Differentiating Attacks and Faults in Energy Aware Smart Home System using Supervised Machine Learning

Georgios Tertytchny Nicolas Nicolaou Maria K. Michael
{tertytchny.georgios,nicolaou.c.nicolas,mmichael}@ucy.ac.cy
Electrical & Computer Engineering Department, University Of Cyprus
KIOS Research and Innovation Center Of Excellence, University Of Cyprus

ABSTRACT

The topics of fault diagnosis and security attack diagnosis in Cyber Physical Systems (CPS) have been studied extensively in a stand-alone manner. However, considering the co-existence of both of these sources of abnormality in a system, and being able to distinguish among them, is an important and timely problem not currently addressed in the literature. In this paper, we study the internal communication environment of an Energy Aware Smart Home (EASH) system. We formally define the problem of differentiating component attacks from component failures in EASH and we provide a methodology based on supervised Machine Learning (ML) algorithms to differentiate between a set of common attacks and faults on the communication channel. To evaluate our approach, we provide experimental results obtained by a simulation framework as well as from a real-time testbed environment.

CCS CONCEPTS

• Security and privacy → Network security; • Networks → Network protocols; • Computing methodologies → Machine learning; • Hardware → Fault models and test metrics.

KEYWORDS

Cyber Security, Fault Diagnosis, Cyber Physical Systems, Machine Learning, Energy Aware Smart Home, Internet Of Things (IoT)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:

Georgios Tertytchny Nicolas Nicolaou Maria K. Michael. 2019. Differentiating Attacks and Faults in Energy Aware Smart Home System using Supervised Machine Learning. In INTERNATIONAL CONFERENCE ON OMNI-LAYER INTELLIGENT SYSTEMS (COINS), May 5–7, 2019, Crete, Greece. , 6 pages. https://doi.org/10.1145/3312614.3312642

1 INTRODUCTION

Cyber Physical Systems (CPS) integrate physical processes with electronic computing devices and digital communication channels. There are two main causes that may affect the proper operation of a CPS: security attacks, and failures due to component malfunctions or deficiencies. As CPS play a vital role for the day-to-day operations of modern societies, they have become an appealing target of attacks by malicious actors [4], while their widespread use lead to a significant increase in the attack surface. At the same time, just like any other physical control system, different components of CPS can suffer benign failures [8].

Both failures and attacks lead to abnormal behaviour of the system, but their implications may differ greatly: a component failure may be controlled leading the system to a stable state until the component returns to an normal operating state; a component attack by a malicious actor may be designed to trigger a chain of possibly unnecessary control actions, causing severe outcomes to the system. Having the means of differentiating between faults and attacks allows CPS operators to choose the proper recovery actions that minimize the negative effects of abnormal behavior.

Specifying the parameters that may lead to such differentiation is not an easy task. In this work, we focus on an Energy Aware Smart Home (EASH) system. In an EASH system, sensor/computing nodes are connected and measure the (instant) energy consumption of home electronic devices/appliances. Those measurements are transmitted to a central entity (coordinator), via common wireless/wired protocols, that stores and analyses the overall energy consumption in the house. From the central coordinator the energy consumption is then reported to the energy utility by
using the Advanced Metering Infrastructure (AMI), which enables a two-way communication between utility consumers (Smart Homes) and utilities (Smart Grids). This allows consumers to be aware of their home’s energy footprint (energy aware), and make decisions on how to minimize their energy consumption. We examine how network’s traffic characteristics in an EASH may relate to a family of failures and attacks, and how those characteristics may be used by algorithms with small information requirements to distinguish between the different abnormalities.

Traditional solutions for fault diagnosis in CPS are based on operator’s experience [3], while more recent approaches [6] utilize sensor and alarm data, characterizing the new era of Internet of Things (IoT). IoT solutions for fault diagnosis combine both machine learning approaches and human expertise. For instance, fault diagnosis in power and smart grid systems utilize artificial neural network which are adaptive systems inspired by biological systems. The most common techniques under the artificial neural networks are Radial Basis Function (RBF) [7] and Support Vector Machines (SVM) [11].

