Physical Communication 52 (2022) 101685

Contents lists available at ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

Injection attack detection using machine learning for smart IoT applications

Tarek Gaber [a,b,e,*], Amir El-Ghamry [c,e], Aboul Ella Hassanien [d,e]

[a] School of Science, Engineering, and Environment, University of Salford, UK
[b] Faculty of Computers and Informatics, Suez Canal University, Ismailia 41522, Egypt
[c] Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
[d] Faculty of Computer and Artificial Intelligence, Cairo University, Cairo, Postcode 12631, Egypt
[e] Scientific Research Group in Egypt (SRGE), Egypt

ARTICLE INFO

Article history:
Received 29 August 2021
Received in revised form 18 February 2022
Accepted 10 March 2022
Available online 16 March 2022

Keywords:
Intrusion detection
Injection attacks
Feature selection
Smart cities
In door wireless network
Machine learning
Internet of Things

ABSTRACT

Smart cities are a rapidly growing IoT application. These smart cities mainly rely on wireless sensors to connect their different components (smart devices) together. Smart cities rely on the integration of IoT and 5G technologies, and this has created a demand for a massive IoT network of connected devices. The data traffic coming from indoor wireless networks (e.g., smart homes, smart hospitals, smart factories , or smart school buildings) contributes to over 80% of the total data traffic of the current IoT network. As smart cities and their applications grow, security and privacy challenges have become a major concern for billions of IoT smart devices. One reason for this could be the oversight of handling security issues of IoT devices by their manufacturers, which enables attackers to exploit the vulnerabilities in these devices by performing different types of attacks, e.g., DDoS and injection attacks. Intrusion detection is one way to detect and mitigate the risk of such attacks. In this paper, an intrusion detection method was proposed to detect injection attacks in IoT applications (e.g. smart cities). In this method, two types of feature selection techniques (constant removal and recursive feature elimination) were used and tested by a number of machine learning classifiers (i.e., SVM, Random Forest, and Decision Tree). The T-Test was conducted to evaluate the quality of this proposed feature selection method. Using the public dataset, AWID, the evaluation results showed that the decision tree classifier can be used to detect injection attacks with an accuracy of 99% using only 8 features, which were selected using the proposed feature selection method. Also, the comparison with the most related work showed the advantages of the proposed intrusion detection method.

1. Introduction

The global cybersecurity market is estimated to reach $270 billion by 2026 as reported by Forbes [1]. IDC International Data Corporation [2], also predicts there will be 55.7 billion connected devices by 2025, of which 75% will be connected to the IoT. In addition, IDC reported that by 2025, the data generated by IoT devices would reach 73.1 zettabytes which would be massively increased by three times of the data generated in 2019 (18.3 zettabytes). Using the current communication network 4G/5G and the future generation (6G), the IoT model is foreseen to become the typical service-centric computing. It will play a large role in building what is called Smart City 4.0 including smart homes, smart hospitals, smart factories etc.

Smart cities rely on the integration between IoT and 5G technologies and this has created a demand for a massive Smart cities rely on the integration of IoT and 5G technologies, and this has created a demand for a massive IoT network of connected devices. Indoor wireless networks (e.g., smart homes, smart hospitals, smart factories, or smart schools) account for more than 80% of total data traffic on the current IoT network. The high volume of data traffic may lead to high risk of security problems such as denial of service, data integrity, etc. One application of the indoor wireless network is a smart hospital. Imagine such a smart hospital being attacked by a DoS (ransomware) attack. Patients could die because the service has been stopped. Recently, in September 2020, while a patient was hospitalized in a smart hospital in Germany, a ransomware attack was mounted on this hospital. Consequently, the service was stopped, and the patient died [3].

* Corresponding author at: School of Science, Engineering, and Environment, University of Salford, UK.
E-mail addresses: t.m.a.gaber@salford.ac.uk (T. Gaber), amir_nabil@mans.edu.eg (A. El-Ghamry), aboitcairo@cu.edu.eg (A.E. Hassanien).
URL: http://www.salford.ac.uk/our-staff/tarek-gaber (T. Gaber).

https://doi.org/10.1016/j.phycom.2022.101685

Generally speaking, with the development and adoption of the IoT model in indoor and outdoor wireless-based applications, there is a significant risk of abuse of IoT services. This could be done by aiming devices installed at the edge of the network, where personal, critical, or semi-critical data could be illegitimately captured and maybe leaked [4,5]. By exploiting the weaknesses of wireless networks, a huge number of attackers could attempt to illegally obtain personal information from target users or get illegitimate access to specific resources. One of the weaknesses is the open nature of these networks. This could be exploited by attackers to easily intercept and collect personal or sensitive data from a particular data traffic. There are already various security solutions for wireless network (including indoor networks) to minimize the security risks and ensure secure communications between senders and receivers. Examples of these solutions include: WEP (Wired Equivalent Protection), WPA (Wi-Fi Protected Access), and WPA2 protocols [5]. No matter how much progress has been made, these solutions/technologies are still suffering from different vulnerabilities, making them insufficient to fully communicate reliable data.

Because they continuously monitor computer networks and block malicious incoming traffic, intrusion detection systems (IDSs) play a critical role in the cybersecurity domain. IDSs are built in order to supplement security protocols, such as the above ones. IDSs aim to provide constant monitoring and identification of cyber attacks throughout a network's operational life. An IDS is a typical security protection mechanism that detects suspicious activity in a system and intercepts the attacking source in real time in order to protect the network [6]. IDSs are generally classified into two main classes based on the types of cyber data that are available: host-based detection and network-based detection. The host-based one refers to a system that detects intrusion within standalone electronic devices, e.g., laptops or smartphones, from their resources such as logs, disc space, and file systems. On the other hand, a network-based detection system is a type of IDS aiming to detect attacks by continuously analysing the traffic exchanged between network devices and the internet [7]. Wireshark is a well-known tool that is usually used for network traffic analysis to identify anomalies.

In this paper, we will focus on network-based intrusion detection systems for monitoring and analysing malicious traffic in IoT-based communication, which is the main infrastructure of many indoor wireless-based applications such as smart hospitals, smart manufacturing, and smart industries [8]. A good design of a network-based IDS should be able to identify and detect various attacks on a wireless IoT network [7]. These attacks could be a wide range of attacks, which include DoS/DDoS attacks, injection attacks, impersonation attacks, and flooding attacks. Because of the breadth of these attacks, the current network-based IDSs are still unable to detect all these attacks accurately or satisfactory [9]. In particular, this paper will focus on the detection of injection attacks that aim to replace (or inject) original code/data with malicious code/data. If these attacks are mounted, they could have a significant impact on the system's decisions (e.g., controlling the devices).

In this paper, an intrusion detection method was proposed to detect injection attacks in IoT applications. In this method, two types of feature selection techniques (constant removal and eecursive feature elimination) were used and tested by a number of machine learning (ML) classifiers (i.e., Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT)) on a public dataset, AWID.

The main contribution and novelty of this paper is fourfold. Firstly, based on the best knowledge of the authors, this paper is the first to suggest machine learning-based IDS for the detection of injection attacks in smart city applications. Secondly, employing two feature selection technique, constant removal and eecursive feature elimination (RFE) to get the best discriminative features that produced a high detection accuracy (99%) of the injection attacks while using fewer features (8 features). Using a fewer number of attributes could enable to decrease computational time overhead, which makes our methodology resource friendly for the IoT limited capabilities devices. Thirdly, the paper reports the 8 features that are used to detect the injection attacks from the AWID dataset. This would help the practical adoption of our method. Fourthly, the proposed method was evaluated using various evaluation metrics: accuracy, recall, precision, and F1-score and compared with the most related published work.

The rest of this paper is organized as follows. Section 2 highlights the impact of injection attacks on the Smart IoT environment. The related work is analysed in Section 3. The proposed intrusion detection method and its evaluation are presented in Sections 4 and 5 respectively. Finally, the conclusion is highlighted in Section 6.

2. Injection attack and smart IoT environment

Smart cities, smart hospitals, and smart factories are important applications of IoT technology that are growing rapidly all over the world. The basic principle in smart cities/hospitals/factories is to connect all their components (called "smart devices") using wireless sensor devices. As the demand on the smart devices grows, the manufactures may oversight to hand security issues of these smart devices. Consequently, enabling attackers to exploit vulnerabilities in the IoT smart devices by performing various types of malicious activities such as DDoS attacks, injection attacks, impersonation attacks, and flooding attacks to uncover user personal information from the IoT sensitive applications such as in the health care system, there is a need for high-security requirements [10].

Injection attacks refer to a wide range of attack spectrums through which an adversary can submit various types of input (code or data) to an application. This input is interpreted by the processor, considering it a legitimate query or command, and so executing it, generating unexpected results. Injection attacks can target web applications, the users that own the applications' data, and other connected applications and services. Injection attacks can have catastrophic consequences in a smart home environment using indoor wireless technology. For example, smart assistants with voice control systems such as Google Home, Alexa, Portal and Siri can actually open the main door of a smart home for an adversary without permission from an authorized user. This can take place by executing a remote command injected into the voice-controlled system. Additionally, smart home devices such as smart TV, smart thermostats, smart lights, and baby monitors can be easily compromised via wireless network injection attacks such as malicious packet injection [11].

