Complex & Intelligent Systems (2022) 8:3719–3746
https://doi.org/10.1007/s40747-021-00498-4

ORIGINAL ARTICLE

An edge based hybrid intrusion detection framework for mobile edge computing

Ashish Singh[1] · Kakali Chatterjee[2] · Suresh Chandra Satapathy[1]

Received: 24 February 2021 / Accepted: 3 August 2021 / Published online: 25 August 2021
© The Author(s) 2021

Abstract
The Mobile Edge Computing (MEC) model attracts more users to its services due to its characteristics and rapid delivery approach. This network architecture capability enables users to access the information from the edge of the network. But, the security of this edge network architecture is a big challenge. All the MEC services are available in a shared manner and accessed by users via the Internet. Attacks like the user to root, remote login, Denial of Service (DoS), snooping, port scanning, etc., can be possible in this computing environment due to Internet-based remote service. Intrusion detection is an approach to protect the network by detecting attacks. Existing detection models can detect only the known attacks and the efficiency for monitoring the real-time network traffic is low. The existing intrusion detection solutions cannot identify new unknown attacks. Hence, there is a need of an Edge-based Hybrid Intrusion Detection Framework (EHIDF) that not only detects known attacks but also capable of detecting unknown attacks in real time with low False Alarm Rate (FAR). This paper aims to propose an EHIDF which is mainly considered the Machine Learning (ML) approach for detecting intrusive traffics in the MEC environment. The proposed framework consists of three intrusion detection modules with three different classifiers. The Signature Detection Module (SDM) uses a C4.5 classifier, Anomaly Detection Module (ADM) uses Naive-based classifier, and Hybrid Detection Module (HDM) uses the Meta-AdaboostM1 algorithm. The developed EHIDF can solve the present detection problems by detecting new unknown attacks with low FAR. The implementation results illustrate that EHIDF accuracy is 90.25% and FAR is 1.1%. These results are compared with previous works and found improved performance. The accuracy is improved up to 10.78% and FAR is reduced up to 93%. A game-theoretical approach is also discussed to analyze the security strength of the proposed framework.

Keywords Mobile Edge Computing (MEC) · Edge network · Edge-based Hybrid Intrusion Detection Framework · Signature detection · Anomaly detection · Machine learning classifiers

Introduction

Mobile Edge Computing is a distributed computing environment in which information technology with networking-based elements such as hardware, memory, and virtual resources are integrated with telecommunications networking such as mobile base stations [1,2]. It extends the capabilities of cloud computing and IT services at the edge of the cellular network. This technology was primarily based on positioning edge nodes on the mobile network so that software and associated computing activities could be run closer to cellular subscribers/customers. Using this feature, a cellular operator can develop efficient services for a set of customers. The MEC technology benefits included closed storage, high network speed, better CPU utilization, and less costly services. Aside from these benefits, some of the other key benefits are: used to develop the new business model, reduced assets' load, improve the operational efficiency, socio-economic benefits, enables distributed computing capabilities, and improve network operations. These MEC benefits will minimize network congestion, increase network capacity/performance and fast response time. Application developers and authorized users can take advantage of MEC features, including real-time Radio Access Network (RAN) information to develop new MEC applications. The

✉ Ashish Singh
ashishashish307@gmail.com

[1] School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, Odisha 751024, India

[2] Department of Computer Science and Engineering, National Institute of Technology, Patna, Bihar 800005, India

MEC technology is specially designed for compute-intensive and latency-sensitive applications like augmented reality [3], real-time data analysis [4], Internet of Things (IoT) [5], video analytics, indoor positing, smart vehicles [6], smart IoT-based healthcare system [7–10], and image processing.

However, in mobile edge computing, multiple smart applications use the same edge device with a different set of users simultaneously. This increases the security issues of edge devices because if the edge device is hacked, it can provide false output data with inaccurate input data and produce incorrect results. This will affect the performance of any sensitive application. Also, the data of the hacked edge device can be misused for other unwanted or illegal processes. Generally, MEC faces two major types of attacks. The first one is many times, the edge device itself compromised after authentication. Any insider can perform an unwanted activity using the device as certain privileges are given to the devices for some inside network activity. This can be called insider attacks or unauthorized attacks [11] through which the attacker tries to attack the edge layer or cloud layer. The second one is sometimes unauthenticated edge device tries to attack the device layer or cloud layer with some advanced techniques. This type of attack is called an unauthenticated attack or outside attack. It is very difficult to identify insider attacks as a mobile edge computing environment is followed a multitenant architecture where resources are shared among different applications. Hence, malicious edge device attacks are the major drawback in the mobile edge computing environment. The firewall is the first line of defensive solutions to protect such networks. But, a traditional firewall solution can only block the packets coming from the outside through the internet. But, it cannot filter the malicious inside packets generated due to insider attack [12,13]. Due to the variability and complexity of the edge nodes, such firewall-based solutions are not effective. Also, most of the MEC devices are resource constrained and costly to manage a huge number of firewalls to implement security solutions.

Motivation

Mobile Cloud Computing (MCC) concepts are introducing the benefits of cloud computing for mobile users [1]. It provides higher data storage capabilities to the users and providing many sophisticated applications with services. But at the same time, it imposes a huge additional load on radio and mobile networks and introduces high latency. This is not helpful for real-time applications. Consider an application is used to process real-time video streaming or traffic monitoring. In such streaming-based transmission, the large number of access requests and faster growth in computing capabilities may reduce the efficiency of the network. This reduction of the efficiency of the mobile network is due to a lack of energy, bandwidth, latency, etc. Hence, cloud-based applications suffer from latency and network-related bottlenecks. Mobile edge devices remove these limitations of cloud computing with lower latencies and jitter. Therefore, there is an immediate requirement of the edge-based framework for such real-time applications. Another very common problem is different malicious attacks like Distributed DoS (DDoS), ransomware, remote recording, routing attacks, and data leakage attacks are the most probable security threats faced by the MEC system [14]. Among these security threats, a DDoS attack is very much focused on which a large number of compromised machines launches a DoS attack on the MEC environment for the disturbance of network service and reduction of service efficiency [14]. With this attack, the intruder overloads the MEC server with a false request packet to block the server to make the resources unavailable and increase the response time. Many security mechanisms such as access control algorithms, homomorphic encryption, public key-based security algorithm, and intrusion detection mechanisms are considered to protect this MEC network. But efficient security solutions are not available in MEC.

This motivates me to do the research in MEC to build an efficient Intrusion Detection System (IDS) that can identify all the attacks as mentioned earlier and secure the mobile edge devices so that efficient transmission of real-time data can be possible.

Need of hybrid IDF

Signature-based IDS (SIDS) is a popular method for intrusion detection as they can be effective by regular updating the signature database. It generally used signatures through the polymorphic behavior of malware. But it has some limitations. The first limitation includes many times it fails to succeed in similarity test because the incoming signature does not match with stored signatures in the IDS database. Thus, the malicious packets can enter in the system very easily. Second, if the signature database volume increases, the IDS automatically takes a longer time for analyzing and processing of huge incoming data packets. Third, it cannot detect zero-day attacks, a very new unknown attack, as the attack signatures are not available in the databases after regular updates. Thus SIDS fails to detect all known and unknown attacks.

However, Anomaly-based IDS (AIDS) can efficiently detect all attacks using different ML-based, statistical-based, and knowledge-based techniques. AIDS mainly creates normal profiles of attacks. But it has some limitations. The first limitation includes if the attack pattern is general in nature, then it could not identify several intrusions. This can give a poor detection rate. Second, in AIDS, if the profiles of the packets are very specific, then only it can identify them. Otherwise, several normal behaviors of network traffic could be classified as attack class. Due to this, it has a high false-

positive rate, giving high alarm generation by the AIDS decision module. Third, most of the AIDS models are trained at the beginning phase, and after that, they can be used for a long time. In many cases, due to dynamic network changes in traffics, there is a need to retrain the model based on new attack behavior.

Thus, both SIDS and AIDS have some strengths and weaknesses. A Hybrid Intrusion Detection Model (HIDM) is developed using an ensemble method of both techniques to overcome the limitations mentioned above. In literature, there are popular ensemble methods such as Bootstrap aggregating, boosting, and stacking. In this model, the boosting strategy is used because this process can steadily create an ensemble by utilizing the misclassified training instances of past models.

Problem formulation

The intrusion detection mechanism is the most common security solution capable of identifying malicious activity or attacks. The IDS helps to find intrusive behavior of network traffic that may harm the system. It is mainly composed of an agent, an analysis engine, and a response module. Several IDS-based security solutions were developed in the previous year [15–25]. But, these IDS-based security solutions cannot cope with the MEC architecture due to the changing behavior of users and devices. Most of the IDS solutions analyzed network traffic for the detection of attacks. But, nowadays, the amount of network traffic is huge and contains heterogeneous MEC devices in terms of protocols, manufacturers, and applications. Hence, anomalies or signatures-based network traffic analysis solutions are not suited to detecting malicious activity or intrusion in the MEC environment. One of the challenging tasks in IDS is building a behavior framework containing a rule set, pattern analysis to differentiate normal behaviors and abnormal behaviors from collected historical data. But, nowadays, attack patterns are changing each day and very difficult to detect new attacks. Hence, it is challenging for intrusion detection with a good performance and low time complexity. These significant discussions comprised two major problems. The first identified problem is found in the current approach because most of the ML-based IDS are not efficiently detecting unknown attacks or new attacks, which is also known as a zero-day attack. The second problem is that with the reduction of FAR, the computational complexities increased. As the devices are constrained, so the performance is also degraded.

Problem solution

An edge-based hybrid IDS framework is essential to solve these identified problems in MEC architecture. This proposed framework can detect unknown attacks or new attacks, and the computational complexities of the system are low. Usually, signature-based models cannot detect unknown attacks, whereas the anomaly detection models detect both known and unknown attacks with increasing FAR. These limitations can be overcome by using hybrid models. While designing a hybrid model, boosting approach is considered as an ensemble learning strategy that combines numerous models rather than utilizing a single model to improve ML performance. [26]. This approach will solve both identified issues. Hence, the main objective of this research work is to develop a solution using a mobile edge-based hybrid IDF that detects known attacks and can identify new unknown attacks. The developed edge-based hybrid IDF can able to prevent the MEC environment from malicious attacks with minimal overhead on MEC devices.

