection models, the authors asserted that Random Forest, Multilayer Perceptron, and Discriminant Analysis models can viably save time and energy on the edge device during data transmission.

*Hasan et al.* [4] analysed different types of vulnerabilities detection algorithm in IoT sensors. The performance of several ML models to predict attacks and anomalies on the IoT systems was compared accurately. The ML algorithms that were used in this work were LR (Logistic Regression), SVM (Support Vector Machine), DT (Decision Tree), RF (Random Forest), and ANN (Artificial Neural Network). Although these techniques have the same accuracy, other metrics showed that RF performs comparatively better.

### D. Smart Home IDS Unsupervised Learning approaches

SPATIO [12] was proposed as an anomaly detection system designed for the IoT, based on the stream processing approach. The system *accuracy* reached close to 80% detection rate in the best scenario. The fog approach showed advantages in both network load and attack detection latency, in comparison with the centralized approach.

Another recent IoT IDS based on Unsupervised Learning models is IoT-NW (Neighbourhood Watch) [13]. In this solution, each device sniffs packets in the network and performs feature extraction both at packet and flow level, along with the device states and user presence detection, with the objective to detect malicious interactions between them. The result is a statistical model of the expected values for the RMSE (Root-Mean-Square Error) of each Autoencoder. The authors of IoT-NW claim that their system is capable of detection, but with some delay.

### E. Smart Home IDS Hybrid Learning approaches

*Cramer et al.* [14] described an approach for detecting anomalous behaviour of devices by analysing their event data with few or no numeric characteristics. The selected ML models in this paper were the EE (Elliptic Envelope) and the One-Class SVM algorithm. In this work, the EE was trained as Supervised Learning and One-Class SVM as Unsupervised Learning. According to the authors, the main benefits of this approach were: the ability of creating an analysis workflow for specific use cases, which takes much less time in comparison to using a general-purpose data mining tool and it is easy to encode domain-knowledge into the analysis workflow.

*Xiao et al.* [15] presented IoT security solutions based on a combination of ML techniques including Supervised Learning, Unsupervised Learning, and Reinforcement Learning (RL) for authentication, access control, secure offloading, and malware detection schemes to protect data privacy. IoT devices usually have difficulty estimating the network and attack state accurately and have to avoid attacks. A suggested potential solution was Transfer Learning [7]. The authors stated that the intrusion detection schemes based on Unsupervised Learning techniques sometimes have misdetection rates that are non-negligible for IoT systems. Additionally, the Supervised and Unsupervised Learning sometimes failed to detect the attacks due to oversampling, insufficient training data, and bad feature extraction.

## IV. STAKE

In this section we present our proposal, STAKE, that stands for Secure Tracing of Anomalies using previous Knowledge and Extensions. The *tracing* refers to the network traffic capture and the *previous knowledge* refers to rules that can be defined to recognise known attacks and enforce limits on device traffic. The *extensions* refer to the plug-ins that can be installed to detect unknown attacks.

### A. Architecture

The baseline of this project is SPYKE [6], an open source network gateway for Smart Home networks, that provides monitoring between devices and remote servers, and also the ability to block and limit unwanted connections. The gateway runs on a off-the-shelf device[2] between the user IoT devices and the cloud service providers.

The STAKE components are represented in Figure 1. The administrator has access to a web management graphical user interface. The network capturer will intercept traffic of devices, running in batch or stream mode. The captured data will be stored in files and in a database. The plug-in manager is able to handle both supervised and unsupervised learning approaches.

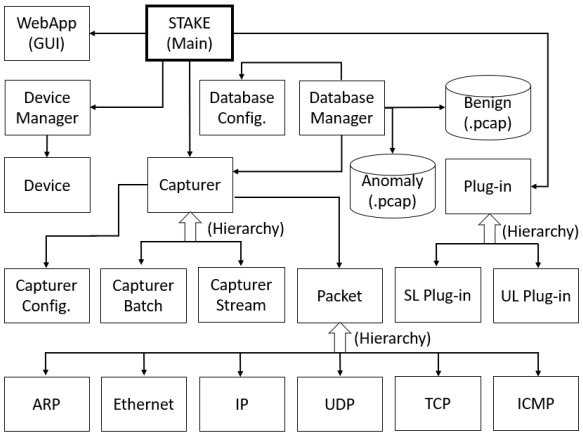


**Fig. 1:** STAKE components.

STAKE operates in four main phases: data collection, data pre-processing, anomaly detection and result/output. The last three phases are executed by plug-ins. Each ML model is trained with data from the captured traffic archives. The goal is to detect device operation anomalies, and to do so in near real-time

### B. Data Collection

In the first phase, STAKE captures incoming and outgoing traffic. This data is stored in a persistent repository with adequate indexing for retrieval.