To further illustrate the impact of an injection attack on IoT applications, a brief overview of the general architecture of the IoT paradigm is needed. The IoT architecture, as illustrated in Fig. 1, consists of four basic layers: perception, network, middleware, and application layer. The Perception Layer is the one that is responsible for managing smart devices across the system. The network layer is the one that transfers data between cloud nodes and IoT devices. The middleware layer is to provide an abstraction between IoT details and user applications. Finally, the application layer is responsible of the analytics, device control, and reporting to users [12,13].

Each of these layers can be threatened by different types of injection attacks. In the perception layer, there is the code injection attack that allows an adversary, who resides near to an IoT node, to inject malicious code into any victim node. This

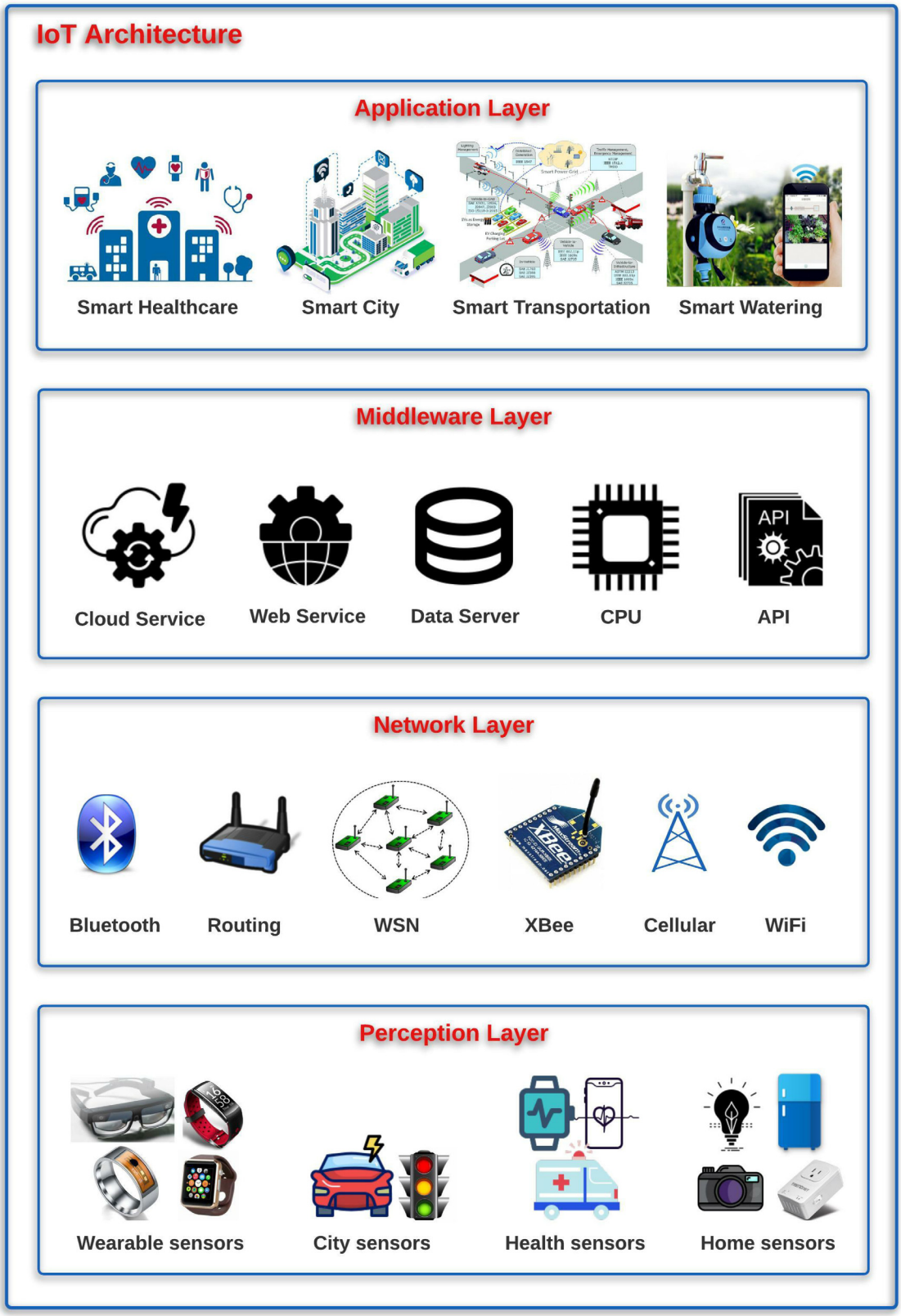

Fig. 1. IoT architecture.

enables the adversary to tamper with the nodeś firmware. The adversary can compromise the entire IoT communication network by compromising just one node [14]. In the network layer, there is the malicious code injection attack, in which an adversary relies on hacked IoT devices to inject malicious codes into the infrastructure of the IoT network. After the adversary succeeds in compromising the infrastructure, he/she can gain control of the overall IoT system. In addition, the network layer can be exposed

to a malicious node injection attack in which the adversary instals a malicious node between two or more nodes and monitors the traffic to and from the nodes [14]. The malicious node injection attack is a very dangerous attack as it can abort and cause the data to be modified. This type of attack is also known as Man-in-The-Middle-Attack [15]. Different network layer attacks can cause key/keystream retrieving attacks, in which the attacker monitors specific network packets to crack a shared secret key or to use the keystream to forge and inject packets into the network as an initial step for more different attacks, such ARP, ChopChop and Fragmentation attacks [7].

ARP attack is a common and effective way to disturb IoT networks. The ARP attack enables adversaries to intercept and change data exchanged between victim nodes in a smart IoT network. In this attack, the adversary transmits a large number of small data frames over a local IoT network for a particular period of time, causing the network to send an appropriate response. Then, the generated responses, captured by the attacker, can be fed to an offline key cracking algorithm to get the network key. Another consequence of the ARP attack is that it allows the attacker to intercept information intended for a given IoT device. This can then cause the collection and modification of the private data during the network transmission. This could happen because the ARP attack helps to establish a link between the MAC address of the adversary and the IP address of the IoT device [7].

In a ChopChop attack, the adversary can derive large portions of the keystream, which allows him/her to create and embed malicious frames into the network. This can be done by chopping the last byte of the encrypted part of the packet, replacing this byte's values, and then recomputing the encryption checksum. The changed packets are subsequently transmitted to the local access point, which may dismiss these packets until the attacker substitutes an acceptable checksum [7]. A similar keystream retrieval attack is the fragmentation attack, in which an intruder can exploit the design flaw in the Wi-Fi's frame fragmentation feature to execute a keystream retrieval attack. The fragmentation functionality helps to increase the reliability of a frame by splitting larger frames into smaller ones. The smaller frames from the same block are encrypted using the same key. This sends fewer messages than the ChopChop attack to extract a large portion of the keystream. Then, the attacker can use the keystream to inject malicious packets into the network. [7]. Another attack scenario of Wi-Fi technology (i.e., indoor wireless network) is the fragment cash attack, which exploits Wi-Fi's frame fragmentation feature. This attack could be mounted because Wi-Fi devices, disconnected from the network, retain non-reassembled fragments in their memory. An attacker could then exploit this vulnerability by injecting a malicious fragment into the memory of the access point.

In the application layer, all IoT infrastructures include a central control application-based interface, enabling users to control IoT devices. If this interface is not protected properly, cross-site scripting attacks can be mounted. Additionally, Denial of Service attacks can occur as a consequence of a code injection attack. So, an adversary can congest the IoT application interface by injecting code, which results in a denial of service for the users [15].

In our work, we focus on the detection of injection attacks (ARP, ChopChop and Fragmentation) that target the network layer in the IoT smart environment. So, we used a training dataset, AWID, that includes instances of fragmentation and ARP injection attacks. In addition, we used a testing dataset of AWID data, which contains ChopChop attacks in addition to fragmentation and ARP attacks.