Apart from fault diagnosis, security concerns for CPS have been studied in the area of systems fault detection, isolation, and recovery [5], where security attacks are modeled as extreme fault cases. Some unique characteristics and vulnerabilities of CPS lead to the need of new, more appropriate, detection and identification techniques. Moreover, as mentioned in [2], traditional IT security methods are not applicable in the case of CPS security. More specifically, the resilience on communication protocols for the transmission of measurements and control packets increases the possibility of attacks that can affect the physical plants [4]. Moreover, existing information security methods as for example authentication, access control, information integrity do not suffice for the protection of CPS.

Research in cyber security and fault diagnosis, affecting EASH as part of the Smart Grid, is mainly focused on detection mechanisms [9, 12]. Such mechanisms are able to identify when a system’s behaviour deviates from normal, but is unable to identify the source of abnormality. The normal behaviour of the system can be described either using a state estimator used in Anomaly Detection [12], or historical data used in intrusion detection mechanisms. But while existing studies deal with the detection and identification of attacks in the Smart Home [10], there is no work that takes into account both malicious attacks and faults on communication channel.

In this paper we study whether it is possible to differentiate component faults, which alter the system communication, from network attacks by examining the traffic on the communication channel. We provide a formal definition for the problem under consideration and propose a general methodology which can be utilized for differentiating a common set of attacks and faults. The proposed methodology uses supervised Machine Learning (ML) algorithms, which utilizes data generated during communication. Our proposed approach has been evaluated through simulation and a real time use case.

2 PROBLEM FORMULATION

We consider an EASH system which comprises of multiple monitoring computing devices (nodes), each communicating directly to a central hub (coordinator node). Our goal is to design a methodology to differentiate faults from attacks in such system. More formally, we model the system as a graph \( G = (V,E) \) composed of a set \( V \) of \( m \) processing nodes \( \{q_1, \ldots, q_m\} \) and a coordinator node \( q_c \). Each \( q_i \in V \setminus \{q_c\} \) processing node reports its messages directly to the coordinator node. Thus, a star topology exists where \((q_i, q_c) \in E, \forall q_i \in \{q_1, \ldots, q_m\} \).

Two nodes \( q_i, q_j \in V \) communicate by exchanging messages via synchronous, reliable, communication channels if \((q_i, q_j) \in E \). We use \( e_{i,j} \) as a shorthand to denote the channel between \( q_i \) and \( q_j \), given that \((q_i, q_j) \in E \).

For each \( q_i \in \{q_1, \ldots, q_m\} \), we define a single binary connectivity variable \( v \) to indicate the direct connection of \( q_i \) to the coordinator:

\[
q_i.v = \begin{cases} 
 1 & \text{if } (q_i, q_c) \in E \\
 0 & \text{otherwise}
\end{cases}
\]

The state of each node \( q_i \in V \), denoted as \( \sigma_i \), is defined over a set of local state variables \( B_i \) and the binary connectivity variable \( q_i.v \). Similarly, the state of each channel \( e_{i,j} \), is defined over a set of channel measurements \( C_{i,j} \), that describe the channel’s conditions and performance metrics (e.g., delay). The state of the system \( \sigma \) is a vector that contains the state of every node \( q_i \in V \) and every channel \( e_{i,j} \) for \((q_i, q_j) \in E \).

Each node \( q_i \in V \) implements a set of actions. When an action \( a \) occurs at a node \( q_i \), it causes the state of \( q_i \), and thus the state of the system, to change. An execution \( \xi \) of the system is an alternative sequence of states and actions, where actions send and receive events that may change the state of the channel or the state of a node in the system. We assume that the system executes in discrete time units. Thus, \( \sigma_0 \) is the initial state of the system at time \( t = 0 \), and \( \sigma_t \) denotes the state of the system at time \( t \) in an execution \( \xi \). We assume that each execution starts with the initial state and ends with a state. If \( \sigma_t \) is the ending state of an execution \( \xi \) then we say that \( t \) is the length of \( \xi \) and is denoted by \( T(\xi) = t \).