The capture operation of the STAKE infrastructure is handled by a libpcap[3] tool, known as Scapy[4]. Parsing of the capture file will be needed. The parser extracts statistics from the packets and store the summary of the packets in database. Packets received by the main server are then appended to a file with the pcap extension.

### C. Data Pre-processing

Data pre-processing has a significant impact on the performance of the Supervised Learning models, since unreliable samples probably lead to wrong outputs. One of the challenges is to choose the features that best represent the user or the system behaviour patterns so that anomalous behaviour will be detected, whereas benevolent behaviour will not be wrongly classified as anomalous. The main phase is divided in sub-phases and sub-phases divided into steps.

*1) Feature Extraction:* This sub-phase can be done in two steps: data integration and data cleaning. The data integration combines data from multiple and heterogeneous sources into one database. The second step should only be done when using Supervised Learning algorithms, because removing the noise would consequently remove Unsupervised Learning algorithms purpose to detect anomalies.

Likewise, training a model with a dataset that has a lot of missing values can also drastically impact the ML model quality. The main solutions for missing values are removing any event with missing features or impute the value of the missing feature with mean, median or mode of the column.

*2) Feature Selection:* This sub-phase should only be performed in Unsupervised Learning models when data normalization or data dimensionality reduction is beneficial.

Feature Selection is done in two steps: data selection and data transformation. The selection of features is expected to reduce overfitting probability, improve model prediction *accuracy* and reduce training time [3]. On the other hand, underfitting occurs when the model or the algorithm does not fit the data well enough.

Data selection allows the user to obtain a reduced representation of the data set to keep the integrity of the original data set in a reduced volume. Having a number of features greater than the number of data points will make the ML model overfit. Thus, for better model performance, down-sample of large-scale events may be needed.

Data transformation is when the selected data is transformed into suitable formats [14]. Before data can be processed within ML models, there are certain data transformation steps that must be performed. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

### D. Anomaly Detection

This phase classifies and detects intrusions on a network. In a Smart Home environment, learning the devices patterns and/or behaviours is critical for attack prediction and intrusion detection. These behaviours can explain data and predict patterns [3].

An anomaly detection system is considered optimal when [3]: returns a low count of False Positives and of False Negatives; is easy to configure, tune and maintain; adapts to changing trends in the data; resource-efficient and suitable for real-time application.

In case of using a Supervised Learning model plug-in, it learns from prior data on the first step and makes a prediction about the future behaviour on the second step. On the other hand, an Unsupervised Learning model plug-in is able to perform the anomaly detection in a single step.

[2]Raspberry Pi: https://www.raspberrypi.org/, accessed on October 12, 2020
[3]libpcap: http://www.tcpdump.org/pcap.html, accessed on March 15, 2020
[4]Scapy: https://scapy.readthedocs.io/en/latest/introduction.html, accessed on March 15, 2020

### E. Result/Output

When using various types of ML plug-ins, having a unified location for reporting and alerting can make a difference in the value of security alerts raised. The auditing of alerts raised by an anomaly detection system is important, since it enables the ability to evaluate the system, as well as investigating False Positives and Negatives [7].

## V. Evaluation Methodology

In our evaluation methodology, we have the STAKE performance evaluation metrics, the hardware, the dataset, and the exemplary plug-ins to demonstrate the functionalities of STAKE to be used for evaluation.

### A. STAKE Performance Evaluation Metrics

The metrics chosen to test the system were focused on computational and time performance. We were interested in answering the following questions about our system:

- "How much of the device CPU (%) is used when no plug-ins are training, when training each plug-in individually and when training both plug-ins simultaneously?";
- "How much of the device RAM memory (%) is used when no plug-ins are training, when training each plug-in individually and when training both plug-ins simultaneously?";
- "What is the average delay between the moment that the packet is captured and when it is reported as an anomaly via an alert from each plug-in when no plug-ins are training, when training each plug-in individually and when training both plug-ins simultaneously?";
- "What is the difference between the time duration of the model training when creating each plug-in and the duration of the model training of each plug-in when implemented on the system?";
- "Does the plug-ins have the same scoring results when implemented and re-trained in STAKE system?";

### B. Hardware

We implemented the system on a Raspberry Pi 4 model B, with Raspbian operating system, installed in a 128 GB SD card. This small single-board computer provides a network interface card with 2.4 GHz and 5.0 GHz IEEE 802.11ac WLAN (*wlan0* adapter) protocol and a network interface via cable with Gigabit Ethernet (*eth0* adapter) up to 300 Mbps.

The devices that constitute the network are: a Smart Plug model TP-Link-HS110, a Raspberry pi 4 Model B 4GB with camera model v1 for video streaming, a Raspberry pi 4 Model B 4GB to implement STAKE system, two Android smart phones, one Android tablet, a laptop with Windows 10 Pro operating system and a malicious laptop with Kali 2019.3 operative system to perform attacks in the network.