3. Related work

Generally, the intrusion detection area has found an attraction among many researchers due to its importance in protecting networks. For wireless sensor networks and IoT applications, there is lots of published work suggesting solutions/approaches for intrusion detection. Batiha et al. [16] has recently proposed an intrusion detection system based on a neural network, mainly focusing on the impact of the learning process on the classification accuracy and energy consumption. In [17], the authors integrated the SVM classifier with the binary grey wolf optimizer and proposed an IDS for weirless sensor networks. Using, the datast, NSL KDD'99, the proposed IDS showed an enhanced performance in terms of the number of features used, accuracy, false alarm and detection rate. Using the Aquila optimizer and the DL (Deep Learning) technique, CNN, the authors in [18] designed a novel feature extraction and selection techniques for the IDS system. The developed IDS technique was tested on four well-known public datasets, including CIC2017, BoT-IoT, NSL KDD, and KDD99. According to the evaluation results, the created approach has a high level of performance. An IDS technique based on Fruitfly optimization and deep Autoencoder was developed by Sekhar et al. [19]. They employed fuzzy C-Means to deal with the missing data in the datasets and then they used the deep Autoencoder to extract robust features, which were then given to the back propagation neural network for attack categorization. Furthermore, the hidden layers of the Autoencoder were optimized using the Fruitfly algorithm. Using data from the NSL-KDD and UNSW-NB15 datasets, the evaluation showed that the proposed method performed well.

As the proposed IDS system was evaluated using AWID dataset [7], in this section, we provide discussion of some relative work that evaluated also using the AWID dataset. In particular, the selected related work discusses the impact of feature selection phase on the detection accuracy. This work is classified into machine and deep learning based work as below.

Machine learning-based work: Kolias et al. [7], the creators of the AWID dataset, applied several traditional ML algorithms (AdaBoost, OneR, J48, Naïve Bayes, Random Forest, ZeroR and Random Tree) to find the best classifier for detecting the intrusions. They achieved an accuracy of 96.20% using J48 as a classifier, considering the complete set of 154 features and 96.26% when reducing the number of features to 20. Also, Thanthrige et al. [20] proposed an IDS in which feature selection was done by information gain and Chi-Squared statistics methods. Then they used different sizes of features vectors (111, 41, and 10 features). The performance evaluation demonstrated that the accuracy increased from 92.17% to 95.12% with 41 features when applying Random Tree as a classifier. Aminanto et al. [21] proposed an approach to enhance the detection of impersonation attack in wireless networks. For the feature selection, they used a weight-based feature selection method, and for the classification, they adopted three machine learning (ML) techniques: ANN, Decision Tree, and SVM. The evaluation showed that the best performance was 99.86% using the SVM method while training 11 selected features. However, they did not consider the other two attacks (injection and flooding attacks). Kaleem et al. [22] proposed an IDS approach based on ANN classifier and cognitive feature ranking technique. Their approach classifies every instance as normal and attack classes. Their approach applies a cognitive feature ranking approach based on feature space analysis. Then they performed a pruning process on the input neurons of the first layer of the ANN to eliminate neurons that corresponded to irrelevant features. This approach showed an improvement in accuracy from 97.84% to 99.3% when reducing the number of features from 154 to 6.

Table 1
Literature review summary.

Reference	Classifier type	Feature selection/extraction	Classifier algorithm	Attack	Num of features	Addressed data imbalance	Accuracy
[21]	Binary	ANN, SVM,C4.8	ANN	Impersonation	11	Yes	99.86%
[20]	Multi-class	Information gain, chi squared statistics	OneR, Ada Boost, J48, Decision tree, Random Forest	Imperson-ation,Flooding,Injection	111, 41,10	No	95.12% for 41 features
[22]	Binary	Feature space analysis	ANN	Group attacks in one class	6	No	99.3%
[23]	Multi-class	SAE	SAE	Imperson-ation,Flooding,Injection	154	No	98.67%
[7]	Multi-class	Theoretic-based Manual Attribute Selection	AdaBoost, OneR, J48, Naïve Bayes, Random Forest, ZeroR	Imperson-ation,Flooding,Injection	154, 20	No	96.26%
[24]	Binary	SAE, Mutual Information, C4.8	SVM	Impersonation	5	Yes	98.22%
[25]	Multi-class, Binary	DL	DL	Imperson-ation,Flooding,Injection	95	Yes	98.54% for multi-class 99.52% for binary
[26]	Binary	SAE	K means	Impersonation	50	Yes	94.81%
[27]	Multi-class	Preprocessing	SAE	Imperson-ation,Flooding,Injection	71	Yes	92.49%
[28]	Multi-class, Binary	Preprocessing	Ensemble Learning	Imperson-ation,Flooding,Injection	36	No	95.88% for multi-class
[29]	Binary	SAE,OneR,SVM,CFS	MLP	Impersonation	21	Yes	99.97% semi-distributed, 97.80% for distributed
Proposed	Binary	Eecursive feature elimination, constant removal	Decision Trees, Random Forest, SVM	Injection	76,13,8	Yes	99%

Vaca et al. [28] introduced an IDS for Wi-Fi networks using an ensemble learning method. For feature reduction purposes, they removed the features having the same values in all rows of the dataset or having missing values in more than half of its records. This process reduced the number of features to 36. Using Ensemble Learning, the classification results of the attacks, impersonation, flooding, injection, and normal, showed that the proposed method achieved an accuracy of 95.88% when all the attacks were grouped into one class. Rahman et al. [29] proposed an IDS based on ML for the resource-constrained IoT that focused on the impersonation attack only. In this work, IoT devices are divided into semi-distributed systems and distributed systems. For feature extraction, they used the stacked autoencoder technique. Different feature selection approaches were applied on each partitioned dataset to select the top 7 features from the original features. They utilized the MLP (multi-layer perceptron) classifier for classification. The evaluation results showed that their work achieved an accuracy of 99.97% in semi-distributed model, and 97.80% for distributed models using 21 features. Lee et al. [24] implemented a three-phase framework for wireless IDS that focused on impersonation attack detection. In the first phase, SAE was used for feature extraction while in the second phase, the Mutual Information and C4.8 algorithms were applied, and finally, the SVM with gradient descent optimization was used for classification. They reported an accuracy of 98.22% with a set of five features.

Deep learning based work:

Thing [23] suggested an IDS based on the deep learning model (using stacked auto-encoder (SAE)) to classify between 4 classes (3 attacks — impersonation, flooding, injection, and normal traffic). The authors suggested two variants of the model: one with two hidden layers, and the second with three hidden layers. Without explicit feature selection, the one consisting of two hidden-layer achieved a better accuracy, 98.67%, than the one with 3 layers. Wang et al. [27] proposed yet another deep learning-based IDS approach. This approach aims to detect and classify wireless network traffic into four categories: normal traffic, impersonation, injection, and flooding attacks. From a data preprocessing, removing features with zero variance and missing values, 71 attributes were obtained. For the classification/detection phase, two variants of the deep neural network were suggested: one with three hidden layers and another with seven hidden layers. The best accuracy, 92.49%, was achieved using the one with the 7-hidden-layer. Also, Kim et al. [26] employed a deep learning approach (stacked auto-encoder) and unsupervised clustering to detect impersonation attacks from wireless traffic. The proposed approach consists of two main steps: feature extraction using a stacked auto-encoder with two hidden layers, and unsupervised clustering using the K-means algorithm. They achieved accuracy of 94.81% when using 50 attributes. Furthermore, Ran et al. [25] proposed an IDS for wireless networks using a semi-supervised deep learning to classify the network traffic into four types of flow: normal, impersonation attack, flooding attack, and injection attack. They adopted a ladder network approach whose architecture consists of a stacked auto-encoder with a clean decoder unit, a noisy encoder unit, and a decoder unit. To increase the generalization ability of the proposed method, they employed a focal loss function. The accuracy of the classification results was 99.77 percent, 89.32 percent, 73.41 percent, and 82.79 percent for normal, impersonation, flooding, and injection, respectively. This method's overall accuracy was 98.54 percent.

Table 1 summarize the difference between the work in the literature and our proposed work.

4. Proposed method

In this section, we present our approach, which aims at improving the accuracy of the detection of injection attacks while using fewer features. The proposed method consists of four steps, as shown in Fig. 2. Firstly, the data is collected. In our work, to achieve the first step, we used the AWID dataset [7], which contains real Wi-Fi traces. Secondly, the cleaning and preprocessing of the data are done. To achieve high performance when using the AWID dataset, we should perform cleaning and preprocessing operations. The preprocessed dataset will be the input to the classifiers. In this case, the number of features used in the classification after the data cleaning process is 76 features.

In the third step, feature selection is achieved. The preprocessed dataset is fed into feature selection methods before the classification phase. Feature selection methods that include constant removal and eecursive feature elimination result in a reduced set of important features to detect injection attacks. When using the constant removal technique, the number of features to be used is 13. When using both constant removal and eecursive feature elimination, the resulting number of features reduced to 8 features. In the final step, the classification using SVM, Decision Tree, and Random Forest was conducted, and the performance was evaluated using several measures (see the results section for more details).

4.1. Cleaning and preprocessing

AWID dataset contains continuous and discrete data types as well as symbolic types with a flexible value range. In addition as any intrusion detection dataset, the AWID dataset is an imbalance dataset. Using the raw dataset will be difficult for the learning process in most pattern classification approaches. Therefore, cleaning and preprocessing techniques should be applied first. The cleaning and preprocessing phase includes a set of steps that make the raw dataset convenient for subsequent phases. Algorithm 1 summarizes these steps.