The key contribution of the work are summarized below:

- An edge-based hybrid IDF has been proposed for the MEC environment, which not only detects known attacks but also able to detect unknown or new attacks.
- The proposed edge-based hybrid IDF comprises a hybrid detection module that includes SIDS and AIDS modules.
- A game theoretical approach is followed to analyze the security strength of the edge-based IDF.
- The proposed framework is demonstrated and analyzed to measure the performance of the system. The achieved experimental results and security analysis of the model shows high accuracy and ability to detect unknown or new attacks.

The remaining parts of the paper are structured as Background is discussed in "Background". "Proposed edge-based hybrid IDF" presents the proposed hybrid edge-based IDF. The implementation results are explained in "Experimental results and performance analysis". A game-theoretical approach for the security analysis is discussed in "Security analysis". Conclusion is discussed in "Conclusion and future scope".

Background

Edge-based IDS security architecture

Most of the current enterprise security solution uses cloud architecture where service providers have responsible for satisfying all the security requirements. But, ad hoc environment and low-latency requirement of the applications like MEC, IoT, these security solution is not scalable. The edge computing concept provides a new way to design and deploy new security solutions for the MEC environment. The main objective of any edge-based security solution is to satisfy all the security requirements at the network's edge.

This paradigm deploys security solutions at the edge computing layer rather than at the cloud. This can also offload the processing task from the end devices by enabling computation and storage capacity at the edge network. Hence, the edge computing paradigm not only improves the system security level but also reduces network latency and network congestion. The edge-based security architecture is mainly classified into user-centric, device-centric, and end-to-end security [27]. An edge-based IDS can work in an uncertain condition where data size is big and needed a fast response.

A mobile edge computing architecture with IDS is shown in Fig. 1. The edge-based mobile computing architecture mainly concerning with three layers: end-user layer, mobile edge networking layer, and data storage layer. The data storage layer consists of resources, information, and services with security features. First, at the end-user layer, the smart end-user is connected with the system using edge devices. Then, all the edge application raw traffic goes to the mobile edge networking layer. In this layer, most of the essential security features are deployed. In this work, the Edge-based Hybrid IDF is deployed in this layer. The responsibilities of this layer are to reduce the network latency, satisfying many real-time needs, offloading heavy computational tasks, process the data very quickly, provide security features, monitor all the traffic data, and many more. After processing the data, the data storage layer stores the data into the cloud.

IDS overview and limitations

Generally, four types of IDS are used to provide the security to MEC environment. These are Host-Based (HIDS) [28], Network-Based (NIDS), Hypervisor-Based, and Distributed IDS (DIDS). Host-based IDS are responsible for monitoring and analyzing the information collected from a specific host machine. Network-based IDS are used to detect network intruders by comparing the current behavior of network traffic with the previously observed traffic behavior in real time. DIDS is composed of many HIDS /NIDS for network traffic monitoring over a huge network. It allows the user to monitor and examine communications between Virtual machines (VM). Hypervisor-based IDS are specifically used in Cloud computing for intrusion detection in a virtual environment. Mainly these IDS use different types of intrusion detection techniques. These techniques are based on: Signature [29–33], Anomaly [34–38], Artificial Neural Network (ANN) [39–43], Fuzzy Logic [44–47], Association Rule [34,48,49], Support Vector Machine (SVM) [50–52], Genetic Algorithm (GA) [53–57], Hybrid Technique [58]. Signature-based IDS mainly detect intrusion by matching captured patterns with previously generated pattern databases. Thus, they cannot detect unknown attacks, which produces high FAR for unknown attacks. Anomaly detects unknown attacks with low accuracy. Due to this drawback, ANN-based IDS was found in the literature. ANN-based IDS [43] classifies unstructured network packets efficiently with multiple hidden layers. But, it requires more time and more training of samples. Fuzzy logic-based IDS and Association Rule-based IDS provides a good result for some uncertain problems. But the detection rate is less. SVM-based IDS can correctly classify intrusions for a given sample data. This type of IDS can handle a large number of preprocessed data with high accuracy. In GA-based IDS, complexity increased with computational cost, whereas in Hybrid IDS is an efficient approach to accurately classifying rules.

The mobile IDS consists of handheld wireless devices (smartphones or mobile devices) with intrusion detection capabilities [59–65]. It has a self-configurable network in which, without the help of any party, the system automatically deployed very quickly. While the IDS technology is moved from a stand-alone computer to a mobile device, various design and implementation constraints arose. In such a mobile-based IDS, battery-powered, limited energy supply, constrained resources, limited processing capability, low memory, and low sensing range are some of the challenging issues. The ad hoc nature and independent running nodes of the mobile IDS make the system more vulnerable and might not be enough to detect intrusions. In some cases, mobile applications are running on a specific platform and allowing third-party applications. Thus, unique vulnerabilities and new intrusive traffic cannot be identified using smartphones. Due to resource constraints, the nature of mobile IDS response time of suspicious activity may be high, and the ability to visualize the traffic patterns is limited. Physical security of mobile nodes, absence of central security management point, single point of failure, undefined network boundary, and low cryptographic supports are some limitations of mobile IDS that will keep in mind while the intrusive packet identification is attempted using a mobile device.

Related works

A literature work has been carried out to examine recent findings in the area of IDS solutions in MEC networks [66–74]. A firewall architecture has been designed to protect the edge network from the insider attack [75]. This architecture support correct, non-bypassable, and tamper-resistant characteristic reside in any protection system. A deep learning approach for the detection of intrusion in Internet society has been discussed in [17]. Recurrent Neural Networks (RNN) lead the binary and multi-class classification, which will help measure the system's performance. In such a system, high computational processing was observed, which will reduce the system efficiency [76]. A Distributed Intrusion Detection Systems (DIDS) has been proposed in [77]. This work aims to reduce the false alarm rate in DIDSs-based edge computing environments. They also reduced the response

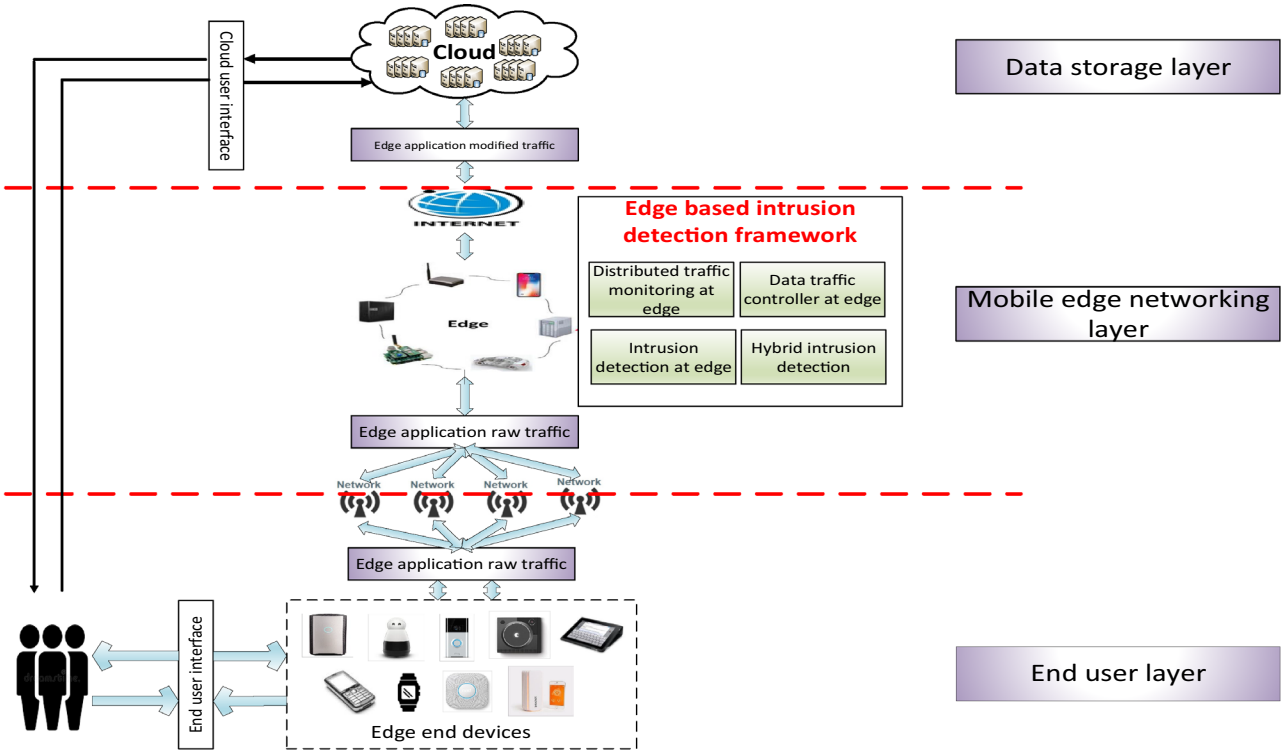

Fig. 1 Mobile edge computing architecture with IDS

time and energy consumption. A deep belief network for the Edge-of-Things (EoT) has been proposed in [16]. The proposed system can detect intrusive activities in the EoT network. The proposed framework consists of data collection, feature extraction, and classification module. But, the computational requirement and cost of this model are high. The security of the Internet of Things (IoT) network is an important issue. To view this security issue, in [18], proposed a robust IDS. This approach is comprised of a multi-agent system, blockchain, and deep learning algorithms. The system's efficiency is high, but combining three different approaches increases the system complexity and the response time. For the IoT infrastructure, device-edge-based IDS has been proposed in [19]. The IDS is made with the help of behavioral profiles and system-level information. The unique split architecture supports effective detection with minimal latency. But, the complexity of the system architecture was computationally overloaded. An IDS has been developed in [15] for the internet industry. They have also designed the concept of Cloudlet on which Edge-based IoT devices are deployed in urban areas. The proposed model consists of a Microcontroller module, a Mobile application module, and a Database module. But, the security efficiency and performance of the model are low. A network IDS has been proposed in [20] for mobile edge computing. This technique captures all the tcpdump packets, extracts and analyses the features, and, if identified as a legitimate packet, forward it into the network. A topic model is trained to learn the behavioral pattern of a normal packet. But, the detection accuracy is affected in case of new types of the packet is come into the networks. Data-driven mimicry and game theory-based IDS have been proposed in [78]. The new attacks are investigated based on the game income of participants and game balance points in the edge computing networks. They also try to reduce the cost of the IDS. Traffic inspection and classification-based distributed attack model has been proposed in [79] for the IoT applications. They have also leveraged the flexibility of cloud-based architectures with edge computing architecture. This model shifted more computational work at the cloud. But, a traffic classification-based mechanism does not give accurate results if new traffic comes into the network. An IDS for smart connected vehicles has been proposed in [80]. They try to achieve user requirements such as quality of service (QoS) and quality of experience (QoE). The detection module contains data traffic analysis, reduction, and classification techniques. Deep belief and decision tree ML mechanisms are used for data reduction and classification purposes, respectively. Multilayer Perceptron (MLP)-based lightweight IDS has been proposed in [81]. Vector space representation and a single hidden layer MLP concept provide a reasonable detection rate which also improves the security of the fog computing environment. But, the com-