Let \((q_i,b_j)\) denote the value of the local variable \( b_j \) of node \( q_i \) at a time unit \( t \). Similarly, we denote by \((q_i,v)\), the
binary connectivity variable of node \( q_i \) and \( (e_{i,j},m_k) \), the value of the channel characteristic \( m_k \) at time \( t \). Given this notation we can define the states of each node, each channel, and the state of the system as a whole at time \( t \):

\[
\sigma^i_t = \{(q_i,b_j) : b_j \in B_i\} \bigcup \{(q_i,v_t)\} \tag{1}
\]

\[
\sigma^{e_{i,j}}_t = \{(e_{i,j},m_k) : m_k \in C_{i,j}\} \tag{2}
\]

\[
\sigma_t = \{\sigma^i_t : (q_i,v_t) \in E\} \bigcup \{\sigma^j_t : q_i \in V\} \tag{3}
\]

For each state variable \( q_i, b_j \in B_i \), we denote by \( L(q_i,b_j) \) and \( U(q_i,b_j) \), the expected low and upper bounds respectively, of the value of \( q_i, b_j \), at a time unit \( t \) with respect to the previous state of the system \( \sigma_{t-1} \). Using, these bounds we can now define normality and abnormality in the system as follows:

**Definition 2.1.** A state \( \sigma^i_t \) of node \( q_i \) in an execution \( \xi \) at time \( t \) is normal iff:

\[
\forall j[(q_i,b_j) \in B_i, (q_i,b_j)_t \in [L(q_i,b_j)_t, U(q_i,b_j)_t]] 
\land (q_i,v)_t = 1
\]

In other words a state of a node is normal if all the state variables are within the expected bounds, and the node is connected and communicating with the central node (i.e., \( (q_i,v)_t = 1 \)). On the contrary in an abnormal state such conditions may not hold as explained in Definition 2.2.

**Definition 2.2.** A state \( \sigma^i_t \) of node \( q_i \) in an execution \( \xi \) is abnormal at time \( t \) if at least one of the following holds:

- **abnormal node behavior** if \( \exists j \) \( q_i, b_j \in B_i \) such that:
  \[
  (q_i,b_j)_t < L(q_i,b_j)_t \lor (q_i,b_j)_t > U(q_i,b_j)_t
  \]

- **abnormal node connectivity** iff:
  \[
  (q_i,v)_t = 0
  \]

By Definition 2.2, an abnormal state may include variables with unexpected values, and/or the direct connection of the node with the coordinator may be interrupted (i.e., \( (q_i,v)_t = 0 \)). We assume that connectivity interruption may be caused by an external event (e.g., man-in-the-middle attack, or interruption attack) without necessarily the sending node be aware of this.

Following Definitions 2.1 and 2.2 we say that the system state \( \sigma_t \) is normal if \( \forall q_i \in V, \sigma^i_t \) is similarly. We characterize a system state \( \sigma_t \) as abnormal if \( \exists q_i \) such that \( \sigma^i_t \) is either an abnormal node behavior or abnormal node connectivity. We define an execution \( \xi \) as Normal-execution if \( \forall \sigma_t \in \xi, \sigma_t \) is normal, for \( 0 \leq t \leq T(\xi) \). Similarly we say that an execution \( \xi \) is \( \psi \)-execution, for \( \psi \in [\text{Fault, Attack}] \), if \( \exists \sigma_t \in \xi \) such that \( \sigma_t \) is an abnormal state, resulting from a fault or attack respectively, for \( 0 \leq t \leq T(\xi) \).

**Definition 2.3 (Class State Snapshot Vector).** Class state snapshot, denoted as \( w^\psi_t \), is the system state \( \sigma_t \), of class \( \psi \), in an execution \( \xi \), time \( t \leq T(\xi) \), \( \psi \in [\text{Normal, Attack, Fault}] \).

So given an algorithm \( A \), we can define the differentiation operator, and thus the differentiation problem, as follows.