Algorithm 1 Dataset Cleaning and Preprocessing

```
RawDataset = ReadDataset(Path)
procedure CLEANING_PREPROCESSING(RawDataset)
    Select rows with normal and injection labels
    Features, Labels = split dataset to fearures and labels
    Encode(Labels)
    dataset = Replace(Features,"?",None)
    for each column col in dataset do
        if nullvalues ≥ 50% then
            delete(col)
        end if
    end for
    for each row r in dataset do
        for each value v in r do
            if v = null then
                delete(r)
            end if
        end for
    end for
    for each datainstance in dataset do
        cast into integer value
    end for
    Normalize(dataset)
end procedure
procedure BALANCE(TrainDataset)
    Pick 10% instances of normal instances randomly
end procedure
```

The AWID dataset contains almost all possible 802.11 fields. In this dataset, the missing values are represented by question marks *("?")*. This means that the dataset contains the question mark ("?") for unavailable values for the corresponding attributes. To deal with this question mark (i.e., clean the dataset), we replaced the question mark with a null value (a distinguishing value from other normal values in the dataset), as reported in [30]. We then dropped the columns (features) with 50% of the null data. Then we deleted the rows that contained at least one null value. Following that, a set of type casting is performed such as non integer values being casted into numbers. The symbolic values (such as transmitter, destination, and receiver) are mapped into integers which are scaled to a value between 1 to max, where max is the number of the symbolic features. The hexadecimal values (such as the value of the Integrity Check and the WEP Initialization Vector) are also mapped into integers. Also, the target class is mapped into binary classes: 0 for normal, 1 for injection attacks. Finally, we linearly normalized the values of attributes to values between 0 and 1. By this, the data should be properly formatted and ready for the classification process.

Addressing AWID imbalance problem: As any intrusion detection dataset, the AWID dataset is an imbalance dataset. As given in Table 3, the training dataset contains 1,795,575 million instances and 154 features. It is clear that the number of records in the normal target class is very large compared to that of any other attack class. Considering the injection attacks, the main focus of this paper, it can also be noticed that there is an imbalanced data problem: the normal class has 1,633 190 instances while the injection class has 65,379 instances only. To address this problem, two preprocessing steps were taken.

Firstly, following the work in [5], we selected 10% of the normal instances randomly, and 65,379 instances for the injection attack class, as shown in Table 5. Selecting 10% from the normal class instances makes the number of normal instances is more than the number of injection attack instances, which maintains the real life scenario in which the normal flow exceeds the injection attack flow in real networks.

Secondly, as shown above, data cleaning and preprocessing were executed on the training dataset. The cleaning and preprocessing steps reduced the number of instances of the normal class to 68,074 instances while leaving the number of injection instances the same as given in Table 6. From this table, it can be seen that the ratio between the normal and injection classes is 1:1, i.e., producing balanced data. In all our experiments below, we used the balanced training dataset in the training phase, while in the testing phase we used the unbalanced test dataset as explained in the evaluation Section 5.3.

4.2. Feature selection

The cleaned and pre-processed data is then given to the feature selection step. Feature selection is the process of reducing data dimensionality in order to enhance machine learning performance. The goal of feature selection is to get rid of useless or redundant features and keep only those that are important and help improve machine learning performance [31]. Further this also would lead to decreasing computation time. In the proposed method, we used two techniques to choose the best features: constant removal and eecursive feature elimination.

Constant removal is a simple feature selection approach that belongs to invariant filter-based methods which can be fed into any machine learning (ML) models. It can be categorized as variance-based feature selection. It is also known as "variance threshold feature selection". The constant removal technique ranks features based on specific criteria and selects the top N from these ranked features. It works by removing all zero-mean features, which have the same value for every instance.

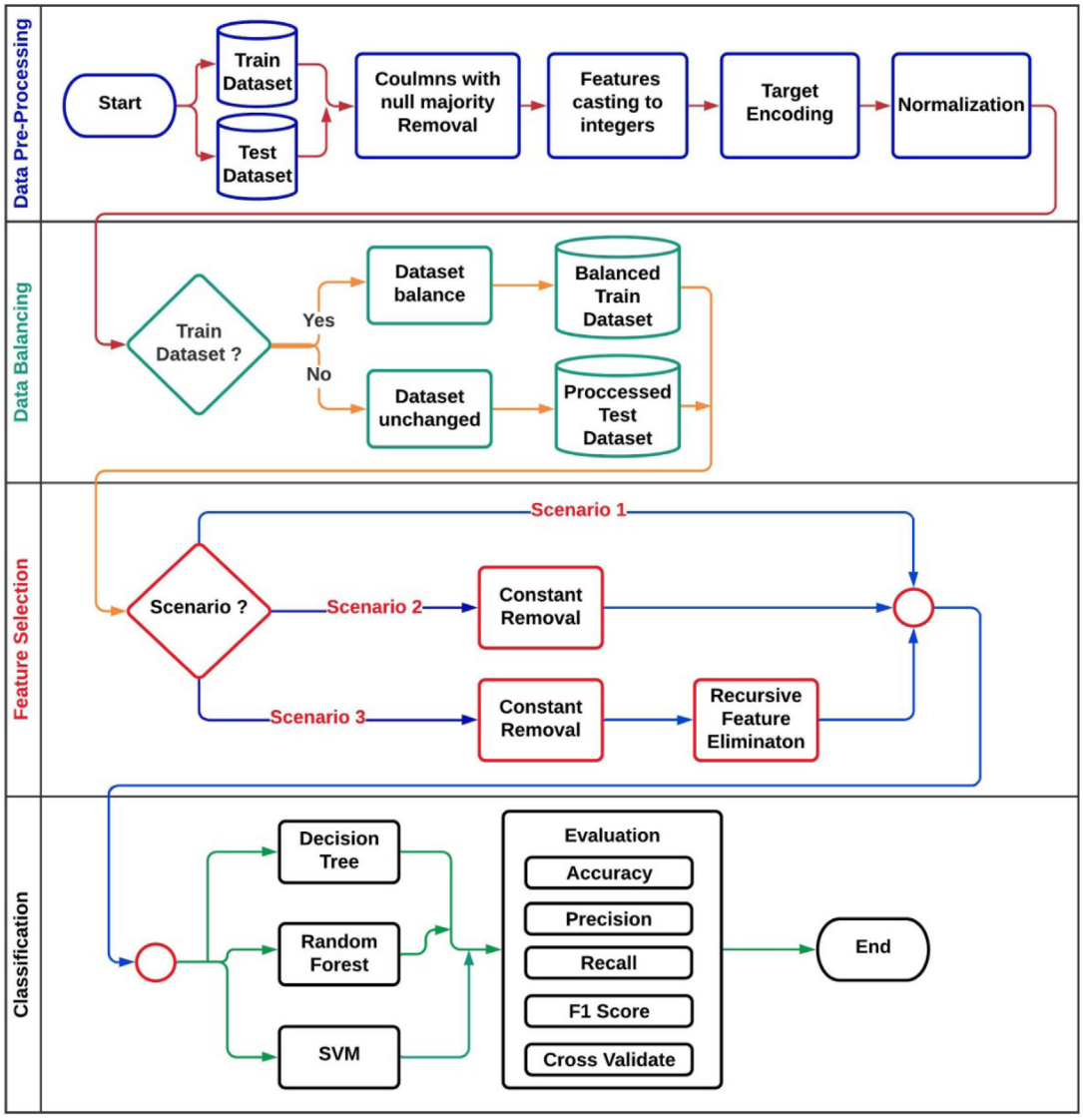

Fig. 2. Proposed approach.

It also removes the columns (features), where the differences between their values are very small (i.e. the varianceis less than a particular threshold). In both of these situations, the data would not contribute to the classification phases because it could not help distinguish a tuple with a normal class from tuples with the injection attack class [31].

Recursive feature elimination (RFE) is a wrapper-based feature selection approach that uses a search mechanism to reduce the feature set. This method gives more accurate results than filter-based methods. The RFE technique works by first training an initial feature set, then the feature weights are computed, and finally, the features with the lowest weight values are removed from the feature set. This process is executed recursively until the relevant features are obtained [32,33]. We adopted the RFE approach because of its flexibility and ease of use. The algorithm can produce the best possible set of features that gives the highest performance in less computation time. The major factor that makes the REF approach stable is the ML algorithm executed in every iteration. For this purpose, in this work, we adopted the random forest classifier as an ML model since the relationship between the features and the target is not linear. The random forest consists of multiple decision trees which compute the entropy of the features' values in the dataset and then partition the dataset samples layer by layer. Finally, the dataset instances are partitioned based on the target column [34].

