

Received May 3, 2021, accepted May 24, 2021, date of publication June 11, 2021, date of current version June 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3088411

# Attack-Detection Architectural Framework Based on Anomalous Patterns of System Performance and Resource Utilization—Part II

ABDULMOHSAN ALOSEEL<sup>1</sup>, SABA AL-RUBAYE<sup>1</sup>, (Senior Member, IEEE), ARGYRIOS ZOLOTAS<sup>1</sup>, (Senior Member, IEEE), AND CARL SHAW<sup>2</sup>

<sup>1</sup>School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, Bedford MK43 0AL, U.K.

<sup>2</sup>Cerberus Security Laboratories Ltd., Bristol BS34 8RB, U.K.

Corresponding author: Abdulmohsan Aloseel (abdulmohsan.aloseel@cranfield.ac.uk)

This work was supported by the Centre of Autonomous and Cyber Physical Systems, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, Bedford, U.K.

**ABSTRACT** This paper presents a unique security approach for detecting cyber-attacks against embedded systems (ESs). The proposed approach has been shaped within an architectural framework called anomalous resource consumption detection (ARCD). The approach's detection mechanism detects cyber-attacks by distinguishing anomalous performance and resource consumption patterns from a pre-determinable reference model. The defense mechanism of this approach acts as an additional layer of protection for ESs. This technique's effectiveness was previously evaluated statistically, and in this paper, we tested this approach's efficiency computationally by using the support-vector machine algorithm. The datasets were generated and collected based on a testbed model, where it was run repeatedly under different operation conditions (normal cases (Rs) versus attacked cases). The executed attack scenarios are 1) denial-of-service (DoS); 2) brute force (BF); and 3) remote code execution (RCE), and man-in-the-middle (MITM). A septenary tuple model, which consists of seven determinants that are analyzed based on seven statistical criteria, is the core of the detection mechanism. The prediction accuracy in terms of classifying anomalous patterns compared to normal patterns based on the confusion matrix revealed promising results, proving this approach's effectiveness, where the final results confirmed very high prediction accuracies in terms of distinguishing anomalous patterns from the typical patterns. Integrating the ARCD concept into an operating system's functionality could help software developers augment the existing security countermeasures of ESs. Adopting the ARCD approach will pave the way for software engineers to build more secure operating systems in line with the embedded system's capabilities, without depleting its resources.

**INDEX TERMS** Anomalous detection, cybersecurity, embedded systems, machine learning, support vector machine algorithm, anomalous resource consumption detection framework.

## I. INTRODUCTION

### A. EMBEDDED SYSTEMS AND CYBERSECURITY CHALLENGES

An embedded system (ES) is a processing unit integrated into a larger cyber-physical system (CPS) to steer its functions. Papp *et al.* [1], [2] described ESs as follows: "An embedded system is a computing system built into a larger system, designed for dedicated functions. It consists of a combination of hardware, software, and optionally mechanical

The associate editor coordinating the review of this manuscript and approving it for publication was Xiali Hei<sup>1</sup>.

parts. Thus, the term refers to any computing systems other than general-purpose [personal computer], (*sic*) or mainframe computers."

ESs typically consist of a central processing unit (CPU), memory, peripheral interfaces, and are constructed optionally with instruction or data caches [3]. In addition, local bus channels are built around them to support communication and data exchange. ESs are connected to a power subsystem, providing power to the components' systems, and most of the ESs are powered by battery.

However, the enormous expansion of the applications of ESs in many different domains highlights the urgent need

to secure these systems against cyber-attacks; the consequences of not doing so might be catastrophic for individuals, companies, organizations, and governments [4]–[6]. The resource limitations of ESs impose tight constraints on computing capacity [7], [8]. The limited processing capability and the limited energy of the ESs block the implementation of high-level security measures, as in conventional computer systems [6], [9]–[13]. Therefore, the capabilities of ESs face the significant challenge of meeting advanced security solution requirements. Marwedel [13] shed light on many tight constraints on ES resources, particularly for SoCs; these limitations are hindering the implementation of advanced security solutions. Therefore, the ESs must be designed with resource-awareness [14]. These constrained capabilities have made the applicability of conventional security mechanisms for ESs impossible [15]. The processing gaps, battery gaps, and costs are at the top of the security challenges list [13]. The high barrier of constraints on computational capability, storage, power, and other aspects of ESs characteristics are complicating the implementation of advanced security solutions, making it difficult to maintain data confidentiality through encryption and key management, as mentioned by Liu *et al.* [16]–[18].

Neglecting security countermeasures at the early stages of ES design increases the number of unprotected entry points, which can result in unauthorized access to the system. As many ESs are battery-powered, and as long as the enhancement of security solutions might sacrifice part of the power sources, it is crucial to at minimum reduce the amount of power consumption with lightweight and efficient solutions [17]. Running robust encryption on ESs with their constrained resources is a critical challenge [19], [20]. Thus, the inability of ESs to handle intensive cryptographic calculations on SoCs is due to their limited resources, giving rise to the need for suitable solutions [14].

Ali *et al.* [21] stated that “the major point of concern is due to the limited resources”; thus, complicated requirements such as higher computational capability and lower power consumption impose serious challenges for the implementation of effective and efficient security solutions. Traditional advanced security solutions in this field offer impractical solutions, considering the ESs’ abilities. The best security solutions must consider the capabilities of software and hardware in the early stages of design; in addition, adopting the “security by design” concept is recommended to avoid unexpected modification costs. Furthermore, we have previously published a comprehensive analytical study of all aspects related to the cybersecurity of ESs [22].

Therefore, I would like to emphasize this study’s main motivation, which is the lack of compatible security solutions in line with ES capabilities. As long as ESs are exposed to the same security threats as multitasked computer systems but with significant limitations for handling advanced security solutions as effectively as conventional systems, and as long as ESs have the same generic security goals (confidentiality, integrity, and availability) as conventional systems, the

implementation of efficient security solutions in line with ES capabilities remains a critical challenge. Therefore, there is an urgent need for efficient security solutions [17], [22]. Efficient and compatible security solutions for ESs are necessary because of their decisive roles in a diverse array of domains. In this paper, we are presenting a new security architecture framework (ARCD), bridging the gap by contributing to this aspect of ES cybersecurity. The ARCD approach derives its rationale in detecting cyber-attacks from the ability to define a comparable reference performance pattern for the ES; this is due to its functions’ determinability. After that, the anomalous pattern (if it exists and is observable during the attack presence) will be used as an indicator to detect the attack, activate an autonomous response, and warn the system administrator.

Based on what has been addressed in this introduction, it has become apparent that one of the most distinctive features of the ES is that ESs are dedicated to performing specific functions, which distinguishes them from conventional systems. Accordingly, we assume that these systems have identifiable performance patterns, and in the event of a cyber-attack, these systems may form abnormal patterns in terms of performance and resource consumption. Therefore, we have reached the linkage point between the main research topic “cybersecurity of ESs” and the purpose of adopting machine learning (ML) methods, where we seek to calculate the prediction accuracy for classifying typical patterns against anomalous patterns that were monitored during the system’s exposure to various cyber-attacks. The collected data values were previously evaluated statistically. Therefore, to complete the validation of the approach, we created training models followed by test models for the datasets to find the predictive accuracy ratio based on selected parameters, which represent the features of datasets in the ML context. Also, the relevant studies to the concept of this approach have been addressed in part I of this work.

## B. MACHINE LEARNING FOR ES CYBERSECURITY

Machine learning, as described by Zhang, is “a subset of artificial intelligence, which builds (*sic*) a mathematical model based on sample data, known as ‘training data,’ in order to make predictions or decisions without being explicitly programmed to perform the task” [23]. The exploitation of ML algorithms in the field of cybersecurity is drawing many researchers in this direction, as it is taking security countermeasures to a new level. Concepts such as anomaly-based detection, recognition strategies, and network traffic pattern analysis have been intensively discussed in many studies [24]–[28]. Therefore, adopting artificial intelligence (AI) concepts, and ML algorithms in particular, contribute to the development of innovative and effective security solutions that differ from known traditional solutions such as anti-virus software, firewalls, and encryption. The domains of anomaly detection implementations are endless, as they can be implemented across many applications such as network traffic, resource utilization, behavioral profiling, statistical analysis,

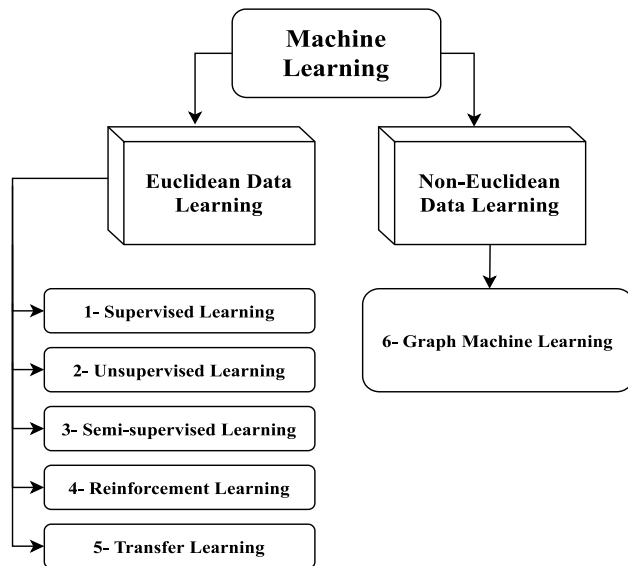


FIGURE 1. Machine learning roots.

and machine learning algorithms. However, ML as a branch of artificial intelligence is a forked domain, and its algorithms and applications are divergent. ML methods can be placed into many categories based on dataset nature or desired purpose (clustering, classifying, regression), or according to the implications of influencing factors, such as environment, object, event, or intended action or decision. According to Zhang, ML can be categorized into two main roots: (1) Euclidean structured data learning and (2) non-Euclidean structured data learning [23]. Based on these two roots, ML methods can be classified into six different types of learning, as illustrated in Figure 1.

Supervised learning, semi-supervised learning, and unsupervised learning are classical learning approaches. The algorithms of the classical learning methods are based on computational statistic concepts. According to the nature of the dataset, the supervised learning methods are suitable when the datasets are labeled, and they can be pre-categorized into groups. The semi-supervised learning method can be divided into three classes: (1) Self-training, by using its own predictions to teach itself; (2) co-training, by using the co-training setting and use their own predictions to teach themselves; (3) active learning, based on an active or participatory role in determining which data points it will be asked to label by an expert or teacher [23]. The semi-supervised method can be adopted when the datasets are partially labeled. Each of these methods has its own algorithms for learning and prediction based on different mathematical equations. The supervised learning methods can be used for classification or regression, while the unsupervised methods can be used for clustering or association, where the features are unknown in early stages.

The fourth and fifth machine learning types based on Euclidean data learning are reinforcement learning and transfer learning. The concept of reinforcement learning is to learn by interacting with an environment, event, or object.

A repeated response to a specific event or object within the surrounded environment will enhance the machine's ability to learn. The reinforcement learning method can be used in a dynamic environment by feeding the learning system with feedback based on the interaction experience. The transfer learning method can be applied to transfer knowledge from the source domain to the target domain [23]. The second root of the ML tree is graph machine learning, which is a sub-category of the non-Euclidean structured data learning root. With this method, the machine will learn based on training samples of the structure of a graph.

Typically, the main purpose of using ML is to establish a regressor or classifier through the training group of a dataset and then to evaluate the performance through a test model with a test dataset. Thus, the two main tasks of ML are classification of discrete data and prediction of continuous data [23]. In the machine learning discussion context, there is an essential term that must be referred to, namely "pattern recognition." A pattern, as described by Watanabe [29], is the "opposite of chaos (*sic*); it is an entity, vaguely defined, that could be given a name." Pattern recognition aims to classify or to cluster a dataset into classes based on their distinctive features, using one of the following four typical approaches: (1) template matching, (2) statistical classification, (3) syntactic or structural matching, or (4) neural networks [23].

In this study, we are applying machine learning method to design a classifier, or predictor model to separate the anomalous patterns from the typical patterns of ES performance. Since the ESs are dedicated to performing specific functions, we assume that they consume their resources in a determinable pattern; thus, we can define a standard performance pattern acting as a reference model; in addition, we assume that during the ES attack, an observable anomalous pattern in resource consumption and performance might be monitored. Thus, our suggested security approach is to detect cyber-attacks based on discriminating anomalous patterns in comparison to a predetermined standard pattern, which can be identified in the early design stages extracted from typical performance patterns. Based on the dataset features and purpose of applying machine learning algorithms, the classification algorithms are the most appropriate algorithms, as we are trying to calculate the prediction accuracy percentage in terms of separating anomalous patterns values from the typical pattern values. Therefore, using classification algorithms for machine learning will prove the validity and efficiency of this technique, or the opposite, depending on the prediction accuracy percentage.

The remainder of this paper is organized as follows. Section II addresses the machine learning role in the cyber-security of ESs and presents the approach hypothesis. Section III presents our case study design, with explanations of normal operating conditions and attack scenario cases. Statistical classification results are evaluated and presented in section IV. The support-vector machine (SVM) algorithm implementation and prediction accuracy results are explained and discussed in sections V and VI. Section VII illustrates the

architectural framework for approach adoption. Finally, the conclusion and future works are summarized in section VIII.