putational efficiency of the proposed model is much high. This can lead to an increased detection time. In [82], proposed an IDS for multi-access edge computing environment. The evaluation criteria consider both the centralized and distributed collaborative environment on which the IDS is deployed. The Distributed Hash Tables (DHTs) in Peer-to-peer (P2P) communication may reduce the data transmission overhead. The cyberattacks in fog-assisted networks are possible if a proper security solution is not implemented [83]. The artificially full-automated IDS is a solution developed in [83] may protect the system against cyber-attacks. An IDS based on multi-layered deep recurrent neural networks has been designed to secure end-users and IoT devices. An IDS for Vehicular Edge Computing has been proposed in [84]. The proposed cooperative IDS offloads the training task from the distributed edge devices. The federated learning approach reduces the resource utilization from the edge server. Blockchain is used to provide security and privacy while the training information is shared between the edge devices. A novel two-phase cycle algorithm has been proposed in [85] for the detection of cyber intrusion in an edge computing environment. KDD cup 1999 benchmark dataset is used to validate the performance of the model. The proposed model achieved 98.81% accuracy and 98.23% detection rate. The proposed model is validated on the ancient datasets, which needs to be improved by measuring the performance on the recent dataset. A deep learning approach has been used in [86] for the detection of host intrusion in an edge-IoT environment. They have achieved 99.74% accuracy and 1 μs attack prediction timing. In Lambda architecture, an IDS has been proposed to provide the security of edge-cloud-based IoT system [87]. The proposed model uses a deep learning approach for the detection analysis. The edge-cloud concept reduces the training time compared to traditional ML algorithms and improves the attack detection accuracy.

A hybrid algorithm for the detection of intrusive packets has been proposed in [88]. The proposed approach uses Artificial Bee Colony (ABC) and Artificial Fish Swarm (AFS) to develop Internet traffic classification. The selected features use the CART technique to generate If–Then rules that will classify the intrusive packets. A misuse-based hybrid IDS has been proposed in [89]. This hybrid approach combines two different techniques in one unit. Packet traffic and network traffic anomaly detection with snort make an anomaly-based hybrid approach. This approach analyzed the system activities and determine the matching score by finding out the relation between system activities and known activities having definitions into the system. The hybrid system can detect up to 146 attacks. Multi-classifier-based approach has been used in [90] for the development of the hybrid IDS algorithm. It combines AIDS and SIDS for the detection of intrusive packets. SIDS used a C5 decision tree classifier, and after that, a one-class Support Vector Machine algorithm was used to split the dataset, which gives a better result than the previous model. They have achieved 83.24% accuracy. Al-Yaseen et al. [91] developed a hybrid model with the multilevel model where SVM and extreme learning machine is used. This model has good performance results for unknown attacks with low FAR for the dataset KDD Cup 1999. An anomaly and signature-based hybrid intrusion detection approach have been proposed in [92] for the detection of DDoS attacks. The integrated output of both methods make the system hybrid and enhanced the overall accuracy. The proposed model is implemented on two datasets (DARPA 2000 Dataset and commercial Bank from a Penetration Test). A hybrid IDS has been proposed in [93] for the security of the smart home system. They have used multiple ML techniques, including random forest, decision tree, X gboost, and misuse detection to implement the work. A hybrid IDS approach has been followed in [94] for the security of Internet-connected smart vehicles. A multi-tiered hybrid IDS incorporates a SIDS and AIDS to detect both known and unknown attacks on vehicular networks. The experiment result illustrated that the model achieves up to 99.99% accuracy.

From the above discussion, it has been found that the hybrid IDS can be used to improve the detection performance in terms of FAR and unknown attack detection rate. But, due to multiple repetitive operations, the complexity and overhead will be increased. Some of the literature works limitations are tabulated in Table 1.

The following research problems have been identified in existing IDS solutions in MEC from these overall works of literature.

- Despite all advantages of ML-based IDS in MEC, there is a lack of a hybrid model which can solve the identified research objective.
- Most of the ML-based IDS in MEC gives high accuracy with high FAR.
- The difficulty in the existing MEC system is achieving low FAR with minimal overhead since the devices in the MEC environment is constrained.
- The hybrid IDS can reduce the FAR but automatically increases complexity and running time.

Attacks on mobile edge computing

- *Denial of Service (DoS) attack:* An attacker fires many packets to a target victim from an innocent computer or host in the network. Due to these false packets, most of the bandwidth is exhausts and engages all the resources. This can lead to flooding attacks and zero-day attacks in which a genuine user request cannot proceed or he cannot access the services.

Table 1 Comparative analysis of IDS solutions in different areas

Paper	Year	Working environment	Aim of the work	Proposed approach	Limitation
Markham et al. [75]	2001	Mobile computing, business to business computing and virtual private networks (VPNs)	Protect the system from insider attack	Network edge security based on distributed firewall architecture	Low accuracy in case of unknown threats
Xia et al. [56]	2005	Internet society	An IDS that monitor the network behavior and identify different threats	Information theory and genetic algorithm	In case of unknown attacks detection accuracy will down
Houmansadr et al. [51]	2011	Mobile phones	An IDS and response engine that will monitor the mobile phones intrusive traffic	Uses the cloud computing concepts	Accuracy and detection level is missing
Subramanian and Ong et al. [36]	2014	Internet society	Designing network IDS	Naive Bayes classifier is used for attack detection	Single Classifier does not provide complete security
Yin et al. [17]	2017	Internet society	An IDS for detection of intrusive packets	Deep learning-based recurrent neural networks	High latency was observed
Kozik et al. [79]	2018	Edge computing environments	Design attack detection and security alert facilities at the edge networks	Cloud-based approach, extreme learning machine classifier and HPC cluster resources are used for development of the system	Low accuracy in case of in case of new network traffic
Meng et al. [77]	2018	Distributed edge computing	Design intelligent alarm filter which will reduce false alarms in IDS	Taken the concept of edge computing for development of intelligent alarm filter	Detection performance is low in highly malicious environment
Aloqaily et al. [80]	2019	Connected vehicles in smart cities	Designing an IDS that protect the system from attacks and meet user service requirements	Data traffic analysis, reduction, and classification technique-based three-phase hybrid IDS, deep belief and ML-based decision tree are used for classification and reduction purpose	Unable to identify unknown attacks
Yao et al. [68]	2019	Edge-Based Industrial IoT	Discussing a hybrid IDS	Machine and deep learning approach with edge computing	Cover only theoretical analysis

Table 1 continued

Paper	Year	Working environment	Aim of the work	Proposed approach	Limitation
Sudqi Khater et al. [81]	2019	Fog computing	Designing lightweight IDS	Multilayer Perceptron (MLP) is used for IDS vector representation	High detection time
Almogren [16]	2020	Internet-of-Things (IoT)-based edge computing environment	Detection of intrusive activities	Developed an advanced IDS based on Deep belief network	Computational requirement and cost of the model is high
Liang et al. [18]	2020	IoT networks	Designing an intelligent IDS for protection of IoT networks from intruders and attacks	Hybrid detection approach based on a multi-agent system, blockchain and deep learning algorithms	Response time is high due to multiple approach
Mudgerikar et al. [19]	2020	IoT devices	Designing a system-level IDS for threat detection	Developed device-edge split IDS that uses device behavior for attack detection	Overloaded system complexity
Vimal et al. [15]	2020	Smart cities development	Designing an edge computing-based IDS	Uses RFID, edge computing, IoT, and cloud computing concepts	Low security efficiency and performance
Cao et al. [20]	2020	Mobile edge computing	Network IDS	Network traffic monitoring using latent Dirichlet allocation based on Bayesian topic model	Detection accuracy is reduced in case of new types of the packet is come into the networks

- *User to root attack:* The attacker wants to access the valid user account by password sniffing. Due to this attack, an attacker can access the system at the root level and modify any useful information.
- *Insider attack:* In this attack, an authorized user (called insider) wants to use the unauthorized privileges and disclose the data to others.
- *Port scanning:* In this attack, the attacker first scans the open port details of the victim's machine. He makes a list of filtered ports, open ports, and closed ports with the help of a port scanner. After that, the attacker performs the attacks on the services running on the listed port and affects the legitimate users' services.
- *Backdoor channel attack:* Hackers can gain access remotely to a compromised machine through the back door. This attack is also known as a passive attack. Hackers can use the backdoor to install malware to steal information from a network or a system.
- *Routing Information Protocol (RIP) attack:* The malicious user gains the routing information that comes from a trusted peer router. A malicious user spoof or corrupts the routing information. Changing the routing table by the malicious user will produce abnormal network traffic in a specific direction which will cause a DoS.
- *Tunneling attack:* This attack is to overcome the firewall filters. The firewall filters network traffic based on information on the network protocol. A tunneling attack is seen at the application level to search application vulnerabilities by sending packets directly to these applications.
- *Side channel attack:* The attacker first gains the public available, non-private sensitive side-channel information. This information is now used to establish a correlation between public and private information. After that, the actual information is fetched using the inferencing technique.
- *Malware injection attack:* The attacker inserts or installed malicious programs into the target system. Such an attack can be performed on either the edge server or storage server. SQL injection, XML signature wrapping,