**Problem 2.1 (Class Differentiation).** Specify a differentiation operator \( \bigoplus_A \) using a (machine learning) algorithm \( A \) such that, given two class state snapshot vectors \( w^\psi_t, w^{\psi'}_t \) from executions \( \xi \) and \( \xi' \) (not necessarily different):

\[
\bigoplus_A w^\psi_t \bigoplus w^{\psi'}_t = \begin{cases} T & \text{If } \psi \neq \psi' \\ F & \text{otherwise} \end{cases}
\]

### 3 PROPOSED METHODOLOGY

The proposed methodology is presented in Figure 1. Our methodology, includes three distinct steps. First, the internal communication environment is modelled for the normal, faulty and attack classes. Then, we define a set of execution scenarios that allow us to generate execution datasets that describe the behaviour of the system under normal, faulty, and attack classes. This step is done for each of the three classes. Finally, at step 3 the generated datasets are used for the evaluation of different supervised machine learning algorithms for classification purposes. Our methodology is general and accommodates different classes of faults and attacks. Without loss of generality, in Section 3.1 we discuss the specific classes we consider in this work.

![Proposed Methodology Flowchart](image)

#### 3.1 Execution Classes

Here we define the three execution classes, based on the definitions given in Section 2, and provide the types of faults and
attacks we consider in this work. Faults in an EASH system, often appear in the sensing nodes rather than in the communication channels between the nodes and the coordinator. On the other hand, attacks are often seen to be launched against the communication channels, while also affecting the state variables (e.g., routing tables) of the communicating nodes. Therefore, the fault and attack types we choose: (i) resemble common abnormalities in EASH, and (ii) have a similar effect in order to challenge the differentiation task. Our differentiation mechanism examines the network’s traffic characteristics, in order to separate common sets of faults from attacks. **Normal Class:** In normal behaviour we consider *normal-executions*, where nodes capture measurements and transmit packets to the central node with no attacks or faults. **Faulty Class:** We consider *fault-executions* where a system state \( \sigma_i \) is a faulty state if \( \exists q_j \in V \) with \( \sigma_j^i \) being an abnormal node behaviour state, but not an abnormal node connectivity state (see Definition 2.2). In other words, a node may experience a fault that affects its local state variables (e.g., measurements, routing tables), but its connectivity to the central node is not interrupted. Such failures may also cause routing failures (due to routing table corruption) or packet drops (due to improper generation of the local packets). We use three types of faults in this class (forming set F): (F1) Low Energy Failure, (F2) Routing Failure, and (F3) Packet Dropped Failure. **Attack Class:** In our experiments we assume attacks that affect the communication channels. In particular, we consider *attack-executions* where the system state \( \sigma_i \) is an attack state if \( \exists q_j \in V \) and \( \sigma_j^i \) being both an abnormal node behaviour and abnormal node connectivity state. Man-in-the-middle (MITM) attacks possess such characteristics, as the attacker tries to modify local variables of the communicating parties in order to gain access on the direct connectivity between them. Common attacks performed by MITM are message modification, replay and sink hole. So we use the following types of attacks from this class (forming set A): (A1) Replay Attack, (A2) Sink Hole Attack, (A3) Message Modification Attack.

### 3.2 ML Classification Algorithms

We perform an exploration for different ML algorithms for classification, focusing on the ones considered more appropriate for the underlying problem. The execution classes are used to generate values for descriptive measurements of the network’s behaviour. For the evaluation of the machine learning classification algorithms, the Waikato Environment for Knowledge Analysis (WEKA) tool is used [1]. Algorithm selection for evaluation was based on the following characteristics of our collected dataset: (i) small, (ii) static and (iii) contain nominal attributes. In particular, small datasets may face the problem of over fitting. Furthermore, static datasets cannot be used in online learning algorithms. We use the same algorithms in both test bed and simulation experiments for comparison reasons, despite the fact that the dataset generated in our real-time testbed is larger. Given our dataset characteristics, we evaluate the following four supervised ML algorithms:

- **J48:** A tree classification algorithm that uses a pruned or unpruned C4.5 decision tree for classification.
- **NaiveBayes (NB):** One of the most common classification algorithms that relies on Bayes’ theorem using estimator classes.
- **MultiLayered Perceptron (MLP):** Multilayered perceptron in neural network class.
- **Multinomial Logistic Regression (MLR):** A classifier that builds and classifies instances using ridge estimator.