### C. Dataset

The dataset is one of the critical elements to be used on anomaly detection systems. The network was captured during a period of one week, between 30/06/2020 19:23:04 and 07/07/2020 19:22:04. This captured dataset consists in a total of 1158539 benign samples (labelled as '1') and a total of 93778 anomaly samples (labelled as '-1'). In other terms, from 1252317 captured samples, 92% are benign and 8% are anomalies

This set of anomaly samples contains a variety of network attacks performed by the malicious laptop, such as: port scanning, TCP SYN flooding, ICMP flooding and ARP spoofing. The devices that had their ports scanned were: STAKE, the Raspberry Pi video streamer and the Windows 10 Pro laptop. The SYN and ICMP flooding victim devices were STAKE and Raspberry Pi video streamer. At last, the ARP spoofing attack was performed with the Raspberry Pi video streamer as the victim device.

Regarding the protocols, every protocol is captured and stored in the database. However, STAKE system only considers the following packets: Ethernet, IP, TCP, UDP, ICMP and ARP.

The main approach for feature selection was to only select the packet fields that could give information of an anomalous devices behaviour in an IoT environment, while excluding the packet fields that return same, similar or no useful information for anomaly detection.

The selected features from the Ethernet protocol are: source Ethernet port, destination Ethernet port and Ethernet type field. The selected features from the IP protocol were: source IP address, destination IP address, encapsulated protocol type, header length, 'do not fragment' flag, 'more fragments' flag, 'reserved' flag, , fragment offset value, time to live (TTL) and type of service (ToS). The selected features from the TCP protocol were: sequence number, acknowledgement number, source port, destination port, packet flags, data offset value, urgent pointer flag and window size. The selected features from the UDP protocol were: source port, destination port. The selected features from the ICMP protocol were: type and code (subtype) of the icmp message. The selected features from the ARP protocol were: hardware type, sender hardware address, target hardware address, target protocol address, target protocol address and operation code. At last, the extra selected features were: packet payload, packet length and timestamp.

The features from the captured data that showed high correlation (Pearson correlation coefficient over 0.95) were the ARP features with the IP type field feature and between the TCP fields.

### D. Exemplary Plug-ins

We need to select plug-ins with different characteristics. Thus, we decided to implement an *Elliptic Envelope* based plug-in and a *Random Forest* based plug-in. This plug-in choice was influenced by algorithms chosen by other works in the literature [4], [5], [14], [15]. By observing the machine model selection from the related works, it is noticeable the trend for *Random Forest* algorithm approach in Smart Home environment. On the other hand, even though *Elliptic Envelope* is not that frequent as *Random Forest*, it has shown interesting results.

*1) Plug-in Evaluation Techniques:* Regarding evaluation techniques, we took into consideration the use of cross-validation to evaluate the plug-in. This type of evaluation method is commonly used when the goal of the algorithm is prediction. In this method, the labeled data is divided into $k$ equal parts and it is trained $k$ different models. Afterwards, each model "holds out" a different one of the $k$ parts and trains on the remaining $k$–1 parts. The held-out part is then used for validation. Additionally, grid search was also a considered evaluation technique. Grid search is an exhaustive search tuning technique that attempts to compute the optimum values of hyper-parameters.

*2) Elliptic Envelope:* (EE) is an Unsupervised and Supervised algorithm, where data is distributed across an ellipse, hence the name [16]. In our solution, we decided to develop this model with an Unsupervised Learning approach, because we are interested to observe if the anomalies can be detected in an unlabeled training data setting. This algorithm models the data as a high dimensional Gaussian distribution with possible co-variances between feature dimensions. In other words, it attempts to find a boundary ellipse that contains most of the normal distributed data. Any data outside of the ellipse is considered to be an anomaly. This algorithm should be suitable in anomaly detection problems with the time dimension excluded [7].

The EE algorithm has the advantage of using a robust covariance estimator such as the MCD (Minimum Covariance Determinant), which minimizes the impact of training data outliers on the fitted model. MCD is able to discriminate between outliers and inliers, generating a better fit that results in inliers having small distances and outliers having large distances to the central mode of the fitted model. EE is known to fit reasonably well in a two-dimensional contaminated dataset with a known Gaussian distribution, but not so well on a non-Gaussian dataset [7]. When fitting a high dimensional multivariate Gaussian distribution, the data is structured as an hyper-ellipsoid. This increases the complexity of anomaly detection, since anomalies are detected by a combination of features.

*3) Random Forest:* (RF) classifier is, per definition, a Supervised Learning algorithm. Unsupervised learning methods are mostly preferred over Supervised Learning methods in most cases for anomaly detection [7]. However, we considered that researching a Supervised Learning algorithm behaviour in the Smart Home environment would be interesting to observe.

Even though the RF classifier is known to fit real-world data effectively [7], the algorithm is black-box regarding the decision making processes, meaning that these processes are completely opaque to an external observer. A noticeable strength is that each randomized DT (Decision Tree) that makes up the forest is independently created and can be individually queried for the generation of the final prediction, this conversely makes the classifier strongly scalable [7].