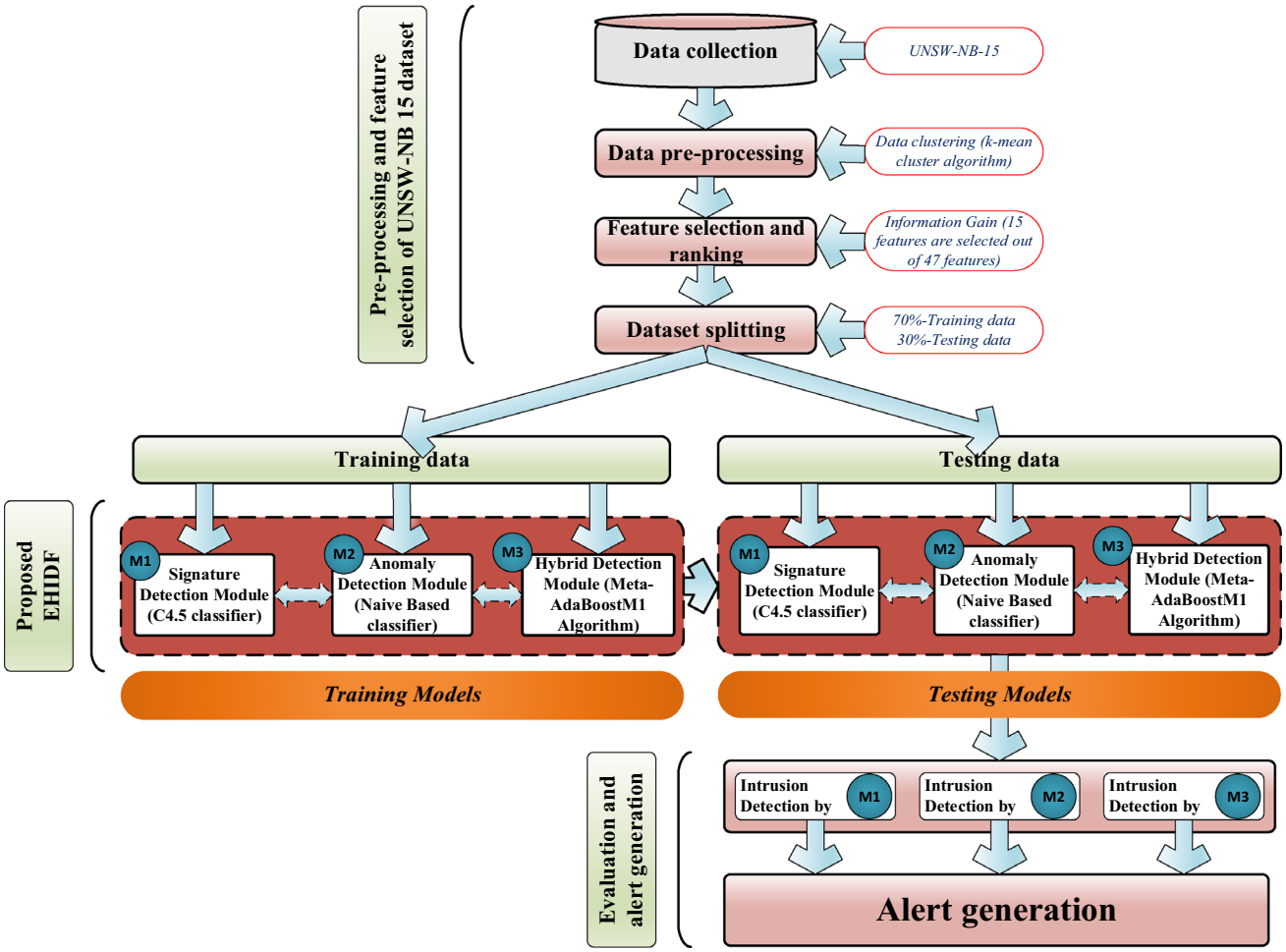

Fig. 2 Proposed edge-based hybrid IDF

cross-site request forgery, and cross-site scripting are categorized as server-side injection attacks.

Proposed edge-based hybrid IDF

This section discusses the working of the proposed EHIDF that includes SDM, ADM, and HDM. It uses UNSW-NB 15 dataset for the experiment and testing purposes. The SDM uses a C4.5 classifier, ADM uses Naive-based classifier, and HDM uses the Meta AdaBoostM1 algorithm for the classification and attack detection purpose. The proposed EHIDF is shown in Fig. 2. The subsequent subsections elaborate the steps as well as the working principle of the proposed EHIDF.

Dataset description

UNSW-NB 15 dataset is quite distinct than KDD CUP 99 and NSL KDD. KDD CUP99 and NSL KDD are old datasets with several disadvantages. NSL KDD is the extended version of KDD CUP99. It improved the weakness of KDD CUP99 by eliminating sample inconsistency and contain more attacks than the KDD CUP99. But the classes remain the same. In KDDCUP and NSL KDD, only four forms of DoS, PRB, U2R, R2L attacks were found, while in UNSW-NB15, nine different types of attacks are discovered.

The UNSW-NB 15 data set was produced with the help of the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [95–97]. The dataset includes nine groups of modern attacks. These attacks are Fuzzers, Analysis, Exploits, Backdoor, DoS, Reconnaissance, Generic, Shellcode, and Worms. The detailed description of 9 attacks are tabulated in Table 2. The dataset included 49 features grouped into five types—Flow features, Content features, Basic features, Time features, and Additional generated features. The flow features defines the identifier attributes (IP address and port number) of the end system (source and destination machine). The protocol connections attributes are represented by grouping them into basic features. The content features encapsulate the attributes

Table 2 Major known attacks in UNSW-NB15 Dataset [96]

Sl. no.	Attack name	Description
1.	Analysis	In this attack, the attacker sends several different instructions into the web application with the help of port scanning, protocol description, and HTML file analyzer
2.	Backdoor	It refers to an attacks technique in which a user can bypass typical system security measures and gain high-level access to a computer or data server
3.	DoS	An attack in which the attacker consumes the system resources, which leads to the unavailability of services and resources
4.	Exploits	A piece of software and leverages code that allows an attacker to access the network and computing resources with the help of security vulnerability/flaws present in the system
5.	Fuzzers	This attacking technique attacker tries to discover the loopholes in the system by feeding semi-random-generated data into the system so that the system will be crashed
6.	Generic	A type of attack against a cryptographical primitive without knowing about the configuration of the block cipher
7.	Reconnaissance	Contains all set of processes and techniques that will be helpful to determine the information of the target system
8.	Shellcode	A small chunk of payload script used to manipulate the flaw of the program
9.	Worms	Worms are malware that replicates themselves from one network to another without any human interaction

of TCP/IP, including some of the attributes related to HTTP service. The time features include the time attributes of the events that occur in the networks. The additional features include a general-purpose feature and connection feature. The general purpose feature is used for its own purpose, whereas connection features include the flow of 100 sequential order of record connections. The description of each feature is shown in Fig. 3. The total number of instance distribution for training and the testing dataset is presented in Table 3. The set of instances are divided into 70%:30% for training and testing purposes, respectively.

Data pre-processing

In the pre-processing data phase, the original UNSW-NB15 dataset size has been reduced by eliminating redundant data from the dataset. The clustering approach has been used to identify similar data behavior of different categories present in the dataset. It is started by removing the labels from the dataset. Silhouette coefficient is applied to determine the required number of clusters and cluster quality. k-mean cluster algorithm is used in which the value of k will be changed to determine cluster configurations. The generated cluster configuration measures the silhouette value. The maximum silhouette coefficient value for the least value of k is selected for further testing purposes. Figure 4 illustrated that silhouette value is maximum (0.7) when the number of clusters is 11. Dataset instances are distributed in 11 clusters which are shown in Table 4. Random data have been selected from each cluster for training and testing purposes. The training data contains approximately 70% of instances, and the remaining 30% samples are used for testing. The detailed description of the reduced dataset is given in Table 5.