## II. APPLYING THE MACHINE LEARNING METHOD FOR ARCD APPROACH VALIDATION

### A. APPROACH HYPOTHESIS

We conceptualized a research hypothesis, as follows: “In ESs, which are designed to perform specific functions, performance and resource consumption are within a determinable pattern. Thus, an anomalous pattern (if it exists and is identifiable) can be exploited in comparison to a pre-identifiable reference pattern as an indicator to detect cyber-attacks.” In the ML context, defining a standard pattern means labeling its parameter values as the standard consumption and categorizing any outlier values from the normal range, or typical patterns as abnormal values that belong to an anomalous pattern. Based on the ARCD approach, we do not focus on the impact of the attack on the system’s functions but instead focus on the changes in system performance and resource consumption, regardless of the type, method, or surface of this attack—in other words, we detect attacks based on anomalous pattern recognition, which is comparable to a pre-definable reference pattern model.

To test the validity of the hypothesis, we designed a realistic experiment as a case study, using the Smart PiCar (manufactured by SunFounder) as a testbed, reflecting a real-life example of an ES application. The Smart PiCar is designed to carry out reconnaissance missions within a specific geographical area. It was equipped with a recording camera and ultrasound avoidance module. The following figure demonstrates the assembled and configured Smart PiCar.

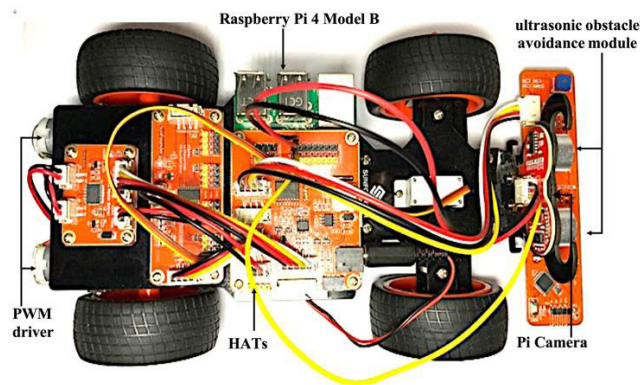


FIGURE 2. Assembled Smart PiCar as a testbed.

### B. MACHINE LEARNING ALGORITHM SELECTION

MATLAB [30], [31], a numeric computing environment platform, was used to train and test the datasets; in particular, we used the classification learner app. The datasets have been examined against 24 different algorithms based on six different training models. These models are (1) decision trees, (2) discrimination analyses, (3) naive Bayes classifiers, (4) support-vector machines, (5) nearest neighbor classifiers,

and (6) ensemble classifiers. The support-vector machine algorithms, along with several other algorithms (such as fine K-nearest neighbor), revealed the highest accuracy percentages in terms of classification, based on the data training sessions. The quadratic support vector machine (QSVM) algorithm was the most accurate algorithm overall, based on the prediction accuracy percentage; accordingly, the test models have been generated.

The SVM algorithm was initially proposed by Vapnik *et al.* [32], [33]; it is based on the Vapnik–Chervonenkis (VC) theory [34], which is related to the statistical learning theory [35], [36]. The SVM algorithm concept involves finding a hyperplane to classify the datasets into classes based on their features. The hyperplane acts as a separator and falls in the middle of a margin, and the margin represents the maximum possible width between two classes supported by support vectors; the support vectors are the closest data points to the hyperplane. The following figure depicts the concept of the SVM algorithm.

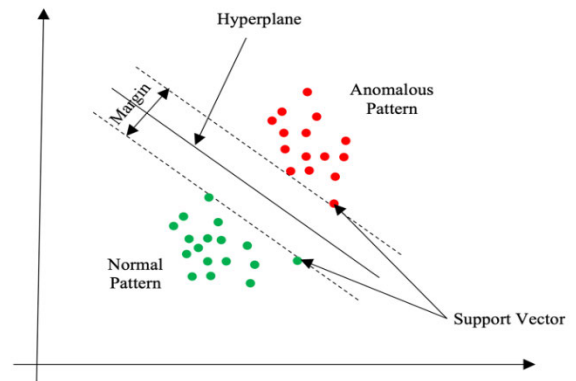


FIGURE 3. The support vector machine algorithm principle.

## III. CASE STUDY DESIGN

This section is divided into two parts. The first part describes the normal operating conditions of the Smart PiCar as an example of CPS applications, with its structure, components, and functions. Based on running the Smart PiCar repeatedly, the parameters’ values have been collected to evaluate the possibility of determining a reference pattern. If the PiCar shows consistency in its performance and resource consumption, we will be able to generate a standard performance pattern act as a reference model; otherwise, defining a reference model will not be possible. The second part explains a set of different attack scenarios that have been executed against the testbed (PiCar) to study ES behavior under these circumstances. If anomalous patterns are observed, and the ability to determine a standard pattern is confirmed, we will develop our approach by merging these two findings and then delivering an architectural framework to facilitate the adoption of this approach.

### A. NORMAL CONDITIONS

The Smart PiCar was assembled and programmed for reconnaissance purposes. It was equipped with a built-in camera

and ultrasonic avoidance module to ensure constant movement and avoid collision with obstacles. In terms of the embedded processing unit, a Raspberry Pi 4 Model B was embedded to take over the computational tasks. During the test period, system performance and resource consumption were monitored, based on seven parameters:

#### I. Resource consumption:

- 1 — CPU utilization.
- 2 — Memory load.
- 5 — CPU temperature.
- 6 — Power consumption in terms of:
  - A — Voltage.
  - B — Current.
  - C — V(D) and V(D-).

#### II. System performance:

- 3 — Alive processes count.
- 4 — Executed threads count.
- 7 — Rate of transmitted and received data.

The terms resource consumption and system performance have been shortened to resource consumption performance (RCP) for the rest of this paper. The proposed approach does not address how the system performs its functions but instead focuses on how it consumes its resources, considering the cyber-attack as an external factor. Regarding the operating condition, the system functions were operated three times repeatedly and separately, and all the data was evaluated statistically. The statistical classification results are discussed in section IV. The three rounds of typical operation were labeled as R1, R2, and R3 and grouped based on modified features, such as dataset-1 and dataset-2, to be tested against different random datasets where their values were collected under attack circumstances.

### B. ATTACKED CASE SCENARIOS

#### 1) DENIAL OF SERVICE ATTACK

The first generated attack scenario is the denial-of-service (DOS) attack. The DoS attack was accomplished by flooding the smart PiCar with directed arbitrary data traffic using the low orbit ion cannon (LOIC) tool. While the Smart PiCar was being attacked, the testbed's parameters were observed, and their values were collected and stored.

#### 2) PASSWORD ATTACK

The second attack scenario is the password attack; some of these techniques include the brute force attack and the dictionary attack. In order to generate the attack, several tools were used. BruteDum was used as a brute-forcing tool, and for port scanning, the Nmap tool was used as the network mapper. The Hydra framework as a parallelized network logon cracker was activated, and it was supported with a random text password list. The password list contained some of the most common and random meaningful words. As in the previous attack scenario, the parameters' patterns were observed, and their values collected and stored.

#### 3) REMOTE CODE EXECUTION AND MALICIOUS SOFTWARE

In the third attack scenario, malicious software was executed remotely as a remote code execution attack. The malicious code is designed to disrupt the Smart PiCar's function by disabling the PiCamera remotely. The expected impact of this attack on the hardware and software performance patterns of the Smart PiCar were observed during the attack period. The testbed's parameter values were collected and stored to be added to the datasets.

#### 4) MAN-IN-THE-MIDDLE ATTACK BY EXECUTING COMMAND REMOTELY

Unlike the previous attack, in this type of attack, the attacker redirects the camcorder recording to their device and gains unauthorized access to a live camera broadcast. To carry out this type of attack, the attacker executes the following command remotely against the victim's device:

```
"raspivid -o - -t 0 -w 960 -h 540 | cvlc -vvv stream:///dev/stdin -sout '#rtp{sdp=rtsp://:8554/}' :demux=h264."
```

The expected impact on system performance patterns was monitored in terms of test parameters. Finally, the parameters' values under these circumstances were collected to be added to the dataset for computational validation.

### IV. STATISTICAL CLASSIFICATION METHOD RESULTS

As we are dealing with real numbers (values of specific parameters), not imaginary values, the principles of real analysis, which is a branch of mathematical analysis and statistics, have been adopted. Based on the research hypothesis (section II.A), we analyzed the collected data according to several statistical criteria (mean, median, mode, standard deviation, range, minimum, and maximum). To evaluate the possibility of defining a standard pattern of system performance and resource consumption under different operational conditions, the above statistical criteria were calculated to define the range and pattern of the RCP, whether in typical or attacked cases.

Based on the statistical criteria results, we determined that the system performs its functions in a similar pattern in terms of performance and resource consumption each time when operated and has an identifiable pattern in a form that can be identified, and a standard performance pattern has been defined accordingly (Part I, Sec V). To test the second part of the approach hypothesis, we generated different attack scenarios to determine if the system was behaving irregularly in terms of its RCP; thus, anomalous patterns might be formed; these anomalous patterns are supposedly distinguishable in comparison to the standard generated pattern. The analysis results proved the validity of what we assumed, as the system showed abnormal performance patterns in all cases while exposed to a cyber-attack. The following table summarizes the statistical classification results.

Table 1 presents a summary of the statistical classification analysis. These results are what encouraged us to measure

TABLE 1. Statistical classification results.

| CPU Utilization |          |      |      |      |      |  | Threads Number |          |      |      |      |      |  | Current    |          |      |      |      |      |  |
|-----------------|----------|------|------|------|------|--|----------------|----------|------|------|------|------|--|------------|----------|------|------|------|------|--|
| Case            | Mean     | Mode | Rang | Min  | Max  |  | Case           | Mean     | Mode | Rang | Min  | Max  |  | Case       | Mean     | Mode | Rang | Min  | Max  |  |
| noml. R1        | 12.3875  | 7.9  | 20.2 | 6.2  | 26.4 |  | noml. R1       | 228.0119 | 232  | 14   | 219  | 233  |  | noml. R1   | 0.6241   | 0.61 | 0.08 | 0.59 | 0.67 |  |
| R2              | 12.6762  | 7.7  | 20.5 | 4.6  | 25.1 |  | R2             | 228.9315 | 232  | 12   | 222  | 234  |  | R2         | 0.6254   | 0.62 | 0.09 | 0.59 | 0.68 |  |
| R3              | 12.7369  | 7.5  | 22.8 | 5.7  | 28.5 |  | R3             | 228.2768 | 232  | 14   | 220  | 234  |  | R3         | 0.6204   | 0.61 | 0.07 | 0.59 | 0.67 |  |
| Attc. DoS       | 31.8988  | 30.9 | 47.3 | 6.8  | 54.1 |  | Attc. DoS      | 241.7589 | 249  | 33   | 226  | 259  |  | Attc. DoS  | 0.6119   | 0.61 | 0.08 | 0.56 | 0.64 |  |
| BF              | 31.1613  | 30.5 | 54.7 | 6.5  | 61.2 |  | BF             | 249.2262 | 240  | 50   | 223  | 273  |  | BF         | 0.6264   | 0.63 | 0.15 | 0.54 | 0.69 |  |
| RCE             | 12.3324  | 6.2  | 32.2 | 4.9  | 37.1 |  | RCE            | 220.5565 | 221  | 11   | 215  | 226  |  | RCE        | 0.557    | 0.55 | 0.14 | 0.51 | 0.65 |  |
| MIMT            | 15.8685  | 7.4  | 33.6 | 5.3  | 38.9 |  | MIMT           | 239.8929 | 241  | 10   | 235  | 245  |  | MIMT       | 0.6174   | 0.61 | 0.17 | 0.53 | 0.69 |  |
| Memory Load     |          |      |      |      |      |  | Temp           |          |      |      |      |      |  | RX packets |          |      |      |      |      |  |
| Case            | Mean     | Mode | Rang | Min  | Max  |  | Case           | Mean     | Mode | Rang | Min  | Max  |  | Case       | Mean     | Mode | Rang | Min  | Max  |  |
| noml. R1        | 9.5238   | 9.5  | 1.4  | 9.3  | 10.7 |  | noml. R1       | 53.4494  | 54   | 5    | 50   | 55   |  | noml. R1   | 4.0357   | 0    | 24   | 0    | 24   |  |
| R2              | 9.508    | 9.5  | 1.5  | 9.3  | 10.8 |  | R2             | 46.5982  | 50   | 14   | 38   | 52   |  | R2         | 4.2887   | 0    | 27   | 0    | 27   |  |
| R3              | 9.5226   | 9.5  | 1.4  | 9.3  | 10.7 |  | R3             | 49.3363  | 51   | 10   | 43   | 53   |  | R3         | 4.997    | 0    | 27   | 0    | 27   |  |
| Attc. DoS       | 10.1381  | 10.2 | 1.8  | 9.7  | 11.5 |  | Attc. DoS      | 61.5714  | 64   | 10   | 55   | 65   |  | Attc. RCE  | 3.9881   | 0    | 27   | 0    | 27   |  |
| BF              | 9.7842   | 9.6  | 1.9  | 9.4  | 11.3 |  | BF             | 58.1071  | 60   | 13   | 50   | 63   |  | MIMT       | 7.7887   | 0    | 55   | 0    | 55   |  |
| RCE             | 9.9      | 9.9  | 0    | 9.9  | 9.9  |  | RCE            | 54.631   | 54   | 8    | 52   | 60   |  |            |          |      |      |      |      |  |
| MIMT            | 10.5003  | 10.5 | 0.1  | 10.5 | 10.6 |  | MIMT           | 56.1488  | 55   | 6    | 54   | 60   |  |            |          |      |      |      |      |  |
| Tasks Number    |          |      |      |      |      |  | Voltage        |          |      |      |      |      |  | TX Packets |          |      |      |      |      |  |
| Case            | Mean     | Mode | Rang | Min  | Max  |  | Case           | Mean     | Mode | Rang | Min  | Max  |  | Case       | Mean     | Mode | Rang | Min  | Max  |  |
| noml. R1        | 167.8958 | 172  | 14   | 159  | 173  |  | noml. R1       | 4.9945   | 4.99 | 0.01 | 4.99 | 5    |  | noml. R1   | 4.0268   | 0    | 24   | 0    | 24   |  |
| R2              | 168.8065 | 172  | 11   | 162  | 173  |  | R2             | 4.996    | 5    | 0.01 | 4.99 | 5    |  | R2         | 4.2917   | 0    | 29   | 0    | 29   |  |
| R3              | 168.1458 | 172  | 13   | 160  | 173  |  | R3             | 4.9952   | 5    | 0.01 | 4.99 | 5    |  | R3         | 5.0536   | 0    | 27   | 0    | 27   |  |
| Attc. DoS       | 181.6369 | 171  | 33   | 166  | 199  |  | Attc. DoS      | 4.9822   | 4.98 | 0.01 | 4.98 | 4.99 |  | Attc. RCE  | 4.0565   | 0    | 26   | 0    | 26   |  |
| BF              | 189.1012 | 180  | 50   | 163  | 213  |  | BF             | 4.99     | 4.99 | 0.01 | 4.99 | 5    |  | MIMT       | 284.2887 | 1    | 645  | 1    | 646  |  |
| RCE             | 167.4524 | 169  | 11   | 162  | 173  |  | RCE            | 5.0001   | 5    | 0.01 | 4.99 | 5.01 |  |            |          |      |      |      |      |  |
| MIMT            | 171.7887 | 173  | 10   | 167  | 177  |  | MIMT           | 4.9948   | 5    | 0.02 | 4.98 | 5    |  |            |          |      |      |      |      |  |