In comparison with the EE algorithm, the RF classifier can be applied on a non-Gaussian anomaly contaminated dataset. However, it must be taken into account that we must avoid using very low-dimensional data with the RF classifier for anomaly detection, since it might not be suitable because of the small number of features on which we can perform splits, which consequently can limit the effectiveness of the algorithm. Furthermore, in contrast to EE, RF can be categorized as an univariate or multivariate anomaly detection technique. In the univariate anomaly detection category, the system looks at each metric by itself, learning its normal patterns and yielding a list of anomalies for each single metric. Often, in univariate algorithms, it is difficult to perform root cause analysis of an issue because it is hard to see the forest for the trees.

*4) Specific Plug-in Evaluation Metrics:* Apart from the general Machine Learning evaluation metrics, specific evaluation metrics for the selected plug-ins were considered. Thus, for EE, it was performed a quality measurement of the plug-in fitted model by calculating the distance between outliers and the model's distribution, using a distance function such as *Mahalanobis* distance. It is an useful metric that measures the distance between a point (vector) and a distribution. This metric has applications in multivariate anomaly detection, classification on highly imbalanced datasets and one-class classification. This distance function transforms the columns into uncorrelated variables, scale the columns to make their variance equal to 1 and, finally, calculates the *Euclidean* distance (equation 7).

$$\text{Euclidian Distance} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{7}$$

In case of the RF plug-in, the MSE (Mean Squared Error) (equation 8) measurement is commonly used, which represents the average of differences between the observed and predicted value.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y}_i)^2 \tag{8}$$

## VI. Results

In this section, we start by discussing the experiments that involves prototype performance evaluation in terms of computational and time resource. Afterwards, we present the Elliptic Envelope and Random Forest plug-ins creation, and the experiments done to optimize the models used.

### A. STAKE

This set of testing involved the designed STAKE system architecture and its developed prototype.

*1) Performance Evaluation:* For the STAKE system evaluations, it was considered that registering only one value would not be a correct evaluation of the system performance. Thus, the evaluation was performed by an average of 10 values during different periods of time while training different combinations of plug-ins. These combinations were: no plug-ins being trained, each plug-in trained individually (Elliptic Envelope and Random Forest respectively) and both plug-ins trained simultaneously.

From the results from table I, we can firstly observe that the processor performance is not significantly affected when training both plug-ins simultaneously. Thus, it is possible to conclude that the processor is not heavily affected in threading two plug-ins simultaneously. On the other hand, RAM memory usage is noticeable increased.

**TABLE I:** STAKE average of 10 tests results performance evaluation. EE = Elliptic Envelope and RF = Random Forest.

| Component Evaluated | Plug-ins being Trained | Average Value | Minimum Value | Maximum Value |
|---|---|---|---|---|
| Processor Usage | None | 18,45% | 0,0% | 28,10% |
| | EE | 38,80% | 25,20% | 51,40% |
| | RF | 33,10% | 25,00% | 50,70% |
| | EE & RF | 39,85% | 27,50% | 51,00% |
| RAM Memory Usage | None | 19,45% | 13,70% | 23,70% |
| | EE | 31,15% | 22,40% | 31,70% |
| | RF | 37,35% | 31,20% | 38,40% |
| | EE & RF | 48,35% | 43,50% | 53,00% |
| EE Anomaly Detection Delay | None | 2,19 s | 1,14 s | 8,02 s |
| | EE | 6,26 s | 5,30 s | 8,16 s |
| | RF | 8,47 s | 3,12 s | 28,36 s |
| | EE & RF | 5,38 s | 4,14 s | 44,15 s |
| RF Anomaly Detection Delay | None | 1,74 s | 1,04 s | 14,34 s |
| | EE | 5,51 s | 4,30 s | 8,49 s |
| | RF | 11,47 s | 2,91 s | 35,10 s |
| | EE & RF | 4,68 s | 3,75 s | 45,14 s |

Regarding the anomaly detection delay results from each plug-in, it was expected that the delay values were lower when no plug-in was training. However, unexpectedly, the average delay when training both plug-in simultaneously was lower than training each plug-in individually. It is suspected that the system and room temperature influenced these last results.

*2) Plug-ins Implementation Evaluation:* The next evaluation performed were related to the Machine Learning model training duration. This comparison was performed between training duration when creating the plug-ins and training duration when the plug-ins are implemented in the STAKE system.

As we can observe from the results in the table II, the training duration from the Machine Learning models when implemented as plug-ins in the STAKE system were longer.