Feature selection and ranking

The UNSW-NB15 dataset consists of total 47 features. In this phase, the attributes (features) are selected from the data set using the Information Gain value. This value shows the feature importance in a given dataset. The high information gain value of a feature contains high significance compared to other features. The Information Gain ($I_G\ (S,\ F)$) value of a feature (F) in a given dataset (size=S) is calculated using Eq. 1.

$$I_G\ (S, F)\ =\ E_N\ (S)\ -\ (E_N)_F\ (S), \tag{1}$$

where Entropy (E_N (S)) is the defined degree of non-homogeneity in the given dataset (size=S). It is computed using Eq. 2, where p_k is the proportion of instances for class k. $(E_N)_F\ (S)$ is the extra needed information for classification when feature F is selected. It can be defined by using Eq. 2, where $V(F)$ is the all distinct values of feature F, and S_v is the number of tuples for which feature F has value v.

$$E_N(S) = \sum_k\ -\ p_k\ log_2\ p_k, \tag{2}$$

$$(E_N)_F\ (S) = \sum_{v=V(F)} \frac{|S_v|}{|S|}\ *\ E_N(S_v). \tag{3}$$

The information gain value of all 47 features of the UNSW-NB15 dataset is presented in Table 6. Table 6 illustrated that $I_G\ (S,\ F)$ value of 25 features is 0. Thus, these features are not important because they do not provide any information or

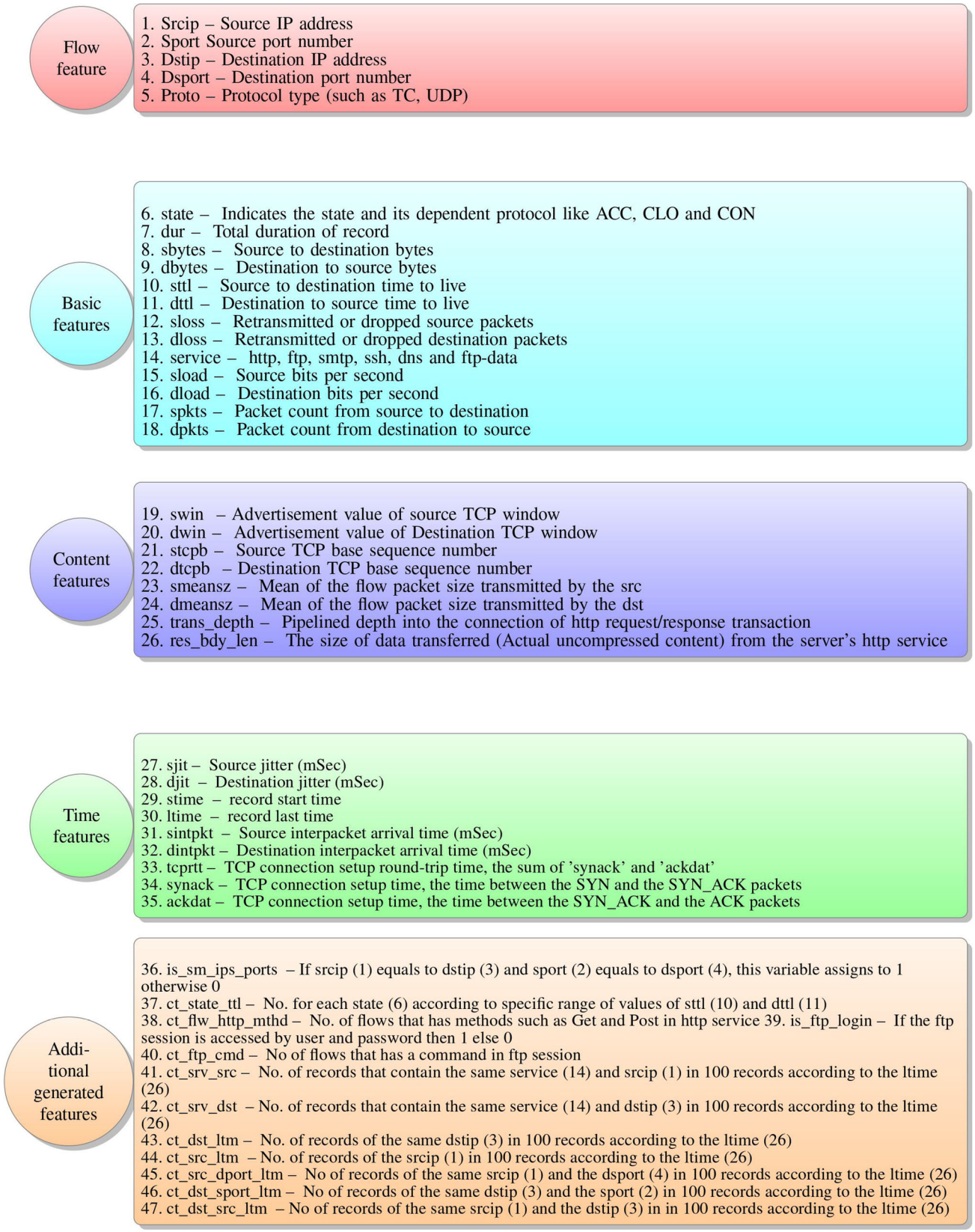

Fig. 3 Features description of UNSW-NB15 dataset [96]

Table 3 Instance distribution for training and testing dataset [96]

Attack category	Training instances	Testing instances
Normal	56,000	37,000
Analysis	2000	677
Backdoor	1746	583
DoS	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1133	378
Worms	130	44
Total instances	1,75,341	82,332

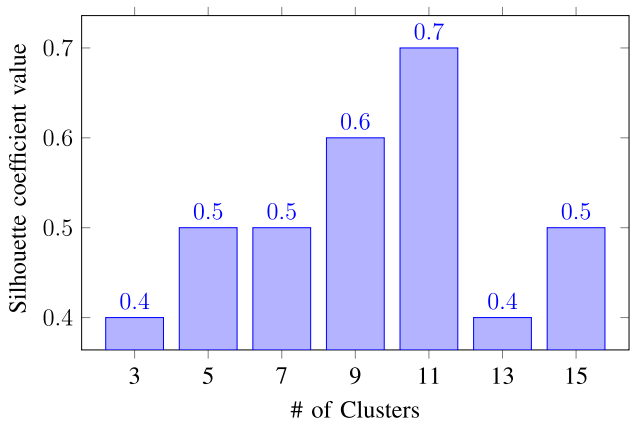

Fig. 4 Silhouette coefficient value for different number of clusters

contribution to the IDS. The remaining 22 features $I_G\ (S,\ F)$ value are also tabulated. It is also observed that less number of features shows significantly less accuracy compared to more number of features. The features are removed whose $I_G\ (S,\ F)$ value is near to 0 and consider the features whose $I_G\ (S,\ F)$ value are significantly high. Based on this phenomenon, the bold black (blue shaded) feature (15 features) is considered for evaluation.

Proposed EHIDF

The proposed EHIDF consist of the following modules.

Signature detection module (SDM)

The SDM is very important for the detection of known attacks. This module is trained by using a decision tree model with 15 features. The UNSW-NB15 dataset is broken into multiple subsets and increasingly recombining the subsets to make a decision tree in this model. Several decision tree models have been given in the literature, which generates a decision based on the records of the dataset. In this proposed model, the C4.5 algorithm is used for the signature-based detection process. . The rule is extracted and stored in the rule-based database at the time of the training phase. If the captured signature is matched with a normal set at the time of the testing phase, then the data is inserted into the table with the class attribute as normal. If any intrusive signature is detected, an alert is generated by the alert generator. The C4.5 algorithm is described in Algorithm 1.

Algorithm 1 Algorithm-C4.5 classifier for SDM

```
1: procedure INPUT:(Data sample (D))
2:     OUTPUT: Decision Tree (T)
3:     FUNCTION: SPLIT_TREE (T,a)
4:     BEGIN
5:     Max_Info_Gain = NULL
6:     Split_Tree = NULL
7:     Compute entropy
8:     for each attribute a in Data Sample D do
9:         if Info Gain > Max Info Gain then
10:            Max_Info_Gain = Info_Gain
11:            Split_Tree = a
12:    return Decision Tree (T)
13:    END
```

Table 4 Data distribution in 11 different clusters

Category	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Normal	4500	–	–	30,567	–	4581	–	–	6234	–	3892
Analysis	–	27	–	–	552	–	–	8	–	1245	–
Backdoor	167	–	67	54	1356	–	–	6	–	-	–
DoS	8745	234	393	56	634	245	874	50	18	78	–
Exploits	11,678	4567	568	346	–	–	867	1623	–	–	12,398
Fuzzers	6734	–	267	1756	–	–	–	–	–	–	5934
Generic	567	–	–	–	2348	12,678	–	–	3812	–	17,843
Reconnaissance	–	–	–	–	3723	–	–	–	4512	–	–
Shellcode	–	445	–	–	–	–	–	–	–	–	556
Worms	–	–	–	13	–	–	95	–	–	–	–

Table 5 Details of reduced data distribution

Category	Training instances	Testing instances	Training+testing
Normal	34,819	14,912	49,731
Analysis	1245	534	1779
Backdoor	1100	485	1585
DoS	7891	3367	11,258
Exploits	22,412	9600	32,012
Fuzzers	10,256	4367	14,623
Generic	26,023	11,145	37,168
Reconnaissance	5694	2456	8150
Shellcode	683	278	961
Worms	67	25	92
Total instances	11,0190	47,169	15,7359

Table 6 $I_G\ (S,\ F)$ value of 47 features

Feature	$I_G\ (S,\ F)$
Srcip, Sport, Dstip, Dstport, State, Sload, dload, Swin, Dwin, Stcpb, Dtcpb, Trans_depth, Res_bdy_len, Djit, Stime, Ltime, Sinpkt, Tcprtt, Synack, Ackdat, Is_sm_ips_parts, Ct_ftw_http_ mthd, Is_ftp_login, Ct_ftp_cmd, Ct_src_ltm	0
Dur	0.0002
Dmeanz	0.0003523
Spkts	0.00076
Sjit	0.00076
Sloss	0.00078
Dttl	0.00095
Smeanz	0.0018
Dpkts	**0.0023**
Proto	**0.0043**
Ct_src_dport_ltm	**0.0059**
Ct_srv_src	**0.0078**
Ct_dst_ltm	**0.0087**
Ct_srv_dst	**0.023**
Sbytes	**0.0234**
Dbytes	**0.02545**
Dloss	**0.033**
Ct_state_ttl	**0.045**
Ct_dst_src_ltm	**0.0512**
Ct_dst_sport_ltm	**0.1178**
Dinpkt	**0.1236**
Service	**0.1456**
Sttl	**0.3545**

Out of 47 features, we have used 15 features that is represented in bold

Anomaly detection module (ADM)

After the signature-based detection process, the detected intrusive signatures are stored in the database, and an alert will be generated based on the detection. But, the signatures recognized as a normal signature may contain some unknown patterns that are not available in the signature database. For this reason, unknown attack signatures may be labeled as normal in the database and the alert alarm will not generate. This is the main reason to apply an Anomaly-based detection module. The output of the signature-based detection module has two types first one is abnormal signatures which will generate alarms and the second one is normal signatures which will again pass through the ADM process for detection of unknown attacks. These input data in the ADM will have two phases, training and testing. In the training phase sample will be trained with the Naive-Based classification algorithm described in Algorithm 2. It is one of the probabilistic models to solve the classification problem. The soul of the Naive Bayes classification is the Bayes theorem. It is considered that all features in the classification problems are independent means that one feature probability does not affect the other features. For this reason, it is called the Naive Bayes Classification.