Normal Cases : (noml. R1, R2, and R3), Attacked Cases: (Attc., DoS, BF, RCE, and MIMT)

the prediction accuracy in distinguishing anomalous patterns from the normal typical pattern by using machine learning algorithms. Some statistical parameters, such as the minimum and the maximum, were calculated to determine the range of the RCP. The other criteria, such as the mode and mean, were calculated to find out the pattern of the RCP, where it was observed that the performance and resource consumption rates might be within the limits of the normal range, but at a different pattern. One of the most accurate and reliably applied statistical criteria are the mode values of threads count. By looking at Table 1, the mode value columns show that, in most cases, the significant similarities reached the point of congruence in the typical performance pattern (Rs) cases. We can recognize the anomalous patterns in the system's performance during a cyber-attack by looking to the same criterion. For example, we noticed that the mode values of the threads count for the R1, R2, and R3 cases were identical (232 threads).

In contrast, when the system was subjected to a cyber-attack, the mode values were 249, 240, 221, and 241 threads for each of the DoS, brute force (BF), remote code execution (RCE), and man-in-the-middle (MITM) attacks, respectively. The results were similar for the rest of the statistical results applied within the testbed parameters (see part I for more details). The statistical classification method showed encouraging results, the approach efficiency defining a standard pattern and distinguishing between typical performance patterns and anomalous patterns. The

significant similarity often reached identical limits in terms of the performance and resource consumption rate and under normal operating conditions proved the possibility of determining a standard reference pattern. In contrast, the anomalous patterns of the system while being exposed to different attack scenarios confirmed that it is possible to detect attack existence through the anomalous patterns caused by abnormal levels of performance and resource consumption.

For an overall evaluation at a glance, two graphs (4 and 5) have been generated. Looking at the left side of Graph 5 or 6, we notice the significant similarities in the system's performance patterns in the R1, R2, and R3 cases. On the other hand, by looking at the right side of Graphs 5 and 6, we can see the system's performance pattern anomalies more clearly while the system is under attack. For example, in Graph 5, the system performance pattern was abnormal in the case of a DoS or brute-force attack. Likewise, in Graph 6, anomalous patterns were inconsistent with the conventional patterns that were formed during the RCE and MITM attacks. Therefore, defining the regular performance pattern as a reference model for the system's performance and then monitoring any deviation from it to predict the presence of suspicious activity is an effective mechanism for ESs dedicated to performing specific functions. Based on this compelling mathematical evidence, it is possible to test the validity and feasibility of the proposed security approach computationally through machine learning algorithms.

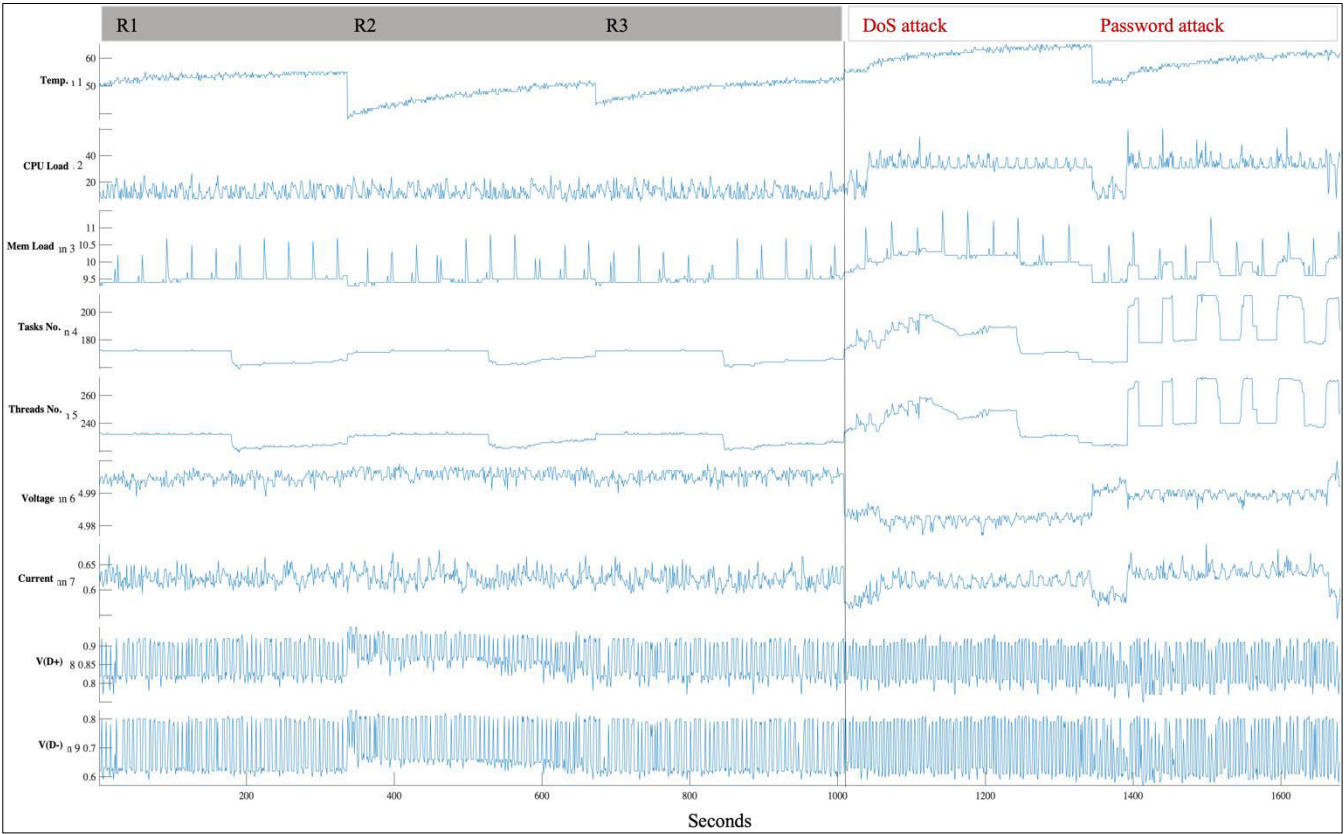


FIGURE 4. Typical patterns (R1, R2, and R3) vs. anomalous patterns under DoS and password attacks.

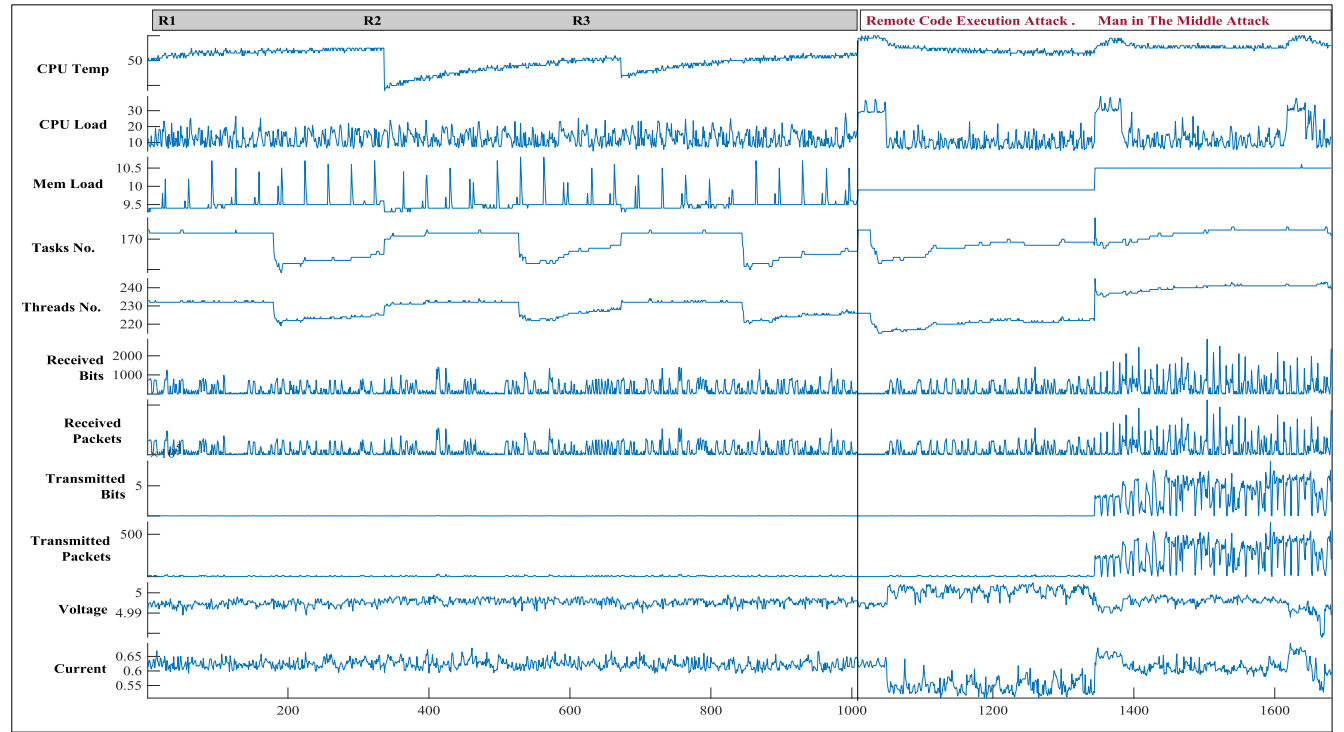


FIGURE 5. Typical patterns (R1, R2, and R3) vs. anomalous patterns under RCE and MITM attacks.

## V. MACHINE LEARNING: DATA NORMALIZATION, MODELING REQUIREMENTS AND ATTACK PREDICTION

Three main components are mandatory for successful machine learning modeling and for accurate computational statistic results. These components are (1) the data, (2) the features, and (3) the algorithm. In this work, we have 24,528 observations that have been generated and collected under different operation conditions. The collected data has been extracted and stored by executing Python code that has been written for this purpose. These datasets represent the performance and resource consumption values of specific parameters and variables that express the features of the dataset. The last component is the algorithm, which is addressed and determined in section II.B.

### A. DATA PREPARATION AND TESTING

For data preparation (DP), the datasets have been grouped or labeled into multi groups. The dataset values of the R1, R2, and R3 cases were grouped into one group and labeled with  $-1$  as the response. Rs cases represent the typical RCP pattern, which was generated by running the Smart PiCar three times under normal conditions. For the attacked cases, the dataset values of the Smart PiCar's performance under DoS attack were labeled with  $1$  as the response. Likewise, the values of the performance and resource consumption of the PiCar under the brute force attack were labeled with  $2$  as the response. Also, the collected data under the RCE and MITM attacks were labeled with  $3$  and  $4$  respectively, which represents the third and fourth attack scenarios. The purpose of labeling the datasets is to determine the response variable, which is mandatory for the creation of trained models based on the supervised learning method. The unbalanced data or misclassification cost factor was taken into account during the data normalization step. The datasets were classified asymmetrically to simulate reality, where the normal pattern is usually the default or the dominant situation (has more observations) and the exception is the anomalous performance pattern. The following table summarizes the classes of the datasets.