**TABLE II:** STAKE average of 10 tests plug-ins training duration evaluation.

| Plug-ins being Trained | Plug-in creation Training duration | STAKE Plug-in Training duration |
|---|---|---|
| EE | 0:34:41.71 | 0:40:59.27 |
| RF | 1:21:05.20 | 3:14:40.60 |
| EE & RF | 1:26:02.11 | 4:15:13.26 |

The suspected reason for this behaviour is that the values were strongly influenced by the model complexity, system temperature and room temperature. However, we believe that the training duration would not deviate significantly from the obtained results even in optimal temperature conditions.

### *B. Elliptic Envelope Plug-in*

Since Elliptic Envelope does not rely on linear assumptions, it was concluded that correlation features wouldn't cause issues. Additionally, since we had an Unsupervised Learning approach with this model, data cleaning was not performed. The next data engineering consideration was feature selection. The Elliptic Envelope model is known to not work efficiently with continuous data. Therefore, the timestamp feature was excluded for this model. Subsequently, data transformation was considered. Given that the Elliptic Envelope model assumes that the training data is Gaussian, the training data had to be transformed into Gaussian values and normalized beforehand.

*1) Hyper-parameters exploration:* The main hyper-parameter of this model, which was selected for optimization, is the *contamination* percentage. The *contamination* parameter indicates the percentage of anomalies found in the training data. Regarding the others hyper-parameters, it was considered that no change would be needed. Hence, the default values for these hyper-parameters were used. Intuitively, it made sense to use the *contamination* percentage equal to the amount of anomaly samples in the data. The first test was performed with a *contamination* percentage value equal to the anomaly percentage of the training data. However, by increasing the anomaly percentage by 5% in relation to the *contamination* percentage, the accuracy increased by 10%. Thus, it was considered a 5% margin for the *contamination* hyper-parameter percentage in relation to the real anomaly data percentage.

*2) Initial Sampling Size Evaluation:* Different amount of samples were used to test the model behaviour, ranging from 10008 to 62381, with 20% *contamination* hyper-parameter percentage and training data anomaly percentage between 20 and 25%. The $n$ samples used for training, were the $n$ last captured packets from the dataset. From the obtained results, we could state that using samples between 31155 and 43581, with 20% *contamination* and 25% anomaly percentage was the best option. However, we must also consider that the 18758 showed a more realistic behaviour in the relation between the *contamination* hyper-parameter percentage and anomaly percentage of the samples (20% used for both percentages). Therefore, it was decided to evaluate the model behaviour in regard to the *contamination* hyper-parameter percentage and anomaly percentage of the samples with 18758, 31155 and 43581 samples.

*3) Hyper-parameter and Anomaly Percentage Optimization:* After finding the range of possible optimal number of samples (18758, 31155 and 43581), we then proceeded to test the model according to different *contamination* hyper-parameter percentage and anomaly percentage, in order to further optimize the model. The *contamination* hyper-parameter percentage and anomaly percentage used for testing ranged from 5% to 45%. Anomaly percentages higher than 50% was not considered, because the objective of the system is to detect anomalies in the devices behaviour and not detect the specific captured anomalies used in the model training. The model demonstrated better results when using lower percentage of anomalies and *contamination* hyper-parameter percentage. This behaviour is logical because, the lower the *contamination* hyper-parameter percentage, the smaller the

generated ellipse will be. Thus, when high *contamination* hyper-parameter percentages are used, the risk of having high amount of benign packets identified as anomalies will be increasingly higher.

The tests using 5% *contamination* percentage with 18758, 31155 and 43581 samples showed good accuracy results. However, these tests demonstrated excessively high false positive rate. It was concluded that further model optimization was required. Therefore, it was decided to evaluate the model again according to the different number of samples, but this time using 5% *contamination* hyper-parameter percentage on all tests. From the 5% *contamination* hyper-parameter percentage sampling testing, we were able observe that the model has a tendency of having high false positive rates. High false positive rates are common in Unsupervised Learning algorithms. However, false positive rate should be minimized, since if it is excessively high, the administrator is forced to investigate each packet, which removes the point of the system.

*4) Extra Optimization:* The first considered solution was to reduce the number of used features. It was suspected that the Elliptic Envelope algorithm was struggling in generating an efficient hyper-ellipsoid for anomaly detection. However, reducing the number of features has the risk of excluding the possibility of detecting anomalies from those specific removed features. Therefore, it was considered the use of PCA (Principal Component Analysis) as alternative. PCA is a dimensionality-reduction method that is used to reduce the dimensionality of large data sets, while retaining most of information from the large set.

The use of PCA to transform the training data into 2 dimensions demonstrated benefits in reducing the model training duration and reducing the high positive rate between 5%-15%. The Elliptic Envelope algorithm has the inherent problem of result inconsistency when using data with more than 2 features. However, when transforming the data with PCA into 2 dimensions, the results became reproducible. An additional PCA benefit, is that the PCA simplifies the representation (figure 2) of the generated ellipse. The representation of the generated hyper-ellipsoid was not previously possible because of the data high dimensionality.