Algorithm 2 Naive Bayes classifier for ADM

```
1: procedure INPUT:(Data sample (D))
2:     OUTPUT: Class for testing Data sample
3:     FUNCTION: Scan (Data sample (D))
4:     BEGIN
5:     Calculate the standard deviation and mean of each class.
6:     for each attribute an in Data Sample D do
7:         Calculate the probability of P(i) using Bayes theorem in each
   class.
8:         Calculate the likelihood for each class.
9:         Get the maximum likelihood for the classifier.
10:    return A class for testing Data sample
11:    END
```

Hybrid detection module (HDM)

It is found that the previous modules (signature and anomaly) have some weaknesses in the classification process. The sig-

nature module cannot detect the unknown classes, whereas the anomaly module has an increased false alarm rate after predicting unknown samples. Hence, to balance between the detection accuracy and false alarm rate, the Hybrid approach has been adopted. Different classes are trained on the same dataset in this approach, and then the result will be combined. The Meta-AdaBoostM1 algorithm is considered to increase detection accuracy. AdaBoost is a boosting technique to make new classifiers that identify and focus on previously misclassified instances by previous classifiers. In this technique, a weak classifier is repeatedly trained on the training data. A decision stump is used for the weak classifier. The decision stump is a one-level decision tree. It has one internal node (root) connected to terminal nodes. The weak classifier is again trained with the same training dataset, and the weights are adjusted for precise classification. The weak classifier is again classified using a strong classifier. The meta-AdaboostM1 algorithm is used as a strong classifier. Meta-AdaBoostM1 Algorithm described in Algorithm 3. After that, the final decision is combined. An alert has been generated if the packet is detected as intrusive.

Algorithm 3 Meta-AdaBoostM1 Algorithm for HDM

procedure INPUT:(Data sample (D))
  OUTPUT: Final Hypothesis : H (x) = $\sin(\sum_{i=1}^{T} \alpha_i h_i(x))$
  FUNCTION: Weight initialization D (i)=$\frac{1}{m}$ for i=1 to m.
  BEGIN
  for each class i = 1 to m over distribution D(i) **do**
    Train the weak classifier D(i).
    Compute weak Hypothesis h(i).
    Compute low error rate.
  Set $\epsilon = \frac{No\ of\ incorrectly\ classified\ intances}{Total\ no.\ of\ intances}$
  Choose $\alpha = \frac{1}{2}[ln(\frac{1-\epsilon}{\epsilon})]$
  Now iterates from 1 to m and Update the weights of the classifier.

$$D_i^{t+1} = \begin{cases} \frac{D_i^t\ e^{-\alpha}}{Sum\ D}, & \text{For correctly classified intances} \\ \frac{D_i^t\ e^{\alpha}}{Sum\ D}, & \text{For incorrectly classified intances} \end{cases}$$

  return final hypothesis
  END

Complexity of the algorithms of different modules

The complexity of the discussed algorithms is calculated to determine the goodness of the proposed framework. The proposed framework consists of three different algorithms. The detailed analysis of the complexity of each algorithm is as follows:

Algorithm 1 contains the C4.5 classifier which is used in SDM. The first operation is to compute the entropy of the attributes which takes $\mathcal{O}(1)$ times. After that, for loop will start and the for loop contains three statements (one comparison and two assignment). The for loop will execute $\mathcal{O}(D)$ times and each statement will run over $\mathcal{O}(1)$ times. Thus, the total complexity of the algorithm is: $\mathcal{O}(D)+\mathcal{O}(1)+\mathcal{O}(1)+\mathcal{O}(1)+\mathcal{O}(1) = \mathcal{O}(D)$. Algorithm 2 runs in the ADM which uses Naive Bayes classifier. In this algorithm, the first operation (computation of standard deviation and means) will be executed on $\mathcal{O}(1)$. After that for loop will be executed D times so the complexity is $\mathcal{O}(D)$. The for loop contains three statements. The first statement completed over $\mathcal{O}(D_i)$. The second and third statement will takes $\mathcal{O}(1)$. Thus, the total complexity of the algorithm is $\mathcal{O}(1)+\mathcal{O}(D)+\mathcal{O}(D_i)+\mathcal{O}(1)+\mathcal{O}(1) = \mathcal{O}(D_i)$. The third (Algorithm 3) will be run in the HDM which is called Meta-AdaBoostM1 algorithm. The algorithm third line weight initialization function takes $\mathcal{O}(m)$ times. In the for loop, line numbers 6, 7 and 8 take $\mathcal{O}(m)$ times. Line numbers 9 and 10 take $\mathcal{O}(1)$ and line number 12 takes $\mathcal{O}(m)$. so the total complexity of the algorithm is $\mathcal{O}(m)+\mathcal{O}(m)+\mathcal{O}(m)+\mathcal{O}(m)+\mathcal{O}(1)+\mathcal{O}(1)+\mathcal{O}(m) = \mathcal{O}(m)$.

Experimental results and performance analysis

This section elaborates the implementation scenario, achieved result, and comparative performance of the proposed EHIDF. It mainly works on the network traffic, which goes to the edge computing-based mobile network. The model has been trained with the standard data set UNSW-NB15, which was operated as a real network. It is a refined data set, so first, it is used for training and testing purposes based on the reduced feature set. Performance analysis has been done with this dataset for the accuracy of the model.

Implementation setup

A mobile edge computing-based executable environment has been developed for the implementation of the work. The proposed EHIDF is evaluated on Raspberry Pi 3 Model B+, which has 1.4GHz 64-bit quad-core processor, 1GB LPDDR2 SDRAM, 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, and many more advanced features. It runs as an edge computing device and meets many ML models. This device runs on several flavors of Linux with the Raspbian operating system. The ML source code is written in Python high-level Multi-paradigm programming language by installing Python libraries and TensorFlow (an open-source software library). It offers two choices Mu editor and SSH for writing the Python code. An edge tensor processing unit has been developed for mobile and embedded devices which accelerate the TensorFlow computation. The proposed EHIDF is implemented on Intel(R) Core(TM) i5-8250U CPU 1.60GHz 1.80 GHz

Table 7 Measurement factors

Factors	Description
True positive rate (T_P)	The total amount of incidents identified as normal while they were truly normal
True negative rate (T_N)	The total amount of incidents identified as attack while they were truly attack
False positive rate (F_P)	The total amount of incidents identified as normal while they were truly attack
False negative rate (F_N)	The total amount of incidents identified as attack while they were truly normal

Table 8 Performance metrics

Measurement metric	Description	Formula
Accuracy (A)	Accuracy metric defines the percentage of correct classification of the test data. It is measured by dividing the correct classification of all classes by the total number of records in the dataset.	Accuracy (A) = $\frac{T_P+T_N}{T_P+T_N+F_P+F_N}$
Precision (P)	Precision is calculated by dividing the value of true positive by the total of true positive and false positive.	Precision(P) = $\frac{T_P}{T_P+F_P}$
Recall (R)	The Recall is the percentage of True Positive divided by Total of True Positive and False Negative.	Recall (R) = $\frac{T_P}{T_P+F_N}$
F-Score	The harmonic mean of Precision and Recall is the F-score.	F-Score = 2 * $\frac{R*P}{R+P}$
AvgAccuracy	AvgAccuracy is the mean recall value across all classes of the given dataset.	AvgAcc = $\frac{1}{C}$ * $\sum_{i=1}^{C} R_i$
AttackAccuracy	Attack Accuracy is a metric used to calculate a model's ability to detect attack classes only by not taking normal traffic into account.	AttAcc = $\frac{1}{C-1}$ * $\sum_{i=2}^{C} R_i$
Attack Detection Rate (ADR)	Attack Detection Rate is the rate of accuracy of the model for attack classes.	ADR = $\frac{\sum_{i=2}^{C} T_{P_i}}{\sum_{i=2}^{C} T_{P_i}+FP_i}$
False Alarm Rate (FAR)	False Alarm Rate shows the non-attack classes categorized as an attack. This metric makes normal traffic possible and quantifies the F_N.	FAR = $\frac{F_{N_i}}{T_{P_i}+F_{N_i}}$

processor with 8.00 GB (7.89 GB usable) RAM and 64-bit Ubuntu Linux operating system.

Evaluation metrics

The proposed edge-based IDS is evaluated based on some measurement criteria. The following evaluation measures are used to assess the model's performance. Accuracy (A), Precision (P), Recall (R), F-Score (F), AvgAccuracy (AvgAcc), AttackAccuracy (AttAcc), Attack Detection Rate (ADR), FAR. For the computation of 4 different measurement factors (T_P, T_N, F_P, and F_N) are needed. The detailed description of all the factors is shown in Table 7. Table 8 shows the detailed description of the evaluation metric with computation equations.