**TABLE 2. Case classification and response.**

| Cases                                 |      | Response / Label |
|---------------------------------------|------|------------------|
| Normal Cases<br>(Typical Performance) | R1   | $-1$             |
|                                       | R2   |                  |
|                                       | R3   |                  |
| Attack Cases                          | DoS  | $1$              |
|                                       | BF   | $2$              |
|                                       | RCE  | $3$              |
|                                       | MITM | $4$              |

### B. MACHINE LEARNING MODELING

Determining the data features will contribute to defining its patterns; the data features will act as predictors in the

**TABLE 3. Dataset features.**

| P   | Features / Predictors    |
|-----|--------------------------|
| p1  | CPU Temp.                |
| p2  | CPU Utilization          |
| p3  | Memory Load              |
| p4  | Task Number              |
| p5  | Thread Number            |
| p6  | Received Bits            |
| p7  | Received Packets (RX)    |
| p8  | Transmitted Bits         |
| p9  | Transmitted Packets (TX) |
| p10 | Voltage (V)              |
| p11 | Current (A)              |
| p12 | D+(V)                    |
| p13 | D-(V)                    |

**TABLE 4. Dataset classification.**

| Data group | Cases                | Features                         | Observations |
|------------|----------------------|----------------------------------|--------------|
| Dataset 1  | R1, R2, and R3       | 9 features                       | 9072         |
| Dataset 2  | R1, R2, and R3       | 11 features                      | 11088        |
| Dataset 3  | DoS and BF attacks   | 9 features,<br>p6–p9 excluded    | 6048         |
| Dataset 4  | RCE and MITM attacks | 11 features,<br>p12–p13 excluded | 7392         |
| Dataset 5  | Random selection     | Based on the category            | 6048–18480   |

ML context, and these features or predictors are the parameters of the testbed in the experimentation context. The datasets are classified according to the predictors' value patterns. The features and predictors of the datasets are listed in Table 3.

After defining the features of the data, we can move on to generate trained and test models for pattern classification, and then measure the prediction accuracy. To generate the trained models, we started with trained multiclass models, using the classification learner app in MATLAB, which classifies the datasets into three classes based on their performance and resource consumption patterns (normal patterns versus anomalous patterns). We started with data loading, and then defined its variables (predictors), to be used for classification. In the first training session, to generate the first trained model, we selected datasets 1 and 3, as shown in Table 4. Datasets 1 and 3 consist of 9 predictors and 15,120 observations. Likewise, for the second trained model, datasets 2 and 4 were loaded for a new training session. The datasets of this session consist of 18,480 observations, 11 predictors, and three classes, as shown in Table 4. After completing the training session by testing the datasets against 24 different algorithms based on six different machine learning methods, we selected the QSVM algorithm as the most accurate algorithm for prediction.

Based on the trained models (1 and 2), models 3 and 4 were generated as the test models. The test models 3 and 4 were

**TABLE 5. Trained and test models.**

|            | Model   | Datasets                                           | Features | Classes               | Observations         |
|------------|---------|----------------------------------------------------|----------|-----------------------|----------------------|
| d<br>Model | Model 1 | Datasets 1 and 3                                   | 9        | 3                     | 15120                |
|            | Model 2 | Datasets 2 and 4                                   | 11       | 3                     | 18480                |
| Test Model | Model 3 | Generated without its trained data as a test model | 9        | QSVN Prediction Based | Depends on Dataset 5 |
|            | Model 4 |                                                    | 11       |                       |                      |

generated without their trained data. For the test session, dataset 5 has been created, which is a random selection from other datasets, in addition to new data. Also, in dataset 5, the class labeling (response) was excluded from the datasets, because we are in the testing phase, not the training stage. The following Table (5) summarizes the details of the trained and test models.

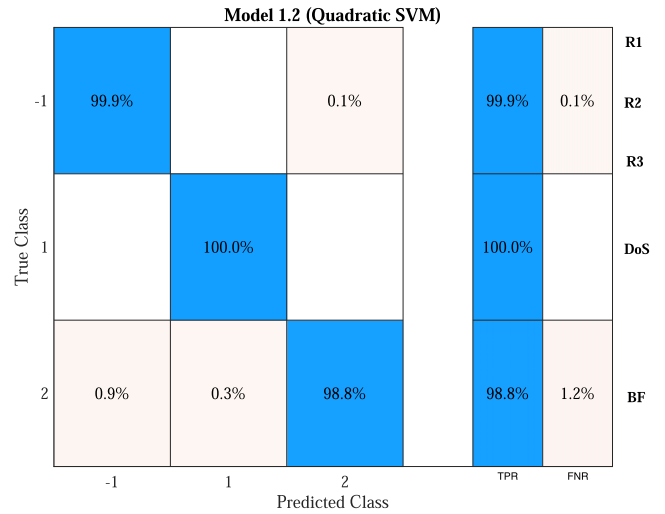
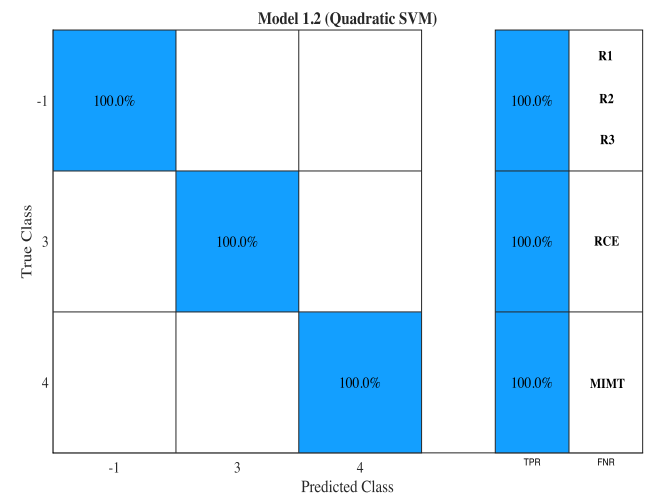
## VI. CLASSIFICATION RESULTS: ANALYSIS AND DISCUSSION

### A. CONFUSION MATRIX RESULTS

Based on the confusion matrix results, the final results showed very high prediction accuracies in terms of distinguishing anomalous RCP patterns from the typical patterns, where the correct prediction has been calculated within the True Positive Ratio (TPR), while the misclassification or (incorrect prediction) has been calculated within the False Negative Ratio (FNR). The accuracy percentage of the prediction is ranged between 100% and 98.8% as the lowest percentage. As shown in Fig. 6, the prediction accuracy in identifying the classic pattern as a standard pattern was 99.9%. The false negative ratio (FNR) percentage was only 0.1%, and this misclassification (0.1%) percentage was classified incorrectly as an anomalous pattern belonging to the BF attack pattern. Regarding the prediction accuracy of the anomalous pattern under the DoS attack, 100% was reached, without any errors. Also, under the BF attack, the true positive ratio (TPR) percentage was 98.8%, and the FNR, which is the misclassification cost, was 1.2%. According to the results, 0.9% of the dataset was incorrectly considered to belong to the typical patterns, and 0.3% of the dataset was considered to belong to the DoS attack pattern.

With regard to the third and fourth attack scenarios, based on the second confusion matrix results in Fig. 7, the prediction accuracy of the typical pattern (Rs cases) reached 100%; in addition, all the anomalous patterns resulting from the attack's existence were identified to be 100% and classified correctly without any errors based on the TPR and FNR percentages. Therefore, increasing the monitored features count helps to make the prediction more precise; the features number of test model 2 was 11 features, compared to 9 features in test model 1.

Overall, detecting a cyber-attack by comparing the in-progressed pattern to a predetermined reference pattern is a very effective method. Also, it is worth noting that the

**FIGURE 6. Confusion matrix 1 (Rs cases vs. attack cases).****FIGURE 7. Confusion matrix 2 (Rs cases vs. attacked cases).**

ability to distinguish anomalous patterns from the standard performance pattern is affected by the number of parameters (predictors), as it is not possible to detect the anomalous patterns accurately based on one predictor. Thus, relay through one parameter or predictor is not a reliable technique: the more parameters, the more accurate the prediction. However, to avoid extensive computing, some sub-parameters that have not shown any changes despite different operating conditions have been excluded in the proposed architectural framework, such as D+(V) and D-(V). In the remaining part of this section, the results for each predictor will be discussed in more detail.

### B. EFFECT OF PREDICTORS

The predictor most impacted by the attack has the most obvious effect in shaping the anomalous pattern. The effectiveness of the predictors varied in terms of the level of their deviation from the typical pattern. Therefore, the combination of these predictors is what contributed to making the anomalous

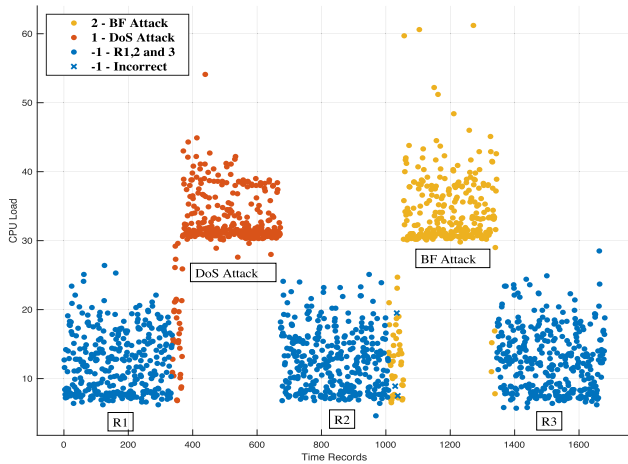


FIGURE 8. CPU pattern classification (Rs cases vs. DoS and BF).

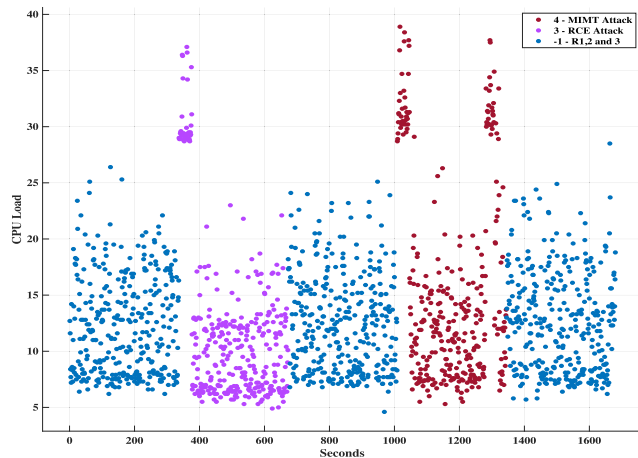


FIGURE 9. CPU pattern classification (Rs cases vs. RCE and MITM).

pattern more apparent. In addition, it was observed that the significant deviation of one predictor compared to another predictor will not only lead to the detection of the anomalous pattern, thus detecting the attack existence, but may also contribute to determining the type of attack. We will address this point in more detail while reviewing the results of the predictors individually.

### 1) CPU UTILIZATION

In terms of the CPU utilization, the ability to discriminate between the typical patterns (reference patterns) of the RCP versus the anomalous RCP is clearly shown in figures 8 and 9. The impacts of the DoS and BF attacks on the PiCar's performance were classified accurately, based on the QSVM algorithm. Under the DoS and BF attack conditions, the differentiation between the standard and anomaly patterns appeared with two different characteristics. First, in terms of the range, which is the deviation from the standard regular consumption range (Rs cases); second, in terms of the pattern. In some cases, the RCPs were within the normal range; however, the RCPs appeared differently in terms of the pattern.

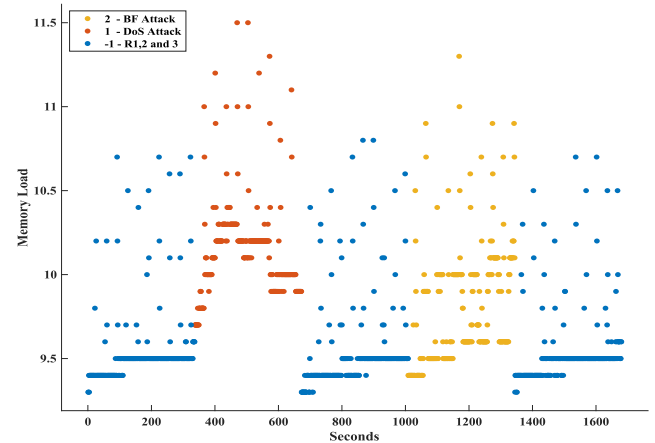


FIGURE 10. Memory pattern classification (Rs cases vs. DoS and BF).