### 3.3 Evaluation Metrics

The evaluation metrics used for the ML classification algorithms are: the observed accuracy \( (p_o) \), and the kappa statistic value \( (\kappa) \). Observed accuracy was selected as it is the most commonly used metric in such cases of evaluation and it is a percentage (%). The kappa statistic value was selected because it is a metric that takes into consideration unbalancing in the number of instances per class; it is a normalized value between zero and one. In our case, unbalancing is observed since the normal class appears more frequently in a monitoring system, and thus generates more instances in our collected dataset. Hence, the kappa statistic value is computed, based on the observed accuracy \( (p_o) \) and expected accuracy \( (p_e) \), by \( \kappa = \frac{p_o - p_e}{1 - p_e} \).

| Actual Class | Predicted Class |
|--------------|-----------------|
| Yes          | a               |
| No           | c               |

**Table 1: Classification Matrix Example**

To explain how \( p_o \) and \( p_e \) are computed, we use a simple example with two classes, as given in Table 1. Here, values \( a,d \) are the correct classification instances between the actual and predicted classes, and values \( b,c \) are the miss-classified instances for this example. The computation of \( p_o \) and \( p_e \) is given by:

\[
\text{Observed Accuracy} : p_o = \frac{a + d}{a + b + c + d} \\
\text{Expected Accuracy} : p_e = p_{yes} + p_{no} \\
\text{P}_{yes} = \frac{a + b}{a + b + c + d} \cdot \frac{a + c}{a + b + c + d} \\
\text{P}_{no} = \frac{c + d}{a + b + c + d} \cdot \frac{b + d}{a + b + c + d}
\]

### 4 SIMULATION IMPLEMENTATION AND RESULTS

The internal communication environment of EASH was simulated using the OPNET network simulator, considering the
ZigBee communication protocol. Scenarios for normal, faulty, and attack executions, including all failures and attacks presented in Section 3.1, were simulated. In particular, F1 was simulated by lowering a single source energy below transmission threshold; F2 was simulated by changing the destination of each packet to take a random value; F3 was simulated by adding noise to the packets which causes the transmitted data to be invalid, rendering the data load invalid and causing the receiver to drop the packets. Attack classes were simulated by entering a middle node interrupting communication between a single peripheral node and the central coordinator. In A1 the attacker is re-transmitting (without modifying) the packets received from the transmitting node. In A2 we just drop the packets transmitted, and in A3 we increase the payload of the packet before re-transmitting it to the central coordinator. From the simulated scenarios we extract a total of twenty five network characteristics (s.t., throughput, packet delays and data transmitted), to build comma separate instances each associated with the execution class (i.e., normal, fault, or attack) that was used to generate the particular instance. The set of those instances yield our complete dataset. A total of 144 instances were generated after the execution of the simulated scenarios. In our evaluation, a 75% - 25% split was used for training and testing, respectively, among the entire dataset, in both the simulation and real-time experiments. For the simulation experiments we evaluated the following cases:

**Case # 1:** Normal Vs Every Fault (N vs F1 F2 F3)
**Case # 2:** Normal Vs Every Attack (N vs A1 A2 A3)
**Case # 3:** Normal Vs Every Attack Vs Every Fault (N vs F1 F2 F3 A1 A2 A3)

The results are reported in Table 2. For each ML algorithm considered, we list the observed accuracy and kappa statistic value as a pair \((p_o, k)\). As it can be observed, results are promising as in all four algorithms the average accuracy is above 85%. Algorithm J48 slightly outperforms the other algorithms. Additional fine tuning of the algorithm parameters and/or examination of additional features (a total of 26 features were examined in our simulations) could be examined to further improve the observed accuracy.