As shown in Algorithm 2, the feature selection process in this paper has been done in three scenarios to show the impact of combining the constant removal and REF techniques. In the first scenario, there was no feature selection. After cleaning and preprocessing the data, we directly used a set of features for the classification. In the second scenario, we only applied the constant removal approach, which produced 13 features that were then used for the classification. In the third scenario, we applied the eecursive feature elimination (RFE) technique to the 13 features obtained from applying the constant removal approach. This produced 8 features which were then used in the classification process and gave the best results, as can be seen in Section 5. Based on these results, using the pipeline of constant removal followed by the RFE technique was reported as the feature selection method for our intrusion detection system for the injection attack. It is important to note that the metrics for the feature selection in this paper are based on two factors: (1) What is the difference between the classification accuracy attained with a certain feature set and the classification accuracy obtained with the best feature set? (2) What is the optimal number of features to employ?

Algorithm 2 Feature Selection

```
procedure FEATURE_SELECTION(preprocessed_dataset,scenario)
    if scenario = 1 then
        Apply_ML_Classifiers (preprocessed_dataset)
    else if scenario = 2 then
        selected_13 = Apply_ConstantRemoval(preprocessed_dataset)
        Apply_ML_Classifiers (selected_13)
    else if scenario = 3 then
        selected_13 =Apply_ConstantRemoval(preprocessed_dataset)
        selected_8 =Apply_RecursiveFeatureElimination(selected_13)
        Apply_ML_Classifiers (selected_8)
    end if
end procedure
procedure CONSTANTREMOVAL(Dataset)
    for each column col in Dataset do
        if col.values.mean() ≤ 1 then
            delete(col)
        end if
    end for
end procedure
procedure RECURSIVEFEATUREELIMINATION(Dataset) input: initial Feature set FList=(1,2,3,4, ..., n)
output: Rank list based on smallest weight criteria RList
    Set RList = {}
    do
        Train Random Forest using feature list (FList)
        Calculate weight vector
        Calculate rank criteria
        Find the features with smallest rank (f)
        Update Rank List features (RList) by adding rank of feature (f)
        Remove feature with lowest rank from feature list (FList)
    while FList ≠ null
end procedure
```

Table 2
The corresponding p-values of each significant feature.

1.	The variable frame.time$_e pochisstatisticallysignificant with a pvalue$ = 0.0
2.	The variable frame.time$_d eltaisstatisticallysignificant with a pvalue$ = 9.8$e-14$
3.	The variable frame.time$_d elta_d isplayedisstatisticallysignificant with a pvalue$ = 9.8$e-14$
4.	The variable frame.time$_r elativeisstatisticallysignificant with a pvalue$ = 0.0
5.	The variable frame.len is statistically significant with a pvalue = 0.0
6.	The variable frame.cap$_l enisstatisticallysignificant with a pvalue$ = 0.0
7.	The variable radiotap.mactime is statistically significant with a pvalue = 0.0
8.	The variable radiotap.datarate is statistically significant with a pvalue = 0.0
9.	The variable radiotap.channel.type.cck is statistically significant with a pvalue = 0.0
10.	The variable radiotap.channel.type.ofdm is statistically significant with a pvalue = 0.0
11.	The variable radiotap.dbm$_a ntsignalisstatisticallysignificant with a pvalue$ = 0.0
12.	The variable wlan.fc.type is statistically significant with a pvalue = 2.8$e-06$
13.	The variable wlan.fc.subtype is statistically significant with a pvalue = 0.0
14.	The variable wlan.fc.frag is statistically significant with a pvalue = 1.5$e-77$
15.	The variable wlan.fc.retry is statistically significant with a pvalue = 0.0
16.	The variable wlan.fc.pwrmgt is statistically significant with a pvalue = 2.2$e-11$
17.	The variable wlan.fc.moredata is statistically significant with a pvalue = 5.3$e-47$
18.	The variable wlan.duration is statistically significant with a pvalue = 0.0
19.	The variable wlan.frag is statistically significant with a pvalue = 6.5$e-60$
20.	The variable data.len is statistically significant with a pvalue = 0.0

To evaluate the quality of the feature selected by the proposed method, the T-Test was conducted. The idea is to evaluate features by comparing the p-value of each feature. The p-value will be computed from t-test. When performing a T-Test of the injection attack features, it was found that the set of features that are statistically significant for the injection attack classification are the 20 features listed with their corresponding p-values in Table 2. We further plotted these features, showing the most significant ones as illustrated in Fig. 3. Based on these results and the results discussed in Section 5, it can be remarked that all the 8 features selected by the proposed method above are statistically significant where their p-value equals zero.

4.3. Classification

In the classification step, as shown in the features selected in the previous step, in order to do performance validation on the set of selected features in each of the three conducted experiments, we adopt three classification algorithms, which are Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). We chose the DT as it is easy to interpret, understand, and visualize. It mimics the human thinking power that helps to make good interpretations. The DT can predict the class by learning decision rules and is not largely influenced by outliers. Also, it has no assumptions about space distributions and classifier structures

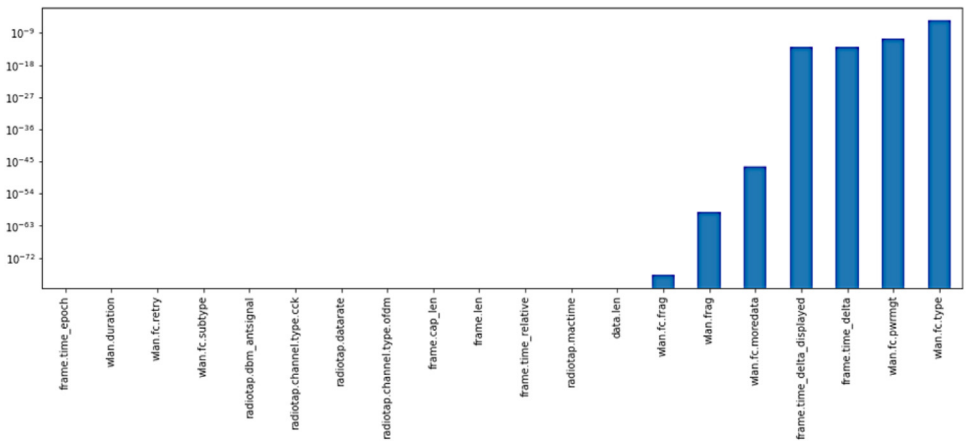

Fig. 3. A bar chart for sorted p-values of most significant features.

[35–37]. We also adopted the RF classifier because it is more robust and highly accurate. It creates multiple decision trees and selects the data sample randomly from the training set, obtaining the prediction from individual decision trees and getting the most votes. The high accuracy of the RF algorithm is based on the number of used decision trees, and it does not suffer from over-fitting [36,38]. Finally we used the SVM classifier due to its good generalization capabilities, which prevent it from over-fitting. The SVM provides a kernel technique that helps solve any complex problem. In this work, we used the linear kernel function. Furthermore, the SVM has a nature of Convex Optimization which is very helpful for optimal results [37,39].

5. Evaluation and discussion

To evaluate the proposed method, the AWID dataset was used, specifically "AWID-CLS-R-Trn" was used for training and "AWID-CLS-R-Tst" was used for testing purposes [7]. For validation purposes, we concatenated the two datasets which are then used to obtain the accuracy and other performance metrics as given below.

The AWID dataset [7] contains a set of records which resemble real traces of WiFi traffic. This dataset is categorized into two groups depending on the number and type of attack classes. The first category is called "CLS" which consists of four target classes. The second category is called "ATK" which consists of 16 target classes, which are the breakdown of (or the types of) the four attacks included in the "CLS" dataset. There is another categorization of the AWID dataset where it is divided into two types depending on the number of records: the first type is the full version, and the second one is the reduced version of the dataset. In this paper, we performed our experiments on the reduced version that falls into the "CLS" category of the dataset. The reduced version of AWID dataset is further divided into training (see Table 3) and testing datasets (see Table 4). For the training purposes, the datasets were cleaned and pre-processed as discussed in Section 4.1.

For the purpose of testing, the number of instances of each class label is shown in Table 4. As the scope of this paper is about injection attacks, we only selected injection attack and normal network data from the testing dataset. After applying the data cleaning and preprocessing techniques, as shown in Table 8, we have got a testing dataset with 277,960 instances from the normal class and 16,682 from the injection attack class). The experimental results and their discussions are given below (see Table 7).

Table 3
AWID training dataset with all classes.

Original training dataset	
Class labels	Number of instances
Normal	1,633,190
Injection	65,379
Impersonation	48,522
Flooding	48,484
Total size	1 795 575, 155

Table 4
AWID testing dataset with all classes.

Original testing dataset	
Class labels	Number of instances
Normal	530,785
Injection	16,682
Impersonation	20,079
Flooding	8097
Total size	575,643

Table 5
Balanced training dataset with normal and injection classes.

Balanced training dataset	
Class labels	Number of instances
Normal	150,500
Injection	65,379
Total size	215,879

Table 6
Cleaned and preprocessed training dataset of normal and injection classes.

Preprocessed training dataset	
Class labels	Number of instances
Normal	68,074
Injection	65,379
Total size	133,453

Table 7
Testing dataset with normal and injection classes.