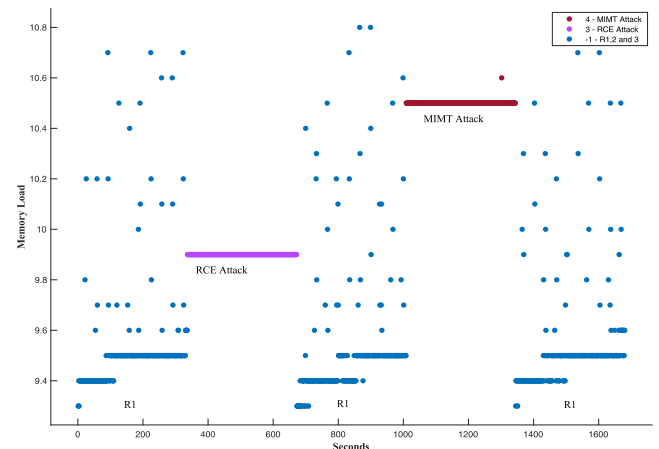


FIGURE 11. Memory pattern classification (Rs cases vs. RCE and MITM).

For example, as shown in Fig. 9, the RCP of the Smart PiCar under the RCE and MITM attacks overlap within the typical range, but due to its anomalous pattern, the attack is still detectable and distinguishable based on the QSVM algorithm.

### 2) MEMORY LOAD

In terms of memory load, as shown in Figures 10 and 11, the memory usage in R1, R2, and R3 were identical in terms of the most frequent value (mode), which was concentrated at 9.50% (Table 1). The general trait of memory usage in Rs cases was consistent and steady, in a manner that can be determined and defined as the reference pattern; whereas, under the cyber-attack circumstances, the memory usage pattern was anomalous in a manner that can be easily detected in comparison with the typical memory pattern. Even though the memory usage rates under the cyber-attacks overlapped within the normal range of memory usage (Figures 10 and 11), the anomalous pattern enabled us to detect the attacks due to the distinguishability of these patterns.

### 3) TASKS COUNT

In terms of the executed processes count, the Rs case mode values were concentrated at 172 processes per second, and as

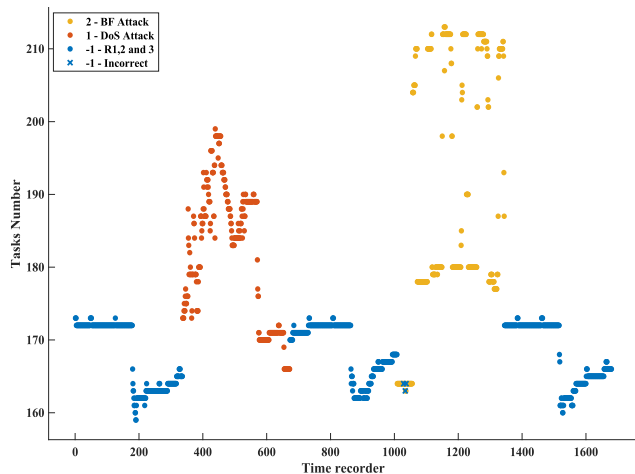


FIGURE 12. Task pattern classification (Rs cases vs. DoS and BF).

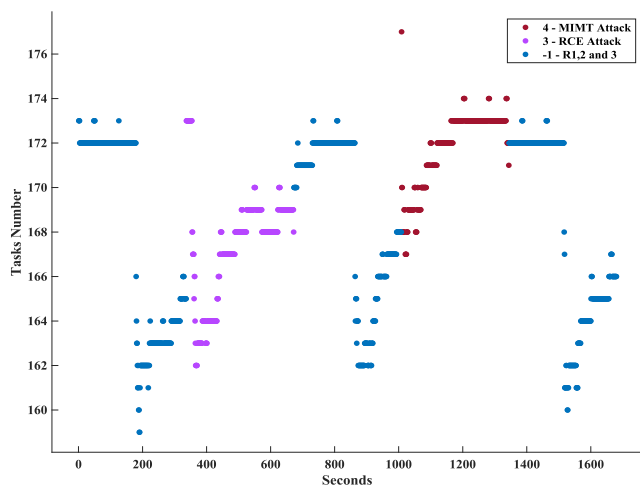


FIGURE 13. Task pattern classification (Rs cases vs. RCE and MITM).

the mode value is identical in R1, R2, and R3 (Table 1), this proves the ability to define a standard range and pattern for the RCP for the Smart PiCar. In contrast, the alive process patterns under all attack cases have been classified accurately as anomalous patterns, as shown in Figures 12 and 13. In Fig. 12, the range and pattern of the executed processes under the BF attack case shaped an abnormal pattern that departed from the normal range and pattern. Likewise, under the DoS attack, the executed process pattern was anomalous, but slightly overlapped with the typical pattern.

In the cases of RCE and MITM attacks (Fig. 13), there was more interference, but due to their anomalous patterns in terms of alive task count (Table 1), it was still possible to separate them using the QSVM algorithm, especially after taking into account the role of other predictors and features with regard to highlighting the anomaly. Based on that, it becomes certain that the ability to detect the existence of cyber-attacks is a consequence of the executed tasks' irregular behavior. This is done by defining a reference pattern of the performed tasks under normal operating conditions and

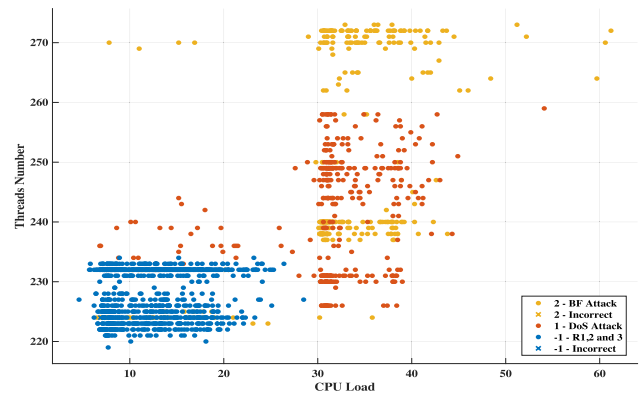


FIGURE 14. Thread pattern classification (Rs cases vs. DoS and BF).

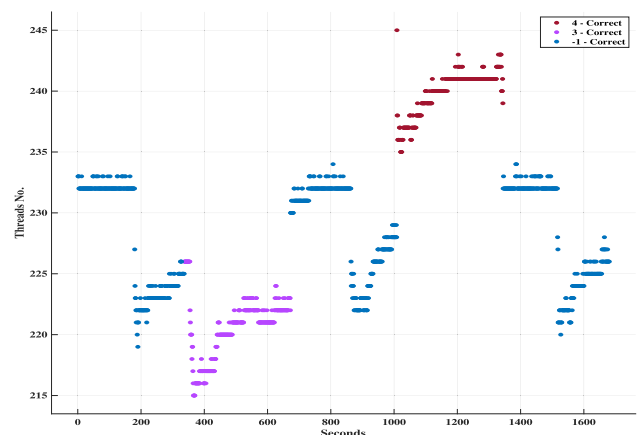


FIGURE 15. Thread pattern classification (Rs cases vs. RCE and MITM).

then using anomalous patterns of executed processes as an indicator referring to the existence of the attack.

#### 4) THREAD COUNT

Regarding the executed threads, in the R1, R2, and R3 cases, the thread count was within a specified area (Fig. 14) and consistent pattern (Fig. 15). The number of executed threads under normal conditions was identifiable (232 threads, Table 1). Thus, it can be defined as a standard performance pattern. On the other hand, the anomalous thread pattern under attack scenarios appeared on the RCP of the Smart PiCar. As shown in Figure 14, the pattern of executed threads under the DoS and BF attacks seemed to be anomalous. It spread more inconsistently across the Fig. 14 axis compared to the usual performance pattern, which concentrated in a specific area.

Also, Fig. 15 shows the pattern of thread count under normal operating conditions and its consistency compared to the Smart PiCar's behavior under attack cases (RCE: purple dots, and MITM: burgundy dots). Therefore, the QSVM algorithm classifies the attack cases from non-attack cases efficiently. It was also noted that the QSVM algorithm classifies the irregular behavior of the Smart PiCar under the MITM attack due to its complete deviation from the normal range. Whereas

the MITM attack was detected because it deviated from the normal range, the other attack (RCE) was marked by its anomalous patterns despite being partially within the normal range. Moreover, the executed thread count in DoS, BF, and MITM cases reached unprecedented numbers, as shown in Table 1; also in the RCE attack case, the thread count significantly decreased in terms of the mode value compared with R1, R2, and R3 (see Table 1).

### 5) CPU TEMPERATURE

The role played by the temperature of the CPU (ARMv8), as one of the data features in terms of distinguishing attack case patterns compared to the typical patterns, was valuable and meaningful, because it helped with the recognition of abnormal patterns resulting from the cyber-attacks. As shown in Figures 16 and 17 and Table 6, under normal operating conditions, the CPU temperature ranged from 38°C to 55°C. In contrast, the maximum CPU temperature in the presence of a cyber-attack reached values that were not previously recorded under normal conditions, such as 65°C, 63°C, and 60°C. This was after taking into consideration the ambient temperature and the initial temperature (Table 6).

**TABLE 6.** CPU temperature (°C).

| Case        | Initial | Maximum | Ambient | Diff. | Mode |
|-------------|---------|---------|---------|-------|------|
| R1          | 50      | 55      | 23      | 32    | 54   |
| R2          | 38      | 52      | 21      | 31    | 50   |
| R3          | 43      | 53      | 23      | 30    | 51   |
| DoS attack  | 55      | 65.00   | 24      | 41    | 64   |
| Brute-Force | 51      | 63.00   | 25      | 38    | 60   |
| RCE attack  | 59      | 60.00   | 25      | 35    | 54   |
| MITM        | 55      | 60.00   | 25      | 35    | 55   |

The elevations in the CPU temperature during the attack periods indicate unconventional levels of CPU temperature. Based on Table 6's results, it was noted that the differences between the maximum temperatures and the ambient temperatures in R1, R2, and R3 were 32°C, 31°C, and 30°C respectively. In comparison, this difference turned to 41°C,

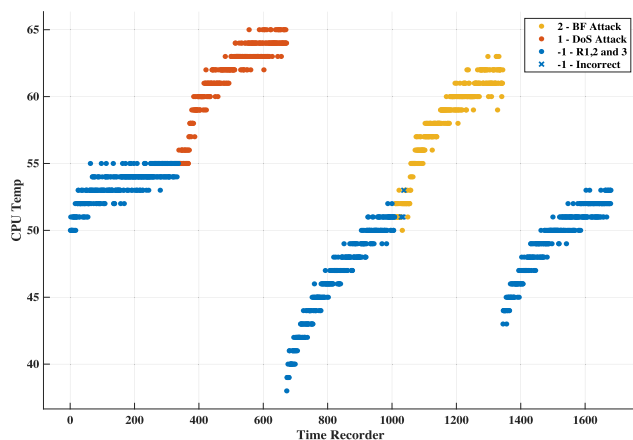


**FIGURE 16.** CPU temp. pattern classification (Rs cases vs. DoS and BF).

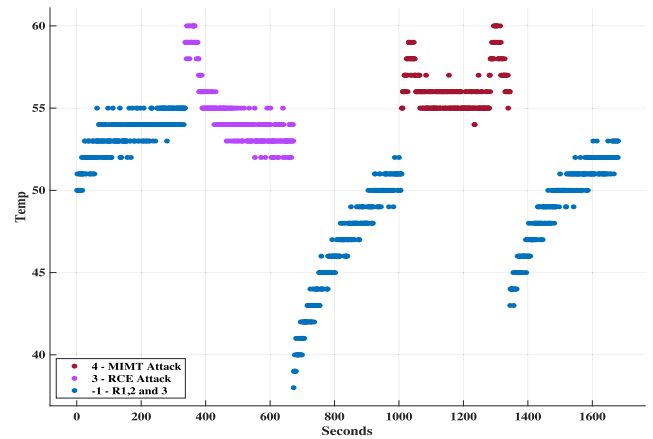


**FIGURE 17.** CPU temp. pattern classification (Rs cases vs. RCE and MITM).

38°C, 35°C, and 35°C under attack conditions. The unusual maximum CPU temperatures support the anomaly-detection technique more closely.

As explained previously, ESs are designed to perform specific functions. Thus, it is to be expected that the ESs consume their resources and perform their functions in an identifiable pattern and in a manner that will be steadily reflected by the CPU temp. Thereby, any anomaly in terms of pattern or any divergence in terms of range compared to the standard pattern performance will indicate the presence of illegal activity, which requires an immediate response.