From the plots from the figure 2, we can observe that the difference between benign and anomaly are too ambiguous for the model to behave efficiently. The benign and anomalous samples are too similar for the algorithm to generate an efficient ellipse that separates both classes. Therefore, it was considered that the Elliptic Envelope algorithm is not able to fit well in this specific environment.

### C. Random Forest Plug-in

The dataset featurizes each of the protocols. However, this come at the cost of having missing values. For example, UDP features will be empty in case of featurizing a TCP packet. For that reason, instead of deleting instances with missing data, the missing data is filled with zeros.

Another consideration is the data balancing. Random Forest is a tree-based model, which uses information gain/gini

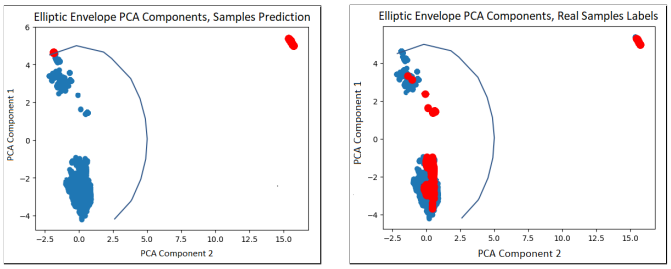


**Fig. 2:** Elliptic Envelope ellipse generation representation with 43581 samples, 5% *contamination* hyper-parameter and 5% anomaly percentage, using PCA 2 dimensional transformation. The blue dots represent benign samples and the red dots represent anomalous samples.

coefficient inherently, hence does not require feature scaling. The next consideration was feature selection. Continuous data makes the Random Forest algorithm generalize poorly, as a consequence, it overfits the model citeBOOK-DT-ML-CS. Therefore, the timestamp feature will also be excluded for this model.

The Random Forest model requires the training data to be numerical. Hence, conversion of the categorical data to numerical was performed with Label Encoding.

*1) Hyper-parameters exploration:* Before analysing the model behaviour according to the amount of samples used and before comparing both data transformation methods, model hyper-parameters should be explored beforehand. The hyper-parameters available for the Random Forest model are: *n_estimators*, *criterion*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *min_weight_fraction_leaf*, *max_features*, *min_impurity_decrease*, *min_impurity_split*, *bootstrap*, *oob_score*, *n_jobs*, *random_state*, *warm_start*, *class_weight*, *ccp_alpha* and *max_samples*. Taking into account the extensive set of hyper-parameters, hyper-parameter selection was recommended. The hyper-parameter selection criteria was based on parameter importance and how much it influences the model.

The selected hyper-parameters for model optimization are: The *n_estimators*, which indicates the number of trees in the forest. Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably; The *max_depth* parameter defines the max number of levels in each decision tree. The deeper the tree, the more splits it has and it captures more information about the data; The *min_samples_split* equals the minimum number of samples required to split an internal node. When we increase this parameter, each tree in the forest becomes more constrained as it has to consider more samples at each node; The *min_samples_leaf* parameter is similar to *min_samples_split*, however, this describe the minimum number of samples at the leafs, the base of the tree; The *max_features* specifies the number of features to consider when looking for the best split, it can be defined by a float or integer numerical value, a square root function or a logarithmic function; The *bootstrap* denotes the method for sampling

data points (with or without replacement). If *bootstrap* is set to "false", the whole dataset is used to build each tree. If *bootstrap* is set to "true", it means that some samples will be used multiple times in a single tree. The rest of the hyper-parameters are set with default values.

*2) Hyper-parameter Behaviour Analysis:* Each of the hyper-parameters was then evaluated with hyper-parameter roc plots using a set of different values. This hyper-parameter evaluation was performed with samples ranging from 8753 to 99881, all with 25% of anomalies and 33% split for test data. The $n$ samples used for training, were the $n$ last captured packets from the dataset. By analysing these plots, we were able to discover the best set of values to avoid both under and overfitting. The amount of samples that showed promising results from the plots were 43581 and 62381.

The range of values inside the dark green dashed boxes from each hyper-parameters evaluation plots (figures 3, 4 and 5) represents the ideal range selection for model complexity. For the grid search, with the objective to optimize the model and find the best hyper-parameter value combination while avoiding underfitting and overfitting. The only static hyper-parameter from the grid is *bootstrap*, this parameter is set as "false" because we desire to use all the samples in the data and avoid missing detecting any anomaly.