Testing dataset	
Class labels	Number of instances
Normal	530,785
Injection	16,682
Total Size	547,467

Table 8
Preprocessed testing dataset with normal and injection classes.

Preprocessed testing dataset	
Class labels	Number of instances
Normal	277,960
Injection	16,682
Total size	294,642

5.1. Evaluation metrics

The performance of the proposed intrusion detect method has been evaluated using the confusion matrix consisting of: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) which are defined below.

- **True Positive (TP)**: # of records successfully recognized as injection attack.
- **True Negative (TN)**: # of records classified as a normal class.
- **False Positive (FP)**: # of records wrongly classified as injection attack
- **False Negative (FN)**: # of injection attacks undetected by IDS

Using the parameters of the confusion matrix above, the following evaluation metrics have been used in this study: Accuracy, Precision, Recall, and F1 metrics.

Accuracy: The ratio of number of records that are correctly classified to the total number of records. The higher the accuracy, the better the model applied.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: The ratio of the number of positive records that are correctly classified to the total number of positive records. This means that the lower the false positive rate, the higher the precision result. The precision measure is a good measure when the cost of a false positive is high.

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity): the ratio of the number of positive records that are classified correctly to the total number of classifications in the actual class. This means that the higher the FN rate, the lower the value of the recall. The recall measure is important when we want to select the best model in case the FN rate is very high.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: This measure is known as the harmonic mean of the precision and the recall measures. It is considered a good evaluation measure for imbalanced data.

$$F1 = \frac{2Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

It is important to note that we used the k-fold (and k = 10) cross validation process after we merged the training and testing datasets. Cross validation is used to evaluate the trained models by partitioning the dataset into n partitions (in our case, we use 10 partitions), and then using the *(n-1)* partitions for the training and the remaining partition for the testing process. We iterated this process by changing the testing partition at each iteration, while using the other partitions for training. As a result of this, the instances of the dataset will be tested at least once and trained *(n-1)* times.

Table 9
Parameter settings of decision tree algorithm.

Parameter	Explanation	Value applied
criterion	Measure the quality of a split	Gini impurity
max_depth	Maximum depth of the tree	Nodes are expanded until all leaves are pure
min_samples_split	Minimum number of samples required to split an internal node	1
min_samples_leaf	Minimum number of samples required to be at a leaf node	1

Table 10
Parameter settings of SVM Algorithm.

Parameter	Explanation	Value applied
c	Regularization parameter which shows that the strength of the regularization is inversely proportional to C	1.0
kernel	Specifies the kernel type to be used in the algorithm	Linear
tol	Tolerance for stopping criterion.	0.001
max_iter	Hard limit on iterations within solver or no limit	−1(No limit)

Table 11
Parameter settings of random forest algorithm.

Parameter	Explanation	Value applied
$n_estimators$	Number of trees in the forest	100
criterion	Measure the quality of a split	Gini impurity
max_depth	Maximum depth of the tree	Nodes are expanded until all leaves are pure
$min_samples_split$	Minimum number of samples required to split an internal node	2
$min_samples_leaf$	Minimum number of samples required to be at a leaf node	1
max_leaf_nodes	Maximum no of leaf nodes generated	Unlimited number of leaf nodes
$min_impurity_decrease$	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.	0.0

5.2. Parameter settings

Each of the adopted classifiers (DT, SVM, RF) has a set of parameters. It is worth mentioning the explanation and the values of these parameters, which were used in the experiments presented below. This is shown in Tables 9, 10, and 11.

5.3. Results and discussion

Three main experiments were designed to evaluate the proposed intrusion detection system. In the first experiment, the cleaning algorithm (Algorithm 1) was executed on both training and testing datasets. The output of this algorithm was 76 features which are listed in Table 12. These 76 features were given to

Table 12
AWID 76 features list.

AWID 76 Features			
1	(frame).(interface_id)	39	(radiotap).(flags).(wep)
2	(frame).(offset_shift)	40	(radiotap).(flags).(frag)
3	(frame).(time_epoch)	41	(radiotap).(flags).(fcs)
4	(frame).(time_delta)	42	(radiotap).(flags).(datapad)
5	(frame).(time_delta_displayed)	43	(radiotap).(flags).(badfcs)
6	(frame).(time_relative)	44	(radiotap).(flags).(shortgi)
7	(frame).(len)	45	(radiotap).(datarate)
8	(frame).(cap_len)	46	(radiotap).(channel).(freq)
9	(frame).(marked)	47	(radiotap).(channel).(type).(turbo)
10	(frame).(ignored)	48	(radiotap).(channel).(type).(cck)
11	(radiotap).(version)	49	(radiotap).(channel).(type).(ofdm)
12	(radiotap).(pad)	50	(radiotap).(channel).(type).(2ghz)
13	(radiotap).(length)	51	(radiotap).(channel).(type).(5ghz)
14	(radiotap).(present).(tsft)	52	(radiotap).(channel).(type).(passive)
15	(radiotap).(present).(flags)	53	(radiotap).(channel).(type).(dynamic)
16	(radiotap).(present).(rate)	54	(radiotap).(channel).(type).(gfsk)
17	(radiotap).(present).(channel)	55	(radiotap).(channel).(type).(gsm)
18	(radiotap).(present).(fhss)	56	(radiotap).(channel).(type).(sturbo)
19	(radiotap).(present).(dbm_antsignal)	57	(radiotap).(channel).(type).(half)
20	(radiotap).(present).(dbm_antnoise)	58	(radiotap).(channel).(type).(quarter)
21	(radiotap).(present).(lock_quality)	59	(radiotap).(dbm_antsignal)
22	(radiotap).(present).(tx_attenuation)	60	(radiotap).(antenna)
23	(radiotap).(present).(db_tx_attenuation)	61	(radiotap).(rxflags).(badplcp)
24	(radiotap).(present).(dbm_tx_power)	62	(wlan).(fc).(version)
25	(radiotap).(present).(antenna)	63	(wlan).(fc).(type)
26	(radiotap).(present).(db_antsignal)	64	(wlan).(fc).(subtype)
27	(radiotap).(present).(db_antnoise)	65	(wlan).(fc).(frag)
28	(radiotap).(present).(rxflags)	66	(wlan).(fc).(retry)
29	(radiotap).(present).(xchannel)	67	(wlan).(fc).(pwrmgt)
30	(radiotap).(present).(mcs)	68	(wlan).(fc).(moredata)
31	(radiotap).(present).(ampdu)	69	(wlan).(fc).(protected)
32	(radiotap).(present).(vht)	70	(wlan).(fc).(order)
33	(radiotap).(present).(rtap_ns)	71	(wlan).(duration)
34	(radiotap).(present).(vendor_ns)	72	(wlan).(frag)
35	(radiotap).(present).(ext)	73	(wlan).(seq)
36	(radiotap).(mactime)	74	(wlan).(fcs_good)
37	(radiotap).(flags).(cfp)	75	(wlan).(wep).(key)
38	(radiotap).(flags).(preamble)	76	(data).(len)

Table 13
Classification results using different set of features.

	Accuracy	Precision	Recall	F1 Score
	Results of using 76 features			
Decision tree	0.9681	0.6731	0.8500	0.7513
Random forest	0.9888	0.9511	0.8470	0.8960
SVM	0.9758	0.7008	0.9999	0.8240
	Results of using 13 features			
Decision tree	0.9908	0.9539	0.8807	0.9158
Random forest	0.9887	0.9507	0.8439	0.8941
SVM	0.9757	0.7001	0.9999	0.8235
	Results of using 8 features			
Decision tree	0.9891	0.9521	0.8500	0.8982
Random forest	0.9891	0.9537	0.8492	0.8985
SVM	0.9756	0.6989	0.9999	0.8227

the three classifiers (SVM, DT, RF) and results were recorded as given in Table 13. From this table, it can be noticed that the RF's results are the best with accuracy, precision and F1-score 98.88%, 95.11 and 89.60% respectively. While the SVM's results are the best in terms of recall with 99.99%. A high recall value is very important in real smart cities attack detection. If an injection attack instance is falsely categorized as normal network traffic, catastrophic consequences (e.g., denial of service) would affect most of smart cities applications. In this case, the SVM could be the best classifier to be used for detecting the injection attacks.

In the second experiment, after applying the cleaning algorithm (Algorithm 1), we then applied the constant removal feature selection to remove redundant values. This gave 13 features, which are listed in Table 14. We then used these 13 features for the classification processing using SVM, RF, and DT and the results are reported in Table 13. From this table, one could observe that the DT performed better than both the RF and SVM in classifying the injection attack. With the DT, the proposed method achieved an accuracy of 99.08 percent and an F1-score of 91.58%. The results are better than using all 76 features. This indicates that applying the constant removal method helped us to select the most discriminating set of features as well as improve the classification accuracy. Also, achieving a higher F1-Score (91.58%) means that the RF could be a good classifier for imbalanced data, as in the case of AWID.