### 6) POWER CONSUMPTION

Turning to the power consumption rate as one of the predictors added to the training model, we note from Figures 18 and 19 that the voltage patterns of the smart PiCar under normal conditions (Rs cases) reflected a consistent power consumption pattern. Furthermore, this pattern can be identified and defined as a standard pattern, due to its significant similarities in R1, R2, and R3. On the other hand, the RCP of the Smart PiCar under the DoS attack, BF attack, and RCE attack caused abnormal patterns that have given the QSVM algorithm the ability to predict anomalous patterns

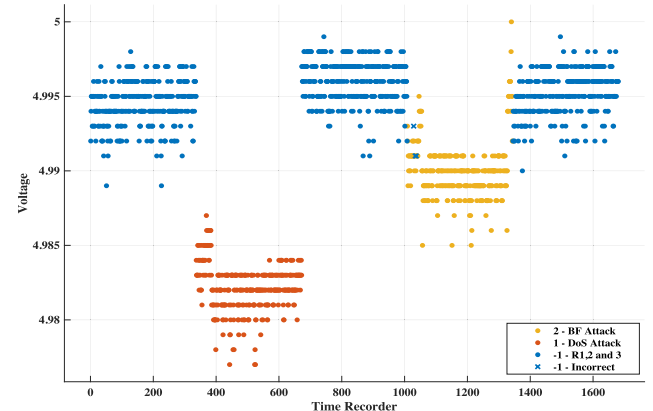


**FIGURE 18.** Voltage pattern classification (Rs cases vs. DoS and BF).

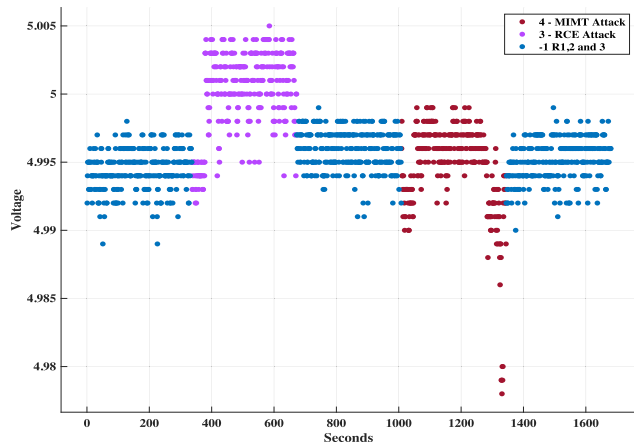


FIGURE 19. Voltage pattern classification (Rs cases vs. RCE and MITM).

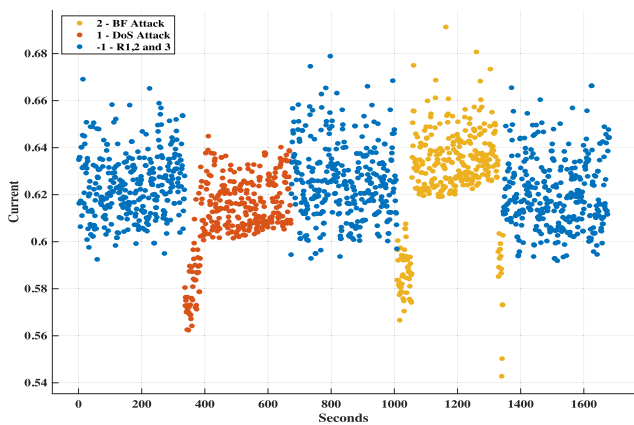


FIGURE 20. Current pattern classification (Rs cases vs. DoS and BF).

more accurately and distinguish them from the patterns of the typical operations. It was also noticed through Fig. 19 that the power consumption under the MITM attack scenario was within the normal range, but at a different rate and an irregular pattern, and here the role of other features supports the ability to recognize the abnormality of the voltage pattern under attack circumstances.

In terms of the current pattern, as shown in Figures 20 and 21, the typical operating modes formed a consistent pattern. While under the attack cases, the current-voltage consumption rate patterns varied in each cyber-attack type. Therefore, this feature enhanced the QSVM algorithm's ability to recognize the cyber-attacks due to their anomalous effects on power consumption, whether voltage or current. In turn, this enhances the effectiveness of the proposed approach for detecting and responding to attacks.

#### 7) RECEIVED & TRANSMITTED PACKET RATE

The rates of received and transmitted packets have been added to the second trained model (as data features) and to the second test model (as predictors). By looking at Figures 22 and 23, it was noticed that the QSVM algorithm greatly benefited from these predictors in detecting the

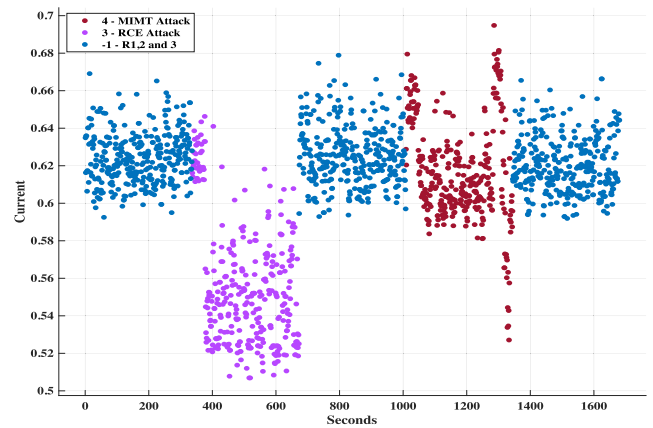


FIGURE 21. Current pattern classification (Rs cases vs. RCE and MITM).

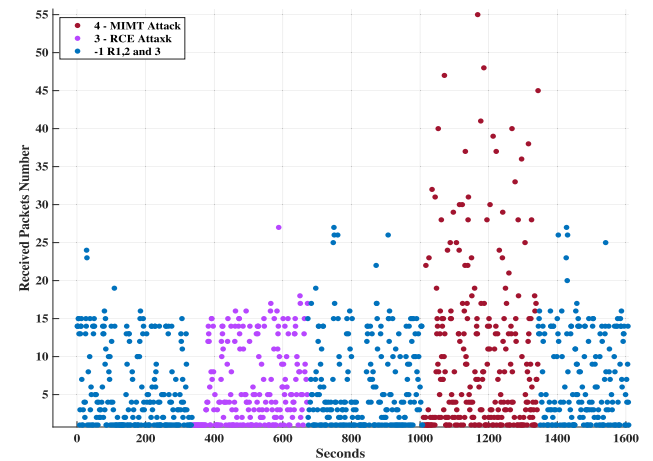


FIGURE 22. RX-rate pattern classification (Rs cases vs. RCE and MITM).

anomalous pattern of the RCP, under the MITM attack in particular.

The RX and TX rates under the MITM attack circumstances reached unprecedented values that enabled the QSVM algorithm to detect the anomaly. Furthermore, the abnormal change in this particular feature, or predictor, will help to detect irregular patterns, detect attack presence, and determine the type of attack as a networked attack. In other words, the unjustified rate of received and transmitted packets indicates a networked attack incident, and this is what has been proven based on these results, both statistically and computationally.

In addition, the average of received and transmitted packets while the system was running was consistent in a way that can be defined as a standard pattern; this will enable us to identify the presence of a networked attack, such as the blackhole attack. In the blackhole attack, the adversary will intentionally breach the data integrity by dropping the received or transmitted packets. Therefore, defining the supposed rate of received and transmitted packets based on specifiable system functions will help to detect any anomalies from the pre-defined standard pattern or any deviation from the typical

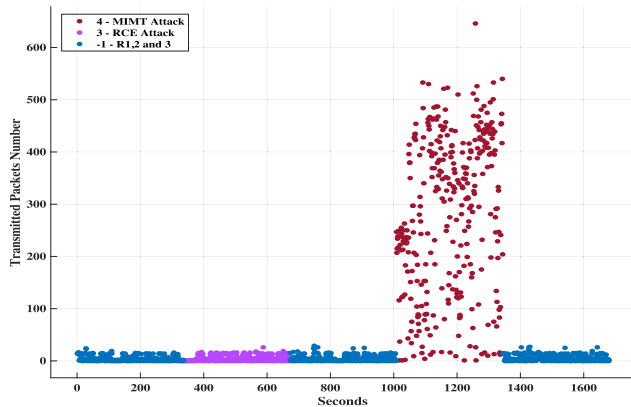


FIGURE 23. TX-rate pattern classification (Rs cases vs. RCE and MITM).

range. As a side note, the received and transmitted packets were added after excluding the  $D+/D-(V)$  features and predictors, which were observed to be unchanged and had no contribution to the identification of abnormal patterns, as was the case with the other parameters and attributes. Throughout different types of attacks, these results compellingly proved the effectiveness of this technique for detecting cyber-attacks due to anomalies in RCP patterns, taking into consideration the ability to determine a standard pattern from the TX and RX rates.

## VII. ARCHITECTURAL FRAMEWORK OF THE ANOMALOUS RESOURCE CONSUMPTION DETECTION (ARCD) APPROACH

After proving the validity of the proposed approach in detecting cyber-attacks, in this section, we address the steps required to facilitate the adoption of this approach for software engineers and operating system designers. The architectural framework (Fig. 24) of the ARCD approach has been built in line with the multiple layers feedback framework of ES cybersecurity (MuLFESC) [22], where steps 1 to 4 take place in parallel with layers 1 to 3, and step 5, which is the detection mechanism, will be implemented in line with layers 3 to 5. Finally, steps 6 and 7 need to be considered in line with layers 7 to 9.

### A. DETERMINING THE COMPONENTS

Defining the system's components is an essential step toward defining the ideal performance pattern and defining the functions that the system will perform. Also, determining the system's application domain will be positively reflected in the prediction of the nature of potential security risks. In our case study, the components of the CPS consist of the physical components that represent the physical parts of a smart car, in addition to the computational part. The physical components consist of:

- Hardware attached on top (Robot HAT),
- Pulse width modulation (PCA9685 PWM Driver  $\times 1$ ),
- Motor driver (TRA9118A  $\times 1$ ),
- Ultrasonic obstacle avoidance module  $\times 1$ ,

- DC gear motor  $\times 2$ ,
- Servo motor  $\times 1$ ,
- Power supply, in addition to the body components of the physical structure.

For the computational part, the Raspberry Pi 4 Model B+ was attached to the Smart PiCar to steer its functions, as the embedded system.

### B. DETERMINING THE FUNCTIONS

Defining the "function" or "functions" for which the system is designed is a fundamental and major step. The determination of system functions is closely related to the recognition of the supposed typical performance. As mentioned earlier, the ESs are dedicated to performing specific functions (sensing, computing, controlling, opening or closing valves, switching, sending and receiving signals, and influencing outcomes in the real world through physical processes), which distinguishes them from conventional multitasked systems. This unique feature makes ESs stand out from traditional computers and enabled us to develop this security approach to protect them. ESs are limited in capabilities due to their nature, which requires them to be small, easy to integrate, inexpensive, portable, and installed in remote sites in harsh conditions, where their energy source is often limited. All these limitations make the application of advanced security solutions effectively impossible. On the other hand, the fact that this type of system is by nature designed to perform specific functions enabled us to use this characteristic to develop this security approach for detecting cyber-attacks.

In our case study, the system functions can be determined as follows: (1) performing reconnaissance tasks in a specific area and making the Smart PiCar move autonomously while maintaining its continuous movement by (2) attaching and enabling an ultrasound module to avoid obstacles and finally, (3) activating an attached recording camera to capture the surrounding area. These are the functions that the CPS was designed to perform.

### C. IDENTIFYING THE TYPICAL PATTERN

Defining the typical, expected, and optimal performance patterns is a fundamental step, and it can be made based on the repeated operation of the CP's functions and by then collecting specific parameter values, which represent the system's performance and resource consumption. Then, these values are analyzed statistically by applying the septenary tuple model the, which consists of seven statistical criteria: minimal, maximal, range, median, mean, mode, and standard deviation on the collected parameter values, and these parameters are: (X): CPU utilization ( $U$ ), memory load ( $M$ ), CPU temp. ( $T$ ), power consumption ( $P$ ), task count ( $N$ ), thread count ( $\vartheta$ ), and rates of received and transmitted packets ( $R$ ), where X refers to a parameter. The CPS must be operated repeatedly under ideal operating conditions (no malfunctions, or presence of cyber-attacks). The collected values must be analyzed statistically to determine the pattern and the range

of each round's performance. To determine the typical ranges ( $Std_r$ ) of each parameter, the minimal ( $X_{minV}$ ), maximal ( $X_{maxV}$ ), and range ( $X_{gV}$ ) values need to be calculated, and to determine the standard pattern ( $Std_p$ ), the following values need to be calculated: the median ( $X_{midV}$ ), mean ( $X_{avgV}$ ), mode ( $X_{modV}$ ), and standard deviation ( $X_{stdV}$ ). Based on (1) the arithmetic mean, we can determine the average performance and resource consumption for each testbed parameter. Based on the (2) mode value, we can determine the most frequent value recorded while the system performs its functions. Thus, we can know the central tendency or density point value and determine the performance and resource consumption patterns. Based on the (3) median value, we can find the middle number in a sorted list of values. Also, the (4) standard deviation criterion will be applied to measure how spread-out the numbers are.

To define the typical range of the RCP, we need to calculate three mathematical criteria—these criteria as follows: (5) maximum value, which is the highest value among the dataset collected during a completed operation cycle in which the system performed all its functions. (6) The minimum value, which is the lowest value recorded while the system was performing its functions. Finally, the distribution range of the values is determined by subtracting the maximum value and the minimum value.

In our case, the Smart PiCar was turned on three times in a row (R1, R2, R3), and the values of each parameter ( $X$ ) = { $U, M, N, J, T, P$ } were collected for each run, in which all system functions were performed completely. After analyzing these parameters statistically according to the septenary tuple model, the results revealed a significant similarity that often reached the point of congruence, thus proving the possibility of defining a standard performance pattern.