### Table 2: Evaluation Matrix

| Case / Alg | J48 \((p_o, k)\) | NB \((p_o, k)\) | MLP \((p_o, k)\) | MLG \((p_o, k)\) |
|------------|-----------------|----------------|---------------|---------------|
| Case #1    | (82.76, 0.72)   | (89.66, 0.834) | (80, 0.718)   | (75.86, 0.61) |
| Case #2    | (100, 1)        | (94.4, 0.89)   | (94.4, 0.89)  | (94.4, 0.89)  |
| Case #3    | (86.11, 0.82)   | (80.55, 0.75)  | (91.67, 0.89) | (88.89, 0.86) |
| **Average**| (89.62, 0.85)   | (88.2, 0.82)   | (88.69, 0.83) | (86.38, 0.79) |

### Table 3: Differentiation Results for J48 and Case #1

Based on the Problem definition 2.1, we evaluate the class differentiation for each algorithm. Table 3 presents these results for the J48 algorithm for the particular case #1. A T appears in a cell if the algorithm can differentiate the two classes corresponding on the row and column of the cell, and F if the algorithm failed to differentiate the two classes. Differentiation does not apply for states of the same class (in the diagonal, denoted by -). For J48 and case #1, all instances were classified correctly except for the Low Energy Fault (F1) instances which were misclassified as Packet Dropped Failure (F3) instances, and vice-versa. In a similar manner, we have constructed tables for each of the algorithms and cases considered, which are omitted here due to space limitations.

## 5 TESTBED IMPLEMENTATION AND RESULTS

We run experiments in a real-time testbed consisting of real sensing units. In particular, we used three Raspberry Pi 3 Model B+ connected via Bluetooth to three Sensor Tag CC2650 for sensing humidity, temperature values (peripheral nodes), a Macbook Pro for collecting the transmitted values and running a Wireshark sniffing tool (central node), and an Ubuntu PC to issue the man in the middle attacks (attacker). We formed a star-based topology, as it was introduced in section 2.

In this real-time testbed we implemented the fault classes F1 and F3 and the attack classes A2 and A3. For the normal case, peripheral nodes (formed by RPis) received humidity and temperature and generated TCP packets that were transmitted to the central node.

F1 was emulated by interrupting a single peripheral node communication by cutting off its power. F3 was emulated by setting a threshold on the packet generation procedure indicating the quality of packets being generated. If the value of a generated packet was above the threshold, then the packet was transmitted; otherwise the packet was dropped. To implement the attack classes, we initiated a Man In The Middle Attack (MITM), where the attacker interrupts the communication between a peripheral and the central node by receiving the packets of their communication. This attack was implemented using the Address Resolution Protocol (ARP) spoofing (poisoning) method that allows the attacker to alter the ARP tables on the communicating nodes by sending (spoofed) ARP messages. The aim of this attack is to associate the attacker’s MAC address with the IP address of another host. In our implementation the attacker associate its MAC address with the default gateway of the network. This attack allows the attacker to intercept the data frames on the network, to modify the traffic, or stop all traffic entirely. By intercepting the channel between a peripheral and the central node we can emulate the behaviour of A2, by stopping all the traffic in the network, and of A3, by modifying the
measurements in the packets transmitted. Using Wireshark, we collected a total of twenty four characteristics (e.g. time sent, origin, destination, protocol etc.) for each message sent, forming comma separated instances for our dataset. Each instance was associated with the execution class emulated at the time the packet was captured. The total amount of instances we collected for the real time dataset was 292. Experiments were performed for the following cases: Case #1: Normal Vs Every Fault (N vs F1 F3) Case #2: Normal Vs Every Attack (N vs A2 A3) Case #3: Normal Vs Every Attack Vs Every Fault (N vs F1A2 A3). Evaluation matrix results is shown below, followed by differentiation operator notation results similarly to Table 5. Differentiation operator results are captured using the same classification algorithm and case, namely J48 and Case #1.