Result analysis

In this subsection, the result of the proposed framework is discussed. Each module of the proposed model uses the UNSW-NB15 dataset for testing purpose. The achieved results are demonstrated in terms of a C×C confusion matrix, where M is the total number of classes/categories. The confusion matrix includes the following classes/categories: Normal, Backdoor, Analysis, DoS, Fuzzers, Exploits Generic, Shellcode, Reconnaissance, and Worms. The tabulated confusion

Table 9 Confusion matrix of SDM (C4.5 classifier) on data UNSW-NB15

Class names	Normal	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms	Classification overall	Precision (%)
Normal	11978	2034	20	5	30	535	35	150	8	117	14,912	**80.33**
Analysis	2	438	6	10	5	4	26	6	19	18	534	**82.02**
Backdoor	20	1	396	34	14	6	5	3	5	1	485	**81.65**
DoS	21	20	5	3177	45	53	15	12	9	10	3367	**94.36**
Exploits	1010	10	5	6	8522	1	7	20	3	16	9600	**88.77**
Fuzzers	6	6	8	12	0	3312	1005	0	9	9	4367	**75.84**
Generic	23	7	0	0	8	0	10,096	555	455	1	11,145	**90.59**
Reconnaissance	23	2	6	0	18	0	0	2389	8	10	2456	**97.27**
Shellcode	1	10	0	7	0	9	0	0	251	0	278	**90.29**
Worms	1	0	0	1	0	0	0	0	0	23	25	**92.00**
Truth overall	13085	2528	446	3252	8642	3920	11,189	3135	767	205	47,169	
Recall (%)	**91.54**	**17.33**	**88.79**	**97.69**	**98.61**	**84.49**	**90.23**	**76.20**	**32.73**	**11.22**		
F-measure (%)	85.567	28.609	85.069	95.997	93.433	79.933	90.409	85.459	48.038	20.001		

The bold values in the table represented the main results/outcomes of the proposed model

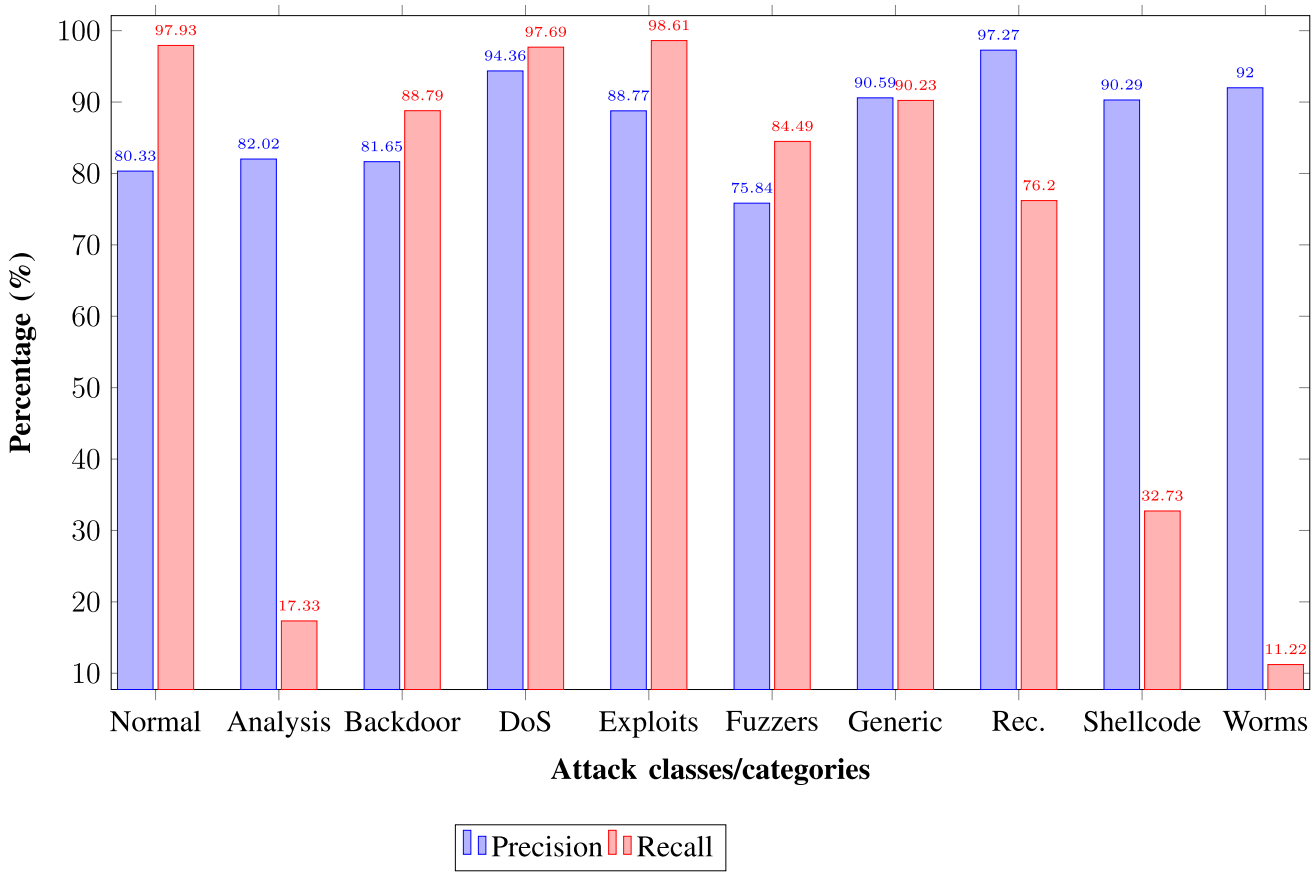

Fig. 5 Comparison of precision and recall of SDM (C4.5 classifier) under different classes

matrix also shows the overall classification value, truth overall value, precision, recall, and F-measure of each attack classes. The confusion matrix obtained from SDM, which uses the C4.5 classifier, is shown in Table 9. The precision and recall of this module are presented in Fig. 5. The confusion matrix obtained from ADM, which uses Naive-Based classifier, is shown in Table 10. The precision and recall of this module are presented in Fig. 6. The confusion matrix obtained from the proposed EHIDF, which uses the Meta-AdaBoostM1 algorithm is shown in Table 11. The precision and recall of the proposed EHIDF are presented in Fig. 7. Table 12 shows the performance of our proposed EHIDF in terms of different measurement metrics. This table illustrated that SDM which uses C4.5 classifier accuracy is 86.04%, AvgAcc is 68.88%, AttAcc is 66.37%, mean f-measure is 71.25%, ADR is 83.92%, and FAR is 8.4%, ADM which uses Naive-Based classifier accuracy is 86.94%, AvgAcc is 69.88%, AttAcc is 66.76%, mean f-measure is 72.49%, ADR is 82.91%, and FAR is 2.1%, and proposed EHIDF which uses Meta-AdaBoostM1 Algorithm accuracy is 90.25%, AvgAcc is 70.54%, AttAcc is 67.46%, mean f-measure is 74.35%, ADR is 86.95%, and FAR is 1.1%. A comparative bar graph in Fig. 8 shows the achieved results of three modules used in the proposed EHIDF. The result also illustrated that our proposed EHIDF has improved performance compared to the other two modules (SDM and ADM).

Test scenario: system statistics

The CPU load, RAM, and disk usage were measured during the experiment of the proposed EHIDF. This information can be retrieved using *psutil* python library by installing both psutil and Flask. It provides a modular interface and different functional tools that support such system stats. The different functions can be used to reports these system statistics. For instance, the function *psutil.cpu_percent(interval=1, percpu=True)* return the current CPU utilization (in percentage). The function takes the time interval (in seconds) as a parameter so that the utilization can be computed over a period of time. The function *psutil.virtual_memory()* provides the percentage of virtual memory (RAM) usage. The function *psutil.disk_usage('/')* return disk usage statistics including total, used and free space (in bytes) plus the percentage usage. All these system statistics are evaluated before moving to the edge computing scenario. After moving to edge computing, the same system statistics are evaluated. The computed

Table 10 Confusion matrix of ADM (Naive-Based classifier) on data UNSW-NB15

Class names	Normal	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms	Classification overall	Precision (%)
Normal	12378	1512	98	25	12	429	35	302	12	109	14,912	**83.01**
Analysis	0	458	3	4	8	0	34	4	13	10	534	**85.77**
Backdoor	10	2	412	31	0	16	5	4	5	0	485	**84.95**
DoS	11	29	5	2987	16	234	4	12	4	65	3367	**88.71**
Exploits	21	10	5	6	8734	145	507	153	3	16	9600	**90.98**
Fuzzers	153	5	0	45	0	3112	1005	29	9	9	4367	**71.26**
Generic	54	6	9	0	12	0	10356	642	65	1	11,145	**92.92**
Reconnaissance	12	3	10	15	9	0	0	2298	1	108	2456	**93.57**
Shellcode	1	10	0	7	0	9	0	0	251	0	278	**90.29**
Worms	0	0	0	0	1	1	0	0	2	21	25	**84.00**
Truth overall	12640	2035	542	3120	8792	3946	11946	3444	365	339	47169	
Recall (%)	**97.93**	**22.51**	**76.02**	**95.74**	**99.34**	**78.86**	**86.69**	**66.73**	**68.77**	**6.19**		
F-measure (%)	89.85	35.66	80.23	92.09	94.98	74.87	89.70	77.90	78.07	11.54		

The bold values in the table represented the main results/outcomes of the proposed model.