In the machine learning context, whenever the patterns of operation in normal conditions show great congruence or convergence in terms of performance and resource consumption, there is less need for data training based on the supervised learning method. Conversely, whenever the patterns of operation under normal operating conditions have diverged, the greater the need for more data training sessions to reach the optimal standard pattern. Therefore, the required training overhead is mainly dependent on the level of convergence or variance in operating patterns under normal conditions.

In contrast, the required training overhead to distinguish the generated standard pattern derived from typical operations versus anomalous patterns depends on the level of anomaly. The greater the anomaly from the standard pattern, the easier it is to detect the attack, and therefore the need to train the data based on the supervised learning method decreases. The lower the level of anomaly, the greater the need for more data training sessions to accurately define the detection threshold without confusion and to prevent a false positive case.

#### D. GENERATING A STANDARD MODEL AS A REFERENCE PATTERN

The main purpose of recognizing typical performance patterns based on the parameter values that have been analyzed statistically is to generate a standardized reference model for a particular implementation of a CPS. The generated model will act as a reference and control model of the optimal performance to distinguish anomalous patterns that deviate from it; this means we are exploiting the anomaly in performance and resource consumption as an indicator that reveals the existence of a cyber-attack, which requires an immediate response. Defining a standard pattern that serves as a reference model will contribute to developing the detection mechanism and improve the CPS performance by imposing the necessary restrictions on resource consumption to prevent the depletion of system resources resulting from unintended software errors or misuse. In our case study, the reference model (Part I, Table 12) was defined based on the Rs cases' statistical analysis results. To create a flexible reference model, we determined the reference models' values as a set of values that fall between the highest and lowest values among Rs cases for each statistical criterion. In other words, the maximal values of the reference model for each statistical criterion are the highest values among the maximum values of Rs cases, and the minimum value is the lowest value among the minimum values of Rs cases. Additionally, consideration was given to the probability of having values very close to those of the reference model by calculating and adding the most significant differences between Rs cases (Table 1) to the reference model values.

#### E. DESIGNING AND APPLYING THE DETECTION MECHANISM

The detection mechanism is the core of the architectural framework. The detection technique is based on recognizing anomalous patterns of the system's performance by observing specific values of selected statistical criteria for designated parameters that are compared to the generated reference model, as illustrated in Fig. 24. Thus, the outlier values are categorized as abnormal values, indicating an anomalous pattern that requires immediate action. The RCP pattern deviation from the standard predefined performance level enables the OS layer to predict a cyber-attack's existence. This mechanism was developed after proving that the ESs have an identifiable pattern under normal operating conditions. Also, it was proven that ESs, during exposure to various cyber-attacks, change their performance patterns and their resource consumption in a way that can be distinguished compared to their typical performance. By adopting this mechanism, it becomes possible to detect cyber-attacks based on deviation from the normal range and based on the abnormality of the pattern within the normal range.

#### F. MONITORING THE SYSTEM PERFORMANCE PATTERN

After the detection mechanism is designed, it must be activated as an integral part of the operating system layer.

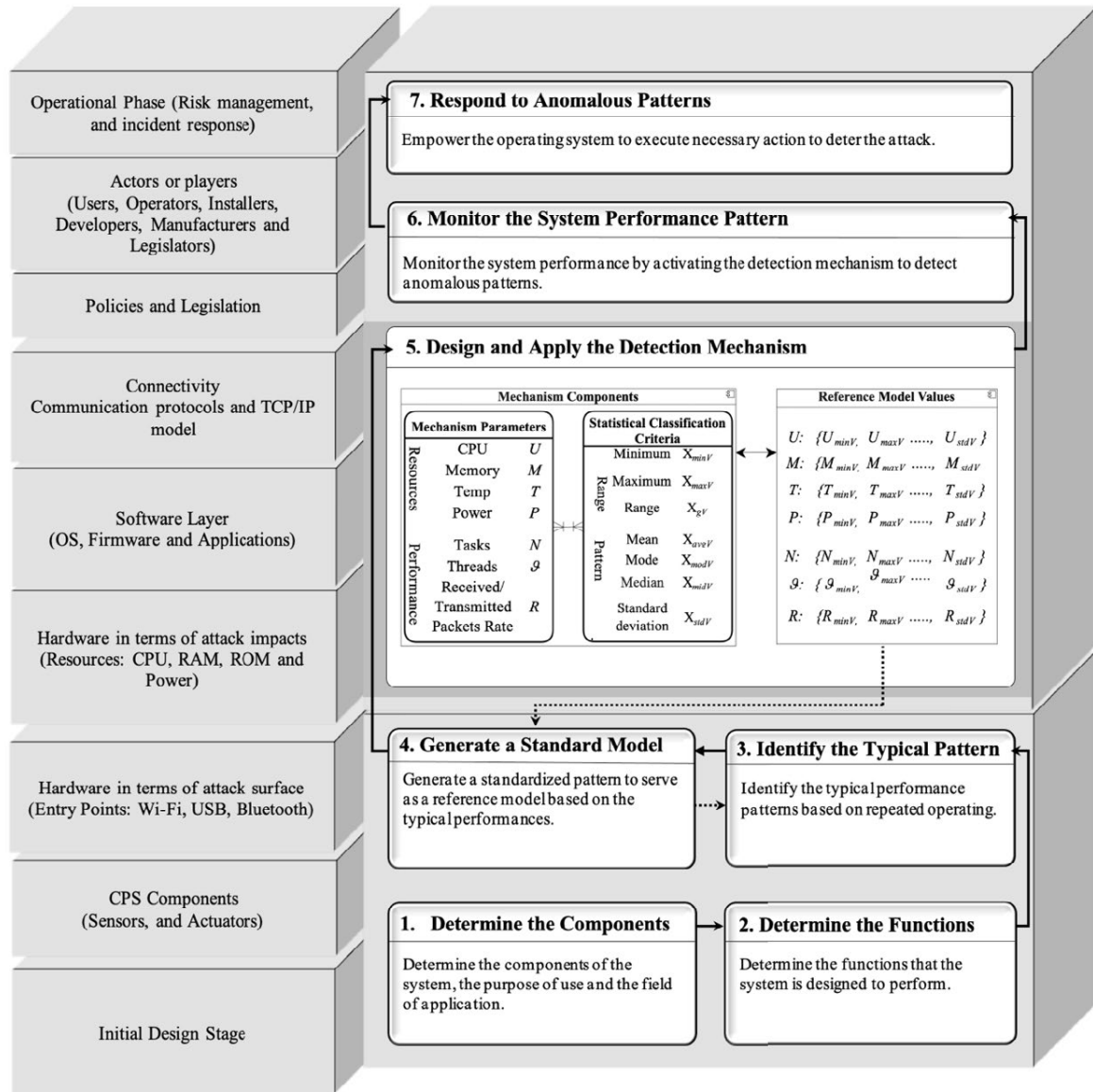


**FIGURE 24.** Architectural framework of the anomalous resource consumption detection (ARCD) approach.

Activating the mechanism involves permanent monitoring of the statistical classification criteria of the performance parameters by observing any deviation from the reference performance model. Embedding the detection mechanism instructions within the OS architecture will create an additional layer of protection. As long as there are no anomalies in the performance or resource consumption patterns, there will be no need to take any action; however, the moment the performance or resource consumption patterns deviate from the defined reference patterns, or its values diverge out of the normal range, then the system must immediately alert the user about the anomaly and attract attention by generating a sound alert or by displaying a warning message. The next step is then to automate the operating system layer by executing an appropriate procedure to prevent the attack.

However, if the functions that the ES was designed to perform have been modified, it is possible that the detected resource utilization pattern might be categorized as an anomalous pattern, even if it is due to legitimate processes. Therefore, it is mandatory to regenerate a new reference pattern according to the modified functions by following steps three and four. Also, as relying on one parameter might lead to false-positive detection, combining a set of parameters will increase the accuracy of prediction, thus preventing such case.

In terms of the detection latency, which is the time between the initiation of the attack and it being detected, based on the proposed approach, it will be the time required to statistically evaluate the parameters' values as well as the time of anomaly formation in the performance pattern. The results of the DoS

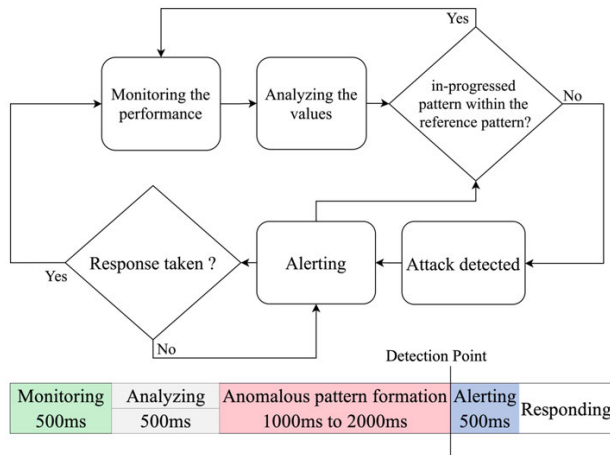


**FIGURE 25.** Chronology of the attack detection and events sequence.

and password attack scenarios (Part I, Sec VII) show that the DoS attack was executed at 00:26:25s with the abnormal pattern appearing at 00:26:26s, and the password attack was executed at 00:16:16s with the anomaly pattern starting to form at 00:16:18s (a 1s delay). Therefore, the time between attack execution and detection, considering the time taken to monitor the stat values in real-time, will be between 2000ms to 3000ms (Part I, Appendix I: snapshots from Real-time monitoring). Moreover, if there is a delay in the formation of the abnormal pattern, it will reflect negatively on the accuracy of the prediction percentage because the dataset collected under attack scenarios would have been classified as being of a normal pattern, thereby reducing the prediction accuracy percentage, and this did not happen in the test or training sessions. However, other factors, i.e., the type of attack, the architecture of the operating system, and the computing capabilities of the ES need to be taken into consideration. Figure 25 gives a visualization of the sequence of events and the timeline for detecting the attack according to the proposed approach.

### G. RESPONDING TO ANOMALOUS PATTERNS

It would be best to build an autonomous response into the OS architecture, in order to benefit from the anomalous pattern auto-detection and provide an optional interaction from the user. The available response options for an operating system mainly depend on the OS layer's architecture and the hardware components used in the system construction. Automating the operating system's response to prevent attacks should not be an obstacle for software developers, as we have demonstrated the ability to detect a cyber-attack based on anomalous pattern recognition. Configuring the OS layer, controlling the hardware, and applying restrictions to resource consumption are all solutions that are under software engineers' control, and they can adapt these components to fit their needs. While a cyber-attack's occurrence is not something we can entirely stop, we can at least deal with it and neutralize its harm by adopting a security approach that

detects such attacks. Since this proposed approach creates an additional integrated layer of protection in line with existing countermeasures, the autonomous response mechanism can be designed to (1) activate the system architecture's built-in defenses, (2) apply restrictions on resource usage, and (3) disable the attacked entry points, or (4) auto-disable the compromised network port and (5) end and re-establish the connection session. Another option is to (6) isolate the hacked system from the network by activating (auto-reboot) to interrupt the attack.

The architectural framework of the Anomalous Resource Consumption Detection (ARCD) is a generalized approach that can be applied to any ES that normally operates within a constrained pattern of behavior. As long as the IoT applications mainly range between monitoring, controlling, sensing, opening or closing, alerting, environment monitoring, safety, process monitoring, logistic management, signals transmitting, computing, and switching, it is possible to define their functions. Thus, the ARCD approach, with its detection mechanism, is compatible with any processing unit embedded into a larger CPS to steer its specified functions. All that is needed is to determine the functions precisely and find the default normal performance pattern. This is defined as a reference pattern by following the proposed steps within the presented architectural framework. Any anomalous pattern will then be recognized as an indication of a cyber-attack or a system fault.

### VIII. CONCLUSION

The proposed approach derived its rationale from three arguments. First, unlike conventional multitasked computer systems, ESs are dedicated to performing specific functions. Thus, (secondly) it was assumed that these systems perform their functions and consume their resources in a consistent pattern, therefore, extracting or generating a standard pattern based upon the typical patterns is possible. Third, based on the second argument, it was assumed that these systems might form observable anomalies in their performance and resource consumption patterns while being subjected to cyber-attacks. Thus, these anomalous patterns would deviate from the pre-defined standard pattern, and we have proven that it is possible to define this pattern in advance. All these assumptions were proven to be accurate.

By merging these findings, the approach's rationale was presented in a hypothesis, and the QSVM algorithm was used to test its validity. Thus, the approach's effectiveness has been tested and statistically and computationally validated by means of a testbed experiment. The high prediction accuracy percentages presented in the confusion matrices revealed promising results, proving this approach's effectiveness. Accordingly, the ARCD approach's architectural framework was developed to facilitate this approach's adoption by software engineers. The ARCD approach acts as an additional layer of protection for ESs, and it uses the system performance's irregular behavior to reflect in the hardware and software layers as suggested improvements, without excessive

computational requirements or additional hardware component costs. This is due to the statistical detection mechanism based on anomalous performance pattern recognition. The aim of delivering this approach is to detect cyber-attacks without conflicting with the embedded system's characteristics. Thus, the ARCD approach has paved the way for software engineers to strengthen operating systems and enhance their rigidity and reliability.