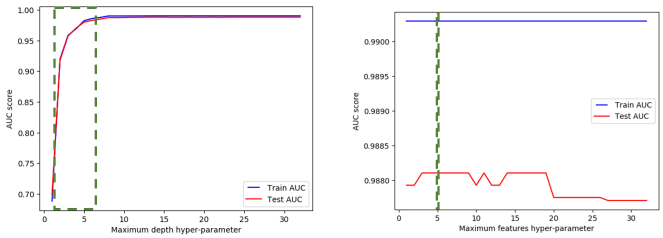


**Fig. 3:** Random Forest *max_depth* and *max_features* hyper-parameters evaluation with 43581 samples.

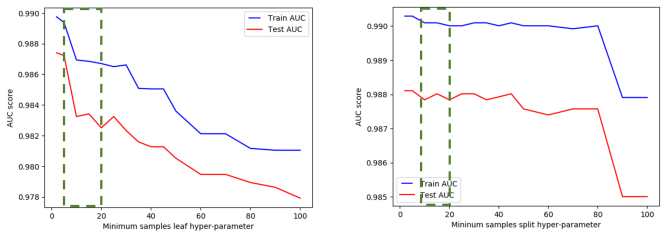


**Fig. 4:** Random Forest *min_samples_leaf* and *min_samples_split* hyper-parameters evaluation with 43581 samples.

*3) Anomaly Percentage Optimization:* The selected amount of samples for grid search and anomaly percentage analysis was 43581 and 62381, using 33% split for test data and 5 fold cross validation. Similar to the Elliptic Envelope model, anomaly percentage higher than 50% was not considered because the objective of the system is to detect all types of anomalies in the devices behaviour and not detect the specific captured anomalies used in the training data. However, every results did overfit using grid search, achieving accuracies over

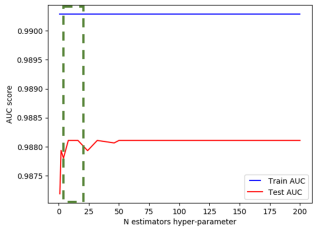


**Fig. 5:** Random Forest *n_estimators* hyper-parameter evaluation with 43581 samples.

99%. It is noticeable that the Random Forest model overfits for large depth values. The trees perfectly predicts all of the training data, however, it fails to generalize the findings for new data. Therefore, it is required to control the complexity of the trees in the forest, or even prune when they grow too much. Thus, from the grid search results, it was decided to use static parameters to avoid overfitting and generalize the findings for new data as much as possible. The chosen hyper-parametrization for the Random Forest were: *bootstrap*=False, *max_depth*=2, *max_features*=10, *min_samples_leaf*=10, *min_samples_split*=20 and *n_estimators*=20.

The results with the chosen hyper-parametrization demonstrated highest accuracy using 62381 samples with 25%, 30% and 35% anomaly percentages. In addition, the test results with 43581 samples with 40% and 45% anomaly percentages also demonstrated high accuracy. However, these tests showed hyper-parametrization overfitting. Therefore, these configurations settings were excluded for Random Forest plug-in creation.

Furthermore, tests ID showed that using anomaly percentages below 25% underfitted the model. This underfitting was observable both on the evaluation scoring metrics. Therefore, they were also excluded.

The remaining tests are the configuration settings with 43581 samples and anomaly percentages between 25% and 35%. Since all of them showed similar results, further score evaluation was required. Even though all these previous model configurations could be used for the Random Forest plug-in creation, the configuration with 30% anomaly percentage was selected because it has a middle ground anomaly percentage, which theoretically increases the probability of data flexibility in case the network changes its behaviour.

### *D. Plug-ins Re-training Evaluation*

The last evaluation was the plug-ins training result comparison between plug-in creation results and plug-in implemented in STAKE system results. Since the system is capturing in real-time and changes in the data are expected, the only difference between these two situations will be the data used for training. The data used to train each plug-in in the STAKE system was: the original training data, plus one day of capture while the plug-in anomaly detection was active. This testing was performed individually, for each plug-in, with the objective to avoid that one plug-in would affect the other final results. All

the other procedures performed and configurations sets used were exactly the same as the ones used during plug-in creation.

**TABLE III:** Elliptic Envelope plug-in creation training and STAKE implementation training results comparison.

| Test ID | Accuracy | Recall | Precision | F-score | False Positive Rate |
|---|---|---|---|---|---|
| Creation | 93,9% | 96,8% | 96,8% | 96,8% | 60,8% |
| STAKE | 90% | 94,7% | 94,7% | 94,7% | 100% |

**TABLE IV:** Random Forest plug-in creation training and STAKE implementation training results comparison.

| Test ID | Accuracy | Recall | Precision | F-score | False Positive Rate |
|---|---|---|---|---|---|
| Creation | 96,7% | 98,6% | 97,4% | 98% | 10,9% |
| STAKE | 100% | 100% | 100% | 100% | 0% |

Both Elliptic Envelope and Random Forest plug-in training results worsen when re-trained in STAKE system. Even though it was expected for the Elliptic Envelope to have inconsistent results with data changes, it was unexpected for the Random Forest to overfit because training process was exactly the same and this model is known to obtain consistent results. It was evident that the plug-ins were overly optimized for the specific trained data and not configured for flexibility.

## VII. Conclusion

In this work, we developed STAKE, a gateway to deploy in a smart home network between the smart devices and the Internet. The system, demonstrated flexibility in implementing different types of plug-ins. Even though some adjustment was required for each plug-in, this could be easily and quickly performed in the user interface.