From Table 13, comparing results of 76 and 13 features, it can be seen that the RF nearly produced the same results using 76 and 13 features respectively. like using 76 features, the SVM gave the highest recall with 99.99%. This indicates that the SVM classifier preserves its high performance while using a reduced set of features.

In the third experiment, the eecursive feature elimination (RFE) technique was applied on the output of the constant removal (i.e., 13 features). This produced the top 8 features, which are listed in Table 15. Giving these 8 features to the three classifiers (SVM, DT, RF), we got the results summarized in Table 13. From this table, it can be seen that the DT and RF gave the same highest accuracy value of 98.91%.

Comparing results of 76 and 13 features, as can be seen in Table 13, it can be noticed that using 8 features, (a) the DT and RF gave slightly better results than ones they achieved with 13 features, (b) the RF performed slightly better than its using 13 features. In addition, it can be seen that using only 8 features, the RF performed better in terms of all metrics. This indicates that the

Table 14
AWID 13 features list.

AWID 13 Features	
1	(frame).(time_epoch)
2	(frame).(time_relative)
3	(frame).(len)
4	(frame).(cap_len)
5	(radiotap).(length)
6	(radiotap).(mactime)
7	(radiotap).(datarate)
8	(radiotap).(channel).(freq)
9	(wlan).(fc).(type)
10	(wlan).(fc).(subtype)
11	(wlan).(duration)
12	(wlan).(seq)
13	(data).(len)

Table 15
AWID 8 features list.

AWID 8 Features	
1	(frame).(time_relative)
2	(frame).(len)
3	(frame).(cap_len)
4	(radiotap).(mactime)
5	(radiotap).(datarate)
6	(wlan).(fc).(subtype)
7	(wlan).(duration)
8	(data).(len)

feature selection proposed in this paper can help in detecting the injection attack using a few features.

The experiments and the results above show that the proposed feature selection method can reduce the features of the injection attack from 76 to 8 while achieving a very good detection rate (0.9891) using the RF classifier, which also achieved high precision, recall, and F1-Score by 95.37%, 84.92%, and 89.85% respectively. These results are very useful when adopted for smart IoT applications such as indoor wireless applications (e.g., smart hospitals, smart factories) and outdoor applications (e.g., smart transportation) in smart cities. All such applications require a huge number of WI-FI network connections between various types of smart devices to exchange information between them. With 8 features out of 76, our proposed method can detect injection attack activities with little processing time and a high detection rate.

These results further showed that the aim of our work (detecting injection attacks with high accuracy while using fewer features) has been achieved. The results proved that our method achieved a detection rate of 99% and F1-Score 90% using only 8 features. Using fewer numbers of attributes means less computational cost is required for attack detection, which makes our method resource friendly. In other words, the proposed method is appropriate for the resource-constrained IoT devices that are the basic building blocks of smart hospitals, smart factories, smart buildings, and smart cities. Also achieving an F1-Score of 90% is a good indicator that our proposed method can handle the imbalanced data expected in a network intrusion detection problem where most of the data is normal traffic.

5.4. Comparison with literature

To further evaluate the proposed intrusion detection method, we conducted two types of comparisons: one for the feature selection and one for the performance of the intrusion detection method.

Feature Selection Comparison: The proposed feature selection method was compared with two well-known feature selection techniques: mutual information [40] and Extra Tree (Extremely Random Tree Classifier) [41]. Mutual information is a technique that applies information gain to select a feature subset from an original feature set. In Extra Trees, each individual tree has a random sample of a specific number of features, and the algorithm selects the best feature to split the data. The comparison was conducted on the best 8 features selected using mutual information and Extra Tree. This is because our method gave the best results when all 8 features were used (see Table 13). The results of this comparison are summarized in Tables 16 and 17. From these two tables and Table 13, it can be noticed that the proposed feature selection method (with the DT classifier) is still better than the results of the 8 features selected by mutual information and Extra Tree techniques.

Table 16
Mutual information-based results of 8 features.

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision tree	0.9889	0.9469	0.8507	0.8948
Random forest	0.9821	0.9535	0.8431	0.8932
SVM	0.9684	0.6432	0.9936	0.7809

Table 17
Extra Tree-based results of 8 features.

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision tree	0.9823	0.9447	0.8507	0.8939
Random forest	0.9860	0.9238	0.8418	0.8862
SVM	0.9744	0.6990	0.9998	0.8138

Intrusion detection comparison: This section presents and discusses a comparison with work in the literature that is related to our proposed method.

The work in [25] utilized a ladder network that is based on Stacked Auto-Encoder (SAE) to select optimal attributes from a list of features and then to perform network traffic classification into normal, impersonation attack, flooding attack, and injection attack. Using 95 attributes from the AWID dataset, the authors reported 98.54% overall accuracy, but for the injection attack detection, they reported 82.79% accuracy. In addition, they only used one experimental scenario for feature selection, which is based on deep learning. They mentioned that more layers of the deep learning algorithm should be added to improve the accuracy improvement. However, this would limit the adoption of their model in the IoT environment as it would be computationally expensive on top of its poor accuracy (82.79%). The work in [27] to detect intrusion, the authors followed a similar preprocessing approach by removing features with missing values and zero variance, which led to a set of 71 features. Then they applied a Deep Neural Network (DNN) model with seven hidden layers. However, they did not apply any known feature selection technique, just employed the preprocessing steps above to perform feature reduction. Although they achieved slightly higher classification accuracy (99.99%) of injection attacks, which is around 1% higher than ours, the use of the DNN algorithm with a seven-layer structure would incur more computational time, thus making it difficult to adopt in smart IoT applications. In the work [28], the authors utilized ensemble learning algorithms and performed feature analysis and correlations to remove and combine features, which ended up with 18 features. For injection attacks, they achieved 44% precision, 93% recall, and 60% F1-Score. Like the work in [27], their results could require more computational power than ours, as the former needs 18 features while the latter requires 8 features only.

As shown in earlier, the proposed intrusion detection method achieved 99% accuracy, 95% precision, and 90% F1 score using just 8 features through the decision tree classifier. Compared with all the related work above, it could be remarked that our proposed method achieves nearly the same accuracy while achieving better

results in terms of precision and F1-Score, and it only needs 8 features to achieve these results, which makes it more appropriate in a smart IoT environment.

Time_dependent features: According to the authors in [24], any IDS system should be independent on any temporal features. This makes it difficult to obtain the values of these features in practical applications. Additionally, if these features are used in the training phase of ML, then the trained model will remember the timing of each attack. Hence, the detected attacks in testing set are dependent on time-dependent information learned during the training phase. Our proposed method partially conforms to this recommendation. For example, consider the scenario of 13 features, the constant removal approach excludes: (frame).(time_delta) and (frame).(time_delta_displayed) features. While in the scenario of 8 features, the RFE technique excludes: (frame).(time_epoch) features before the classification step. Thus the final 8 features includes two temporal features which are (frame).(time_relative) and (radiotap).(mactime). The work in [5, 42] utilized all three temporal features: (frame).(time_epoch), (frame).(time_relative), and (radiotap).(mactime). While the work in [24] has only reported temporal feature (frame).(time_epoch), but it uses it for impersonation attack detection. The work in [25] and [27] did not mention whether they rely on temporal features or not when doing the classification task. While the work in [28] eliminated all time_specific features. Their evaluation metrics is lower than our results using more number of features, and their work lacks data balancing. So, it can be said that our proposed injection attack detection method conforms to the temporal features recommendations reported in [24] about building any IDS system.

In terms of applications, our proposed method can find many applications in the IoT field. For example, the operation of smart cities is mainly based on wireless communication to transfer an enormous amount of information between smart IoT devices. Thus, the success of smart city applications such as smart irrigation systems, smart parking systems, smart lighting, smart waste management, and smart air quality management is subject to the pivotal development of security architectures. Our methodology can help to secure these applications by facing injection attacks such as ChopChop, fragmentation, and ARP attacks, which target the network layer in IoT smart city architecture.

6. Conclusion

An injection/intrusion attack detection method was proposed in this paper to address the problem of detecting these attacks in IoT applications, including indoor wireless IoT applications such as smart hospitals, factories, or buildings. This method used two types of feature selection techniques (constant removal and recursive feature elimination). The impact of the performance of these techniques was evaluated using three classifiers (SVM, Random Forest, and Decision Tree). The experimental results, using the public dataset, AWID, showed that the decision tree is the best classifier to be used to detect injection attacks (ARP, ChopChop and Fragmentation).