Finally, for an overall conclusion of the contributions of this work considering all its parts from the analytical review of cybersecurity for embedded systems, the presented novel approach for detecting cyber-attacks in embedded systems, and the attack-detection architectural framework, the following points have been listed:

- The Multiple Layers Feedback Framework of Embedded System Cybersecurity (MuLFESC) could contribute to implementing comprehensive and effective "Security by Design" solutions by providing feedback to the design stage of cybersecurity for embedded systems (CSES).
- The Anomalous Resource Consumption Detection (ARCD) approach consists of a septenary tuple model as the detection mechanism presenting a security solution that can work along with existing countermeasures to augment the security of the operating system layer.
- The Architectural Framework of the ARCD approach facilitates the adoption of this approach by software engineers.

As future extensions and applications of this work, it would be beneficial to test other cyber-attacks such as Blackhole attack or SQL injection and expand the research to include other testbeds such as temperature monitoring systems or intelligent motion detection systems. Implementing the concepts of this approach within an open-source OS, and generating a practical software framework, would further test its feasibility and applicability as an off-the-shelf solution. Also, applying the concept of this approach to low-end devices such as microcontrollers with hardware machine learning support would be another possible extension.

## REFERENCES

- [1] D. Papp, Z. Ma, and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," in *Proc. 13th Annu. Conf. Privacy, Secur. Trust (PST)*, Jul. 2015, pp. 145–152, doi: [10.1109/PST.2015.7232966](#).
- [2] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*. Hoboken, NJ, USA: Wiley, 2002. [Online]. Available: <https://www.wiley.com/en-gb/Embedded+System+Design%3A+A+Unified+Hardware+Software+Introductionp-9780471386780> and [https://www.mendeley.com/catalogue/2b2b3b86-236b-3dd4-91fc-c8b8053f1a78/?utm\\_source=desktop&utm\\_medium=1.19.8&utm\\_campaign=open\\_catalog&userDocumentId=%7B9dc868ee-9425-4667-8ef8-3e20bc6e4934%7D](https://www.mendeley.com/catalogue/2b2b3b86-236b-3dd4-91fc-c8b8053f1a78/?utm_source=desktop&utm_medium=1.19.8&utm_campaign=open_catalog&userDocumentId=%7B9dc868ee-9425-4667-8ef8-3e20bc6e4934%7D)
- [3] P. V. Pham Van and N. N. Binh, "Embedded system architecture design and optimization at the model level," *Int. J. Comput. Commun. Eng.*, vol. 1, no. 4, pp. 345–349, 2012, doi: [10.7763/ijcce.2012.v1.87](#).
- [4] E. Levy, "Crossover: Online pests plaguing the off line world," *IEEE Secur. Privacy*, vol. 1, no. 6, pp. 71–73, Nov. 2003, doi: [10.1109/MSECP.2003.1253573](#).
- [5] H. He, C. Maple, T. Watson, A. Tiwari, J. Mehnen, Y. Jin, and B. Gabrys, "The security challenges in the IoT enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1015–1021, doi: [10.1109/CEC.2016.7743900](#).
- [6] M. Abomhara and G. M. Køien, "Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks," *J. Cyber Secur. Mobility*, vol. 4, no. 1, pp. 65–88, 2015, doi: [10.13052/jcsm2245-1439.414](#).
- [7] S. Ravi, P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *Proc. 41st Annu. Conf. Design Autom. (DAC)*, 2004, p. 753, doi: [10.1145/996566.996771](#).
- [8] S. Hameed, F. I. Khan, and B. Hameed, "Understanding security requirements and challenges in Internet of Things (IoT): A review," *J. Comput. Netw. Commun.*, vol. 2019, pp. 1–14, Jan. 2019, doi: [10.1155/2019/9629381](#).
- [9] V. H. Meshram and A. B. Sasankar, "Security in embedded systems: Vulnerabilities, pigeonholing of attacks and countermeasures," *IOSR J. Comput. Eng.*, vol. 5, pp. 11–15, 2016. [Online]. Available: [http://www.iosrjournals.org/iosr-jce/pages/Conf.15013\(5\).html](http://www.iosrjournals.org/iosr-jce/pages/Conf.15013(5).html)
- [10] C. Bodei, S. Chessa, and L. Galletta, "Measuring security in IoT communications," *Theor. Comput. Sci.*, vol. 764, pp. 100–124, Apr. 2019, doi: [10.1016/j.tcs.2018.12.002](#).
- [11] S. Parameswaran and T. Wolf, "Embedded systems security—An overview," *Design Autom. Embedded Syst.*, vol. 12, no. 3, pp. 173–183, Sep. 2008, doi: [10.1007/s10617-008-9027-x](#).
- [12] D. D. Hwang, P. Schaumont, K. Tiri, and I. Verbauwhede, "Securing embedded systems," *IEEE Secur. Privacy Mag.*, vol. 4, no. 2, pp. 40–49, Mar. 2006.
- [13] P. Marwedel, *Embedded System Design*. Cham, Switzerland: Springer, 2018.
- [14] W. Wang, X. Zhang, Q. Hao, Z. Zhang, B. Xu, H. Dong, T. Xia, and X. Wang, "Hardware-enhanced protection for the runtime data security in embedded systems," *Electronics*, vol. 8, no. 1, p. 52, Jan. 2019, doi: [10.3390/electronics8010052](#).
- [15] P. I. Radoglou Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, threats and solutions," *Internet Things*, vol. 5, pp. 41–70, Mar. 2019, doi: [10.1016/j.iot.2018.11.003](#).
- [16] X. Liu, M. Zhao, S. Li, F. Zhang, and W. Trappe, "A security framework for the Internet of Things in the future Internet architecture," *Future Internet*, vol. 9, no. 3, p. 27, Jun. 2017, doi: [10.3390/fi9030027](#).
- [17] H. Chai, G. Zhang, J. Zhou, J. Sun, L. Huang, and T. Wang, "A short review of security-aware techniques in real-time embedded systems," *J. Circuits, Syst. Comput.*, vol. 28, no. 2, Feb. 2019, Art. no. 1930002, doi: [10.1142/S0218126619300022](#).
- [18] A. Riahi Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the Internet of Things," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 118–137, Apr. 2018, doi: [10.1016/j.dcan.2017.04.003](#).
- [19] H. Habibzadeh, B. H. Nussbaum, F. Anjomshoa, B. Kantarci, and T. Soyata, "A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities," *Sustain. Cities Soc.*, vol. 50, Oct. 2019, Art. no. 101660, doi: [10.1016/j.scs.2019.101660](#).
- [20] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017, doi: [10.1109/JIOT.2017.2703172](#).
- [21] S. Ali, T. Al Balushi, Z. Nadir, and O. K. Hussain, *Cyber Security for Cyber Physical Systems*, vol. 768. Cham, Switzerland: Springer, 2018, pp. 11–33, doi: [10.1007/978-3-319-75880-0\\_2](#).
- [22] A. Aloose, H. He, C. Shaw, and M. A. Khan, "Analytical review of cybersecurity for embedded systems," *IEEE Access*, vol. 9, pp. 961–982, 2021, doi: [10.1109/ACCESS.2020.3045972](#).
- [23] X.-D. Zhang, "Machine learning," in *A Matrix Algebra Approach to Artificial Intelligence*, vol. 45, no. 13. Singapore: Springer, 2020, pp. 223–440.
- [24] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020, doi: [10.1109/ACCESS.2020.2973023](#).
- [25] T. J. Shimeall and J. M. Spring, "Recognition strategies," in *Introduction to Information Security*. Amsterdam, The Netherlands: Elsevier, 2014, pp. 253–274.
- [26] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos, and G. Marreiros, "Performance evaluation of unsupervised techniques in cyber-attack anomaly detection," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 11, pp. 4477–4489, Nov. 2020, doi: [10.1007/s12652-019-01417-9](#).

- [27] M. Ahmed and A. N. Mahmood, "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection," *Ann. Data Sci.*, vol. 2, no. 1, pp. 111–130, Mar. 2015, doi: [10.1007/s40745-015-0035-y](https://doi.org/10.1007/s40745-015-0035-y).
- [28] R. Fontugne, Y. Himura, and K. Fukuda, "Evaluation of anomaly detection method based on pattern recognition," *IEICE Trans. Commun.*, vol. E93-B, no. 2, pp. 328–335, 2010, doi: [10.1587/transcom.E93.B.328](https://doi.org/10.1587/transcom.E93.B.328).
- [29] S. Watanabe, *Pattern Recognition: Human and Mechanical*. Hoboken, NJ, USA: Wiley, 1985.
- [30] *The MathWorks*. Accessed: Apr. 16, 2020. [Online]. Available: <https://www.mathworks.com/products/matlab.html>
- [31] C. Moler and J. Little, "A history of MATLAB," *Proc. ACM Program. Lang.*, vol. 4, pp. 1–67, Jun. 2020, doi: [10.1145/3386331](https://doi.org/10.1145/3386331).
- [32] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).
- [33] W. S. Noble, "What is a support vector machine?" *Nature Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006, doi: [10.1038/nbt1206-1565](https://doi.org/10.1038/nbt1206-1565).
- [34] W. K. Cheng, B. Dibble, and D. Assanis, "Guest editorial vapnik-chervonenkis (vc) learning theory and its applications," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 985–987, Sep. 1999, doi: [10.1109/TNN.1999.788639](https://doi.org/10.1109/TNN.1999.788639).
- [35] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, 1999, doi: [10.1109/72.788640](https://doi.org/10.1109/72.788640).
- [36] H. Zhen, L. Hong, F. Mujiao, and X. Chunbi, "Application of statistical learning theory to predict corrosion rate of injecting water pipeline," in *Proc. 9th IEEE Int. Conf. Cognit. Informat. (ICCI)*, Jul. 2010, pp. 132–136, doi: [10.1109/COGINF.2010.5599754](https://doi.org/10.1109/COGINF.2010.5599754).



**ABDULMOHSAN ALOOSEEL** received the B.S. degree in management information systems (MIS) from King Faisal University, in 2006, and the M.S. degree in computer and network security from Middlesex University London, in 2015, where he is currently pursuing the Ph.D. degree in cybersecurity of embedded systems with the School of Aerospace, Transport and Manufacturing (SATM), Cranfield University.



**SABA AL-RUBAYE** (Senior Member, IEEE) received the Ph.D. degree in electrical and electronic engineering from Brunel University London, U.K. She is currently a Senior Lecturer and a Fellow of DRATeC, she is leading connected system research group in the School of Aerospace, Transport and Manufacturing, Cranfield University, U.K. She is participating in developing industry standards by being an Active Voting Member of IEEE P1920.2, Standard for Vehicle-to-Vehicle Communications for Unmanned Aircraft Systems, and of IEEE P1932.1, Standard of License/Unclassified Interoperability. She has published many articles in IEEE journals and conferences and was a recipient of the best technical paper award (twice) published in IEEE Vehicular Technology in 2011 and 2015, respectively. Her main research interests include, but not limited to, UAV connectivity, communication networks, artificial intelligent, and safety and security of autonomous vehicle. She has been the general co-chair, TPC co-chair, and has held other leading roles for many international conferences. She has organised and chaired the 6G Network workshop in IEEE ICC 2020. She is a Chartered Engineer (C.Eng.), a member of IET, and a certified Unmanned Aircraft System (UAS) Pilot.



**ARGYRIOS ZOLOTAS** (Senior Member, IEEE) received the B.Eng. degree (Hons.) from the University of Leeds, the M.Sc. degree from the University of Leicester, and the Ph.D. degree from Loughborough University. He was with the University of Lincoln, University of Sussex, Loughborough University, and Imperial College London. He was a Visiting Professor at Grenoble INP, from May 2018 to June 2018. He is currently a Reader at Cranfield University. His research interests include advanced control, systems autonomy, and applied AI. He is a fellow of HEA.



**CARL SHAW** received the Ph.D. degree in physics from Queen's University Belfast, in 1997. He worked in U.K. defense before leaving for the private sector, where he worked in the semiconductor industry for STMicroelectronics. Throughout this period, he worked on the electronic design, system architecture, and software of embedded systems. For the last 16 years, he has been active in software and hardware security and is currently a Co-Founder of Cerberus Security Laboratories Ltd., a U.K. security consultancy, where he advises global multinationals on electronic product cybersecurity and works closely with academic institutions researching secure hardware and embedded systems.

...



**DSPACE**

<https://dspace.org/>

## **Attack-detection architectural framework based on anomalous patterns of system performance and resource utilization - Part II**

**Aloseel, Abdulmohsan; Al-Rubaye, Saba; Zolotas, Argyrios; Shaw, Carl**

**2021-06-11**

Institute of Electrical and Electronics Engineers (IEEE)

<https://doi.org/10.1109/ACCESS.2021.3088411>

Aloseel A, Al-Rubaye S, Zolotas A, Shaw C. (2021) Attack-detection architectural framework based on anomalous patterns of system performance and resource utilization - Part II, IEEE Access, Volume 9, pp. 87611-87629