Table 4: Evaluation Matrix

| Case / Alg  | J48 (p0, kr) | NB (p0, kr) | MLP (p0, kr) | MLG (p0, kr) |
|-------------|--------------|-------------|--------------|--------------|
| Case #1 SO | (96.36, 0.935) | (92.73, 0.87) | (98.18, 0.97) | (96.37, 0.93) |
| Case #2 SO | (95.83, 0.925) | (97.92, 0.96) | (97.91, 0.96) | (91.66, 0.84) |
| Case #3 SO | (76.71, 0.68) | (76.08, 0.70) | (82.19, 0.75) | (73.97, 0.64) |
| Average SO | (89.64, 0.846) | (89.58, 0.84) | (92.76, 0.89) | (67.33, 0.80) |

Table 5: Differentiation results for J48 and Case #1

Based on Table 4, we can see that results are promising as in all four algorithms the average accuracy is above 85% with algorithm MLP being slightly better. Based on Table 5 we can observe that, Low Energy Fault (F1) instances are miss-classified as Packet Dropped Failures (F3), and vice-versa. As before the rest of the cases are omitted.

An important finding in Case #3 lead us to further examine classes F1 and A2. These classes have a similar effect, as in both, packets stop being delivered from a peripheral node to the coordinator. To this end, we considered case NF1A2, consisted of N, F1 and A2 instances. During this experiment, both TCP and ARP packet traffic was captured. The results using TCP only or TCP & ARP packets are shown below:

Table 6: Evaluation NF1A2 case

As shown on Table 6, the results are further improved with the additional information of the ARP traffic, as the injection of ARP packets during the MITM attack in A2 is captured in the second case, allowing our algorithms to differentiate the two classes.

6 CONCLUSION

This paper presents preliminary results on the differentiation task between a set of common failures and attacks, in an EASH system. Results show that the use of supervised machine learning algorithms is a promising approach as it achieves to differentiate with high accuracy rate between faults and attacks. Some misclassifications for cases with similar impact on the network show the need for further improvements of data generation and evaluation. In future work, we aim to expand the current work by examining correlations between faults and attacks and considering time series datasets.

ACKNOWLEDGMENTS

This work is funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

REFERENCES

[1] Downloading and installing weka, Machine Learning at Waikato University.
[2] Cardenas, A., Amin, S., Sinopoli, B., Giani, A., Persic, A., Sastry, S., et al. Challenges for securing cyber physical systems. In Workshop on future directions in cyber-physical systems security (2009), vol. 5.
[3] Cardoso, G., Rolim, J. G., and Zütn, H. H. Identifying the primary fault section after contingencies in bulk power systems. *IEEE Transactions on Power Delivery* 23, 3 (2008), 1335–1342.
[4] Farwell, J. P., and Rohozinski, R. Stuxnet and the future of cyber war. *Survival* 53, 1 (2011), 23–40.
[5] Gao, Z., Cecati, C., and Ding, S. X. A survey of fault diagnosis and fault-tolerant techniques. *IEEE Transactions on Industrial Electronics* 62, 6 (2015), 3757–3767.
[6] Kim, D., Han, S. C., Lin, Y., Kang, B. H., and Lee, S. Drdr-based knowledge based system to the failure detection in industrial cyber physical systems. *Knowledge-Based Systems* 150 (2018), 1–13.
[7] Mahanty, R., and Gupta, P. D. Application of rbf neural network to fault classification and location in transmission lines. *IEEE Proceedings-Generation, Transmission and Distribution* 151, 2 (2004), 201–212.
[8] Micciché, E. E., Setola, R., Bernieri, G., Panzieri, S., Pascucci, F., and Polycarpou, M. M. Fault diagnosis and network anomaly detection in water infrastructures. *IEEE Design & Test* 34, 4 (2017), 44–51.
[9] Mitchell, R., and Chen, I.-R. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 55.
[10] Pasqualetti, F., Dörfler, F., and Bullo, F. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control* 58, 11 (2013), 2715–2729.
[11] Thukaram, D., Kihmsa, H., and Vijayararasimha, H. Artificial neural network and support vector machine approach for locating faults in radial distribution systems. *IEEE Transactions on Power Delivery* 20, 2 (2005), 710–721.
[12] Uebina, D. I., Giraldo, J., Cardenas, A. A., Valente, J., Faisal, M., Tippennounder, N. O., Ruths, J., Candell, R., and Sandberg, H. Survey and new directions for physics-based attack detection in control systems. US Department of Commerce, NIST, 2016.