The proposed injection attack detection method achieved 99% accuracy, 95% precision, and 90% F1 score using the decision tree classifier with just 8 features. This was further achieved by only using 8 features selected using a pipeline of the constant removal and eecursive feature elimination. Compared with all the related work above, it could be remarked that our proposed method achieves nearly the same accuracy while achieving better results in terms of precision and F1-Score and it only needs 8 features to achieve these results, which makes it more appropriate in a smart IoT environment. Thus, it could be concluded that the proposed method is not only accurate but could also save resources which are limited in the IoT environment. The main purpose of this paper is to propose an IDS with high accuracy and a minimum number of features. In the future, it is planned to further investigate a deep comparison between the proposed feature selection technique and other well-known techniques in terms of convergence time, convergence iteration, and their impact on the detection accuracy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] L. Columbus, 2020 Roundup of cybersecurity forecasts and market estimates, 2020, URL: https://www.forbes.com/sites/louiscolumbus/2020/04/05/2020-roundup-of-cybersecurity-forecasts-and-market-estimates/?sh=1b00bf61381d.
[2] IDC, IoT Growth demands rethink of long-term storage strategies, 2020, URL: https://www.idc.com/getdoc.jsp?containerId=prAP46737220.
[3] Y. Chahid, M. Benabdellah, N. Kannouf, Smart hospitals and cyber security attacks, in: International Conference on Digital Technologies and Applications, Springer, 2021, pp. 291–300.
[4] H.H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, K.-K.R. Choo, A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks, IEEE Trans. Emerg. Top. Comput. 7 (2) (2019) 314–323.
[5] M.E. Aminanto, R. Choi, H.C. Tanuwidjaja, P.D. Yoo, K. Kim, Deep abstraction and weighted feature selection for Wi-Fi impersonation detection, IEEE Trans. Inf. Forensics Secur. 13 (3) (2017) 621–636.
[6] L. Tian, Design and implementation of a distributed intelligent network intrusion detection system, in: 2010 International Conference on Electrical and Control Engineering, IEEE, 2010, pp. 683–686.
[7] C. Kolias, G. Kambourakis, A. Stavrou, S. Gritzalis, Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset, IEEE Commun. Surv. Tutor. 18 (1) (2015) 184–208.
[8] S.K. Singh, Y.-S. Jeong, J.H. Park, A deep learning-based IoT-oriented infrastructure for secure smart city, Sustainable Cities Soc. 60 (2020) 102252.
[9] M. Kalinin, V. Krundyshev, P. Zegzhda, Cybersecurity risk assessment in smart city infrastructures, Machines 9 (4) (2021) 78.
[10] E. Bou-Harb, N. Neshenko, Cyber Threat Intelligence for the Internet of Things, Springer, 2020.
[11] T. Takeshi, C. Benjamin, R. Sara, et al., Light commands: laser-based audio injection attacks on voice-controllable systems, 2019.
[12] N. Abosata, S. Al-Rubaye, G. Inalhan, C. Emmanouilidis, Internet of things for system integrity: a comprehensive survey on security, attacks and countermeasures for industrial applications, Sensors 21 (11) (2021) 3654.
[13] S. Latif, Z. Idrees, Z. e Huma, J. Ahmad, Blockchain technology for the industrial internet of things: A comprehensive survey on security challenges, architectures, applications, and future research directions, Trans. Emerg. Telecommun. Technol. 32 (11) (2021) e4337.
[14] V. Varadharajan, U. Tupakula, K. Karmakar, Study of security attacks against IoT infrastructures, Technical Report TR1: ISIF ASIA Funded Project, 2018.
[15] M.R. Islam, K. Aktheruzzaman, An analysis of cybersecurity attacks against internet of things and security solutions, J. Comput. Commun. 8 (4) (2020) 11–25.
[16] T. Batiha, P. Krömer, Design and analysis of efficient neural intrusion detection for wireless sensor networks, Concurr. Comput.: Pract. Exper. 33 (23) (2021) e6152.
[17] M. Safaldin, M. Otair, L. Abualigah, Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks, J. Ambient Intell. Humaniz. Comput. 12 (2) (2021) 1559–1576.
[18] A. Fatani, A. Dahou, M.A. Al-Qaness, S. Lu, M. Abd Elaziz, Advanced feature extraction and selection approach using deep learning and aquila optimizer for IoT intrusion detection system, Sensors 22 (1) (2022) 140.
[19] R. Sekhar, K. Sasirekha, P. Raja, K. Thangavel, A novel GPU based intrusion detection system using deep autoencoder with fruitfly optimization, SN Applied Sciences 3 (6) (2021) 1–16.
[20] U.S.K.P.M. Thanthrige, J. Samarabandu, X. Wang, Machine learning techniques for intrusion detection on public dataset, in: 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2016, pp. 1–4.
[21] M.E. Aminanto, H. Tanuwidjaja, P.D. Yoo, K. Kim, Weighted feature selection techniques for detecting impersonation attack in Wi-Fi networks, in: Proc. Symp. Cryptogr. Inf. Secur.(SCIS), 2017, pp. 1–8.

[22] D. Kaleem, K. Ferens, A cognitive multi-agent model to detect malicious threats, in: Proceedings of the 2017 International Conference on Applied Cognitive Computing (ACC'17), 2017.
[23] V.L. Thing, IEEE 802.11 Network anomaly detection and attack classification: A deep learning approach, in: 2017 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2017, pp. 1–6.
[24] S.J. Lee, P.D. Yoo, A.T. Asyhari, Y. Jhi, L. Chermak, C.Y. Yeun, K. Taha, IMPACT: IMpersonation attack detection via edge computing using deep autoencoder and feature abstraction, IEEE Access 8 (2020) 65520–65529.
[25] J. Ran, Y. Ji, B. Tang, A semi-supervised learning approach to IEEE 802.11 network anomaly detection, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, 2019, pp. 1–5.
[26] K. Kim, M.E. Aminanto, H.C. Tanuwidjaja, Network Intrusion Detection Using Deep Learning: A Feature Learning Approach, Springer, 2018.
[27] S. Wang, B. Li, M. Yang, Z. Yan, Intrusion detection for WiFi network: A deep learning approach, in: International Wireless Internet Conference, Springer, 2018, pp. 95–104.
[28] F.D. Vaca, Q. Niyaz, An ensemble learning based wi-fi network intrusion detection system (wnids), in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–5.
[29] M.A. Rahman, A.T. Asyhari, L. Leong, G. Satrya, M.H. Tao, M. Zolkipli, Scalable machine learning-based intrusion detection system for IoT-enabled smart cities, Sustainable Cities Soc. 61 (2020) 102324.
[30] D.T. Larose, C.D. Larose, Discovering Knowledge in Data: An Introduction to Data Mining, Vol. 4, John Wiley & Sons, 2014.
[31] C. Sammut, G.I. Webb, Encyclopedia of Machine Learning, Springer Science & Business Media, 2011.
[32] N.V. Sharma, N.S. Yadav, An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers, Microprocess. Microsyst. (2021) 104293.
[33] S. Ustebay, Z. Turgut, M.A. Aydin, Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier, in: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), IEEE, 2018, pp. 71–76.
[34] B.F. Darst, K.C. Malecki, C.D. Engelman, Using recursive feature elimination in random forest to account for correlated variables in high dimensional data, BMC Genet. 19 (1) (2018) 1–6.
[35] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2015) 1153–1176.
[36] N. Chaabouni, Intrusion Detection and Prevention for Iot Systems Using Machine Learning (Ph.D. thesis), Université de Bordeaux, 2020.
[37] P.S. Kumar, S. Akthar, Execution improvement of intrusion detection system through dimensionality reduction for UNSW-NB15 information, in: Mobile Computing and Sustainable Informatics, Springer, 2022, pp. 385–396.
[38] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[39] V. Vapnik, The Nature of Statistical Learning Theory, Springer science & business media, 2013.
[40] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Trans. Neural Netw. 5 (4) (1994) 537–550.
[41] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[42] M.E. Aminanto, K. Kim, Detecting impersonation attack in WiFi networks using deep learning approach, in: International Workshop on Information Security Applications, Springer, 2016, pp. 136–147.

Tarek Gaber is a Lecturer at University Salford, UK. Dr. Gaber received a Ph.D. in Computer Science (Information Security) from the University of Manchester in 2012. He received a M.Sc. in Computer Science from Faculty of Computers and Information Sciences, Ain Shams University, Egypt in 2003. He has worked in many universities including Faculty of Computers and Informatics, Suez Canal University, Egypt, Faculty of Computers and Information Sciences, Ain Shams University, Egypt and the School of Computer Science, University of Manchester, Manchester, UK. He had a postdoctoral position at the Faculty of Electrical Engineering and Computer Science, VSB Technical University of Ostrava, Ostrava, Czech Republic. He is also a member of IEEE and the Scientific Research Group in Egypt (SRGE). As a guest editor, he has co-edited several special issues of SCI journals. He has served as a co-chair and PC member in many international conferences and reviewed many scientific papers. He has more than 70 publications in international prestigious journals, conferences, and book chapters. In addition, he has 6 edited books. He is the PI and Co-PI of several research projects funded by governments and industries. Tarek has successfully supervised 4 MSc-by-Research students and he is currently supervising other PhD/MSc students. His major research interests include cybersecurity, secure software engineering, behaviour authentication, machine learning, wireless sensor network, Internet of Things, and pattern recognition.