In addition, the hardware used for evaluation demonstrated being efficient in threading the local HTTP server hosting, data collection (packet capturing), plug-ins training and plug-ins anomaly detection processes. The hardware evaluation results shows that the CPU is able to thread all the processes, along with multiple plug-ins, without being strongly affected by it. The system demonstrated being capable of announcing anomalies in a timely manner, even when training two plug-ins simultaneously. The most affected hardware element, even though moderately, was the RAM memory.

Regarding the plug-ins re-training in the system, the results were disappointing. On the contrary from the plug-in creation results, when the plug-ins were implemented in the system, each of the plug-ins behaviour and results deteriorated. Due to the high false positive rate in the Elliptic Envelope plug-in, we suspect that the model will affect the system negatively. The difference between benign and anomaly will be less clear over time. Thus, the model performance will deprecate significantly on the following trainings.

The results of the Random Forest model during plug-in showed potential in anomaly detection efficiency. The biggest perceived challenge with this model was to avoid overfitting. However, this was expected because it is a common problem with this model. For this model to work the most efficiently possible, there must be a clear dissimilarity between benign and anomalous behaviour. This implies that the Random Forest plug-in will work very efficiently if the devices behaviour does not change frequently and if the dataset is not heavily affected by false positive and/or false negatives.

Hence, for good re-training results, a plug-in with a flexible and reproducible model or integrating human feedback will need to be considered for future work. A human feedback can be used to fine tune the global comprehension of the anomalies, while preserving the efficiency of the system and plug-ins [3].

## References


[1] C. Hosmer, *Defending IoT Infrastructures with the Raspberry Pi*. Apress, 2018.

[2] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home IoT privacy." https://dl.acm.org/doi/pdf/10.1145/3274469?download=false: Princeton University, USA, Nov. 2018.

[3] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. Taylor and Francis Group, LLC, 2011.

[4] M. Hasan, M. Islam, I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches." https://www.sciencedirect.com/science/article/pii/S2542660519300241: Internet of Things Research Lab, Department of Computer Science and Engineering, Mar. 2019.

[5] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices." https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8424629: Department of Computer Science Princeton University Princeton, New Jersey, USA, 2018.

[6] S. Wang, M. L. Pardal, and R. Claro, "Spyke: Security proxy with knowledge-based intrusion prevention." http://web.tecnico.ulisboa.pt/~miguel.pardal/www/pubs/2019_Wang_Pardal_INForum_SPYKE.pdf: Instituto Superior Técnico, Universidade de Lisboa, Dec. 2019.

[7] D. F. Clarence Chio, *Machine Learning and Security*. O'Reilly Media, Inc., 2018.

[8] D.-R. Berte, "Defining iot," 05 2018.

[9] Z. Li, D. Zou, J. Tang, Z. Zhang, M. Sun, and H. Jin, "A comparative study of deep learning-based vulnerability detection system." https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8769937: National Key Research and Development (R&D) Plan of China, Jul. 2019.

[10] Y. Aung and M. Min, "Hybrid intrusion detection system using k-means and random tree algorithms." https://ieeexplore.ieee.org/abstract/document/8441094: University of Computer Studies, Mandalay Mandalay, Myanmar, Jun. 2018.

[11] K. Kamaraj*, B. Dezfouliy, and Y. Liuz, "Edge mining on IoT devices using anomaly detection." http://www.apsipa.org/proceedings/2019/pdfs/295.pdf: Internet of Things Research Lab, Department of Computer Science and Engineering, Nov. 2019.

[12] G. de Oliveira and V. Mouta, "Spatio: end-user protection against IoT intrusions." Instituto Superior Técnico, Universidade de Lisboa, Oct. 2019.

[13] P. E. Carmo and M. L. Pardal, "IoT neighborhood watch: device monitoring for anomaly detection." http://web.tecnico.ulisboa.pt/~miguel.pardal/www/pubs/2019_Carmo_Pardal_INForum_NWatch.pdf: Instituto Superior Técnico, Universidade de Lisboa, Dec. 2019.

[14] I. Cramer, P. Govindarajan, M. Martin, A. Savinov, A. Shekhawat, A. Staerk, and A. Thirugnana, "Detecting anomalies in device event data in the IoT." https://pdfs.semanticscholar.org/7e3c/9790f6ff6fe2fcbe691ca21a5113a4c25814.pdf: Bosch Software Innovations GmbH, IoT Analytics, Mar. 2019.

[15] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use ai to enhance security?" https://ieeexplore.ieee.org/abstract/document/8454402: University of Computer Studies, Mandalay Mandalay, Myanmar, 2018.

[16] A. Ekholm and A. Avdic, "Anomaly detection in an e-transaction system using data driven machine learning models," http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1335216&dswid=-7153, 07 2019.