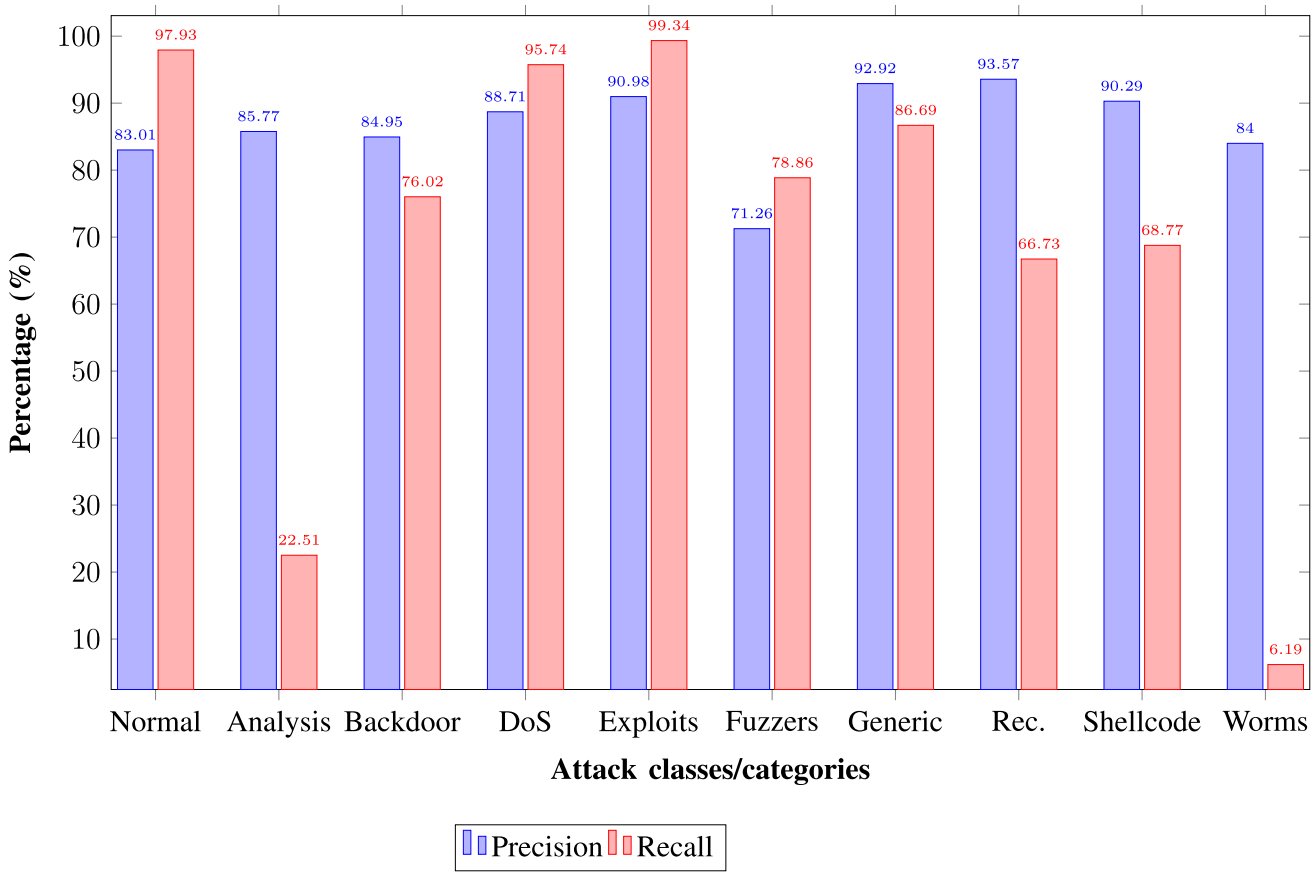

Fig. 6 Comparison of precision and recall of ADM (Naive-Based classifier) under different classes

system statistics results are shown in Fig. 9. The results illustrated that CPU utilization is 92.11% without MEC, 72.45% with MEC, RAM usage is 67.19% without MEC, 47.31% with MEC, disk usage is 28.81% without MEC and 24.56% with MEC. These values can show that CPU utilization, RAM usage, and disk usage are reduced using edge computing.

Comparative performance analysis

The achieved performance results are compared with the previous works. From the literature, three similar works have been found. The works are Papamartzivanos et al. [98], Kumar et al. [99], and Kumar et al. [100]. The performance are compared in terms of AvgAcc(%), AttAcc(%), Mean F-Measure(%), and ADR(%). The first work Papamartzivanos et al. [98] AvgAcc is 52.21%, AttAcc is 47.19%, Mean F-Measure 48.81%, and ADR is 63.76%. The second work Kumar et al. [99] AvgAcc is 65.21%, AttAcc is 57.01%, Mean F-Measure 68.13%, and ADR is 90.32%. The third work Kumar et al. [100] AvgAcc is 77.87%, AttAcc is 73.29%, Mean F-Measure 79.12%, and ADR is 86.15%. The comparative results presented in Table 13 illustrated that the proposed model shows improved performance compared to these three models. A comparative bar graph is shown in Fig. 10.

The accuracy and FAR is computed of the proposed EHIDF and compared it with the other previous works (Papamartzivanos et al. [98], Kumar et al. [99], Moustafa et al. [97], Kumar et al. [100], Panda et al. [101], Khraisat et al. [90], and Almogren [16]). The accuracy and FAR of all these works and proposed EHIDF are tabulated in Table 14. These results illustrated that the accuracy (90.25%) of the proposed EHIDF is high compared to other work, and FAR (1.1%) is low, showing the improved performance of the proposed EHIDF. A comparative figure of the accuracy and FAR is shown in Fig. 11.

The above comparative results illustrated that the proposed EHIDF has improved performance. Table 15 shows the improvement (in percentage) of the proposed EHIDF compared to other previous works. The tabulated results show that the accuracy of the proposed EHIDF is improved up to 10.78% and the FAR is reduced up to 93.03%.

Table 11 Confusion matrix of HDM (Meta-AdaBoostM1 Algorithm) on data UNSW-NB15

Class names	Normal	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms	Classification overall	Precision (%)
Normal	12845	1324	80	137	34	256	45	145	12	34	14,912	**86.14**
Analysis	20	475	4	4	9	0	15	4	23	0	540	**88.95**
Backdoor	8	4	432	14	9	0	6	9	3	0	485	**89.07**
DoS	13	12	4	3070	12	132	47	12	20	45	3367	**91.18**
Exploits	34	10	51	7	8987	145	234	53	23	56	9600	**93.62**
Fuzzers	36	23	78	23	0	3567	257	156	64	163	4367	**81.68**
Generic	121	6	9	0	10	0	10567	432	0	0	11,145	**94.81**
Reconnaissance	9	0	12	2	9	0	0	2345	12	67	2456	**95.48**
Shellcode	3	4	0	7	0	4	0	0	260	0	278	**93.52**
Worms	0	0	0	0	0	0	0	1	1	23	25	**92.00**
Truth overall	12980	1858	670	3264	9070	4104	11,171	3157	507	388	47,175	
Recall (%)	**98.29**	**25.56**	**64.48**	**94.06**	**99.08**	**86.91**	**94.59**	**74.28**	**62.20**	**5.93**		
F-measure (%)	91.812	39.716	74.805	92.595	96.272	84.217	94.703	83.556	74.713	11.138		

The bold values in the table represented the main results/outcomes of the proposed model

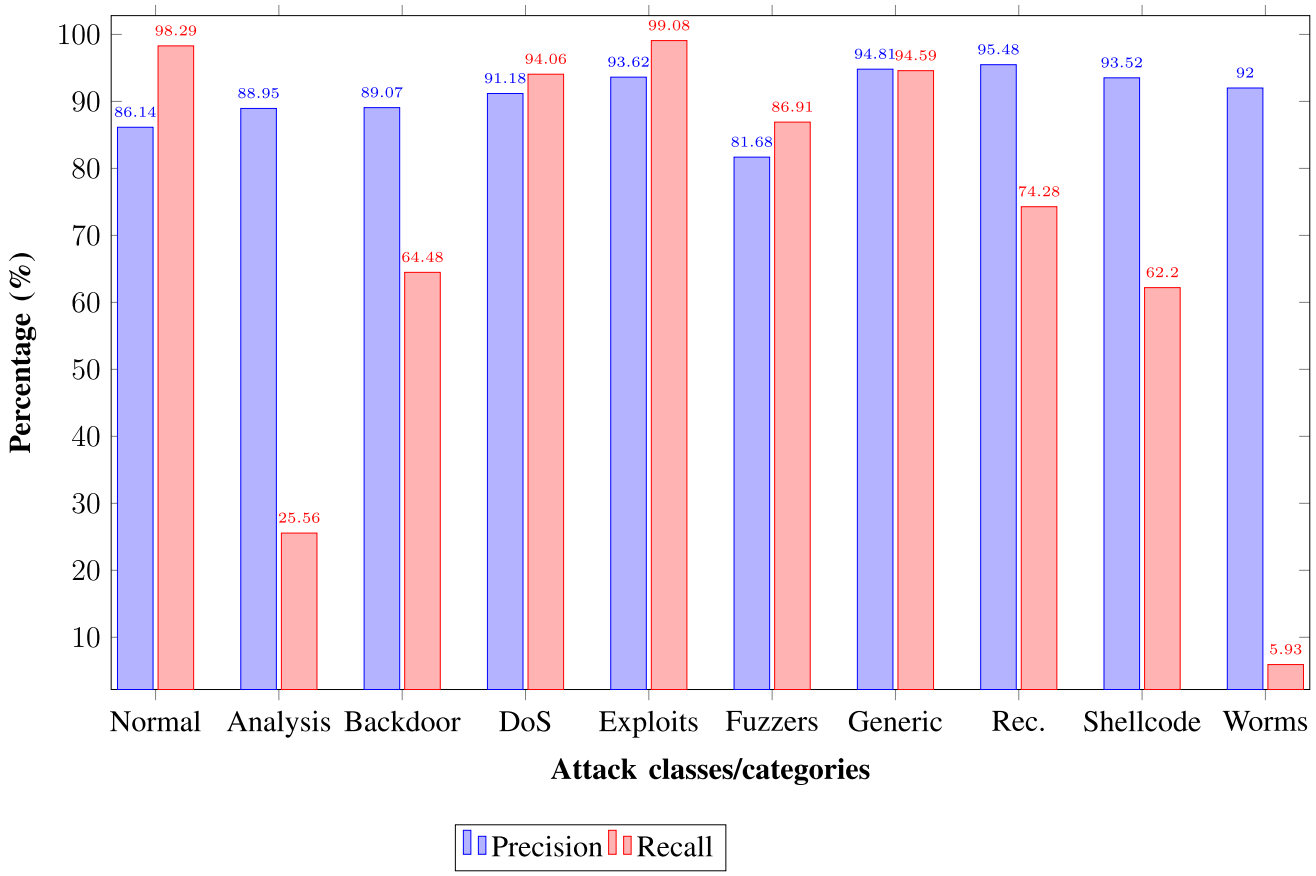

Fig. 7 Comparison of precision and recall of proposed EHIDF (Meta-AdaBoostM1 Algorithm) under different classes

Table 12 Performance metrics summary of our proposed EHIDF

Method	Accuracy (A %)	AvgAcc (%)	AttAcc (%)	Mean F-measure (%)	ADR (%)	FAR(%)
C4.5	86.04	68.88	66.37	71.25	83.92	8.4
Naive Bayes	86.94	69.88	66.76	72.49	82.91	2.1
EHIDF	**90.25**	**70.54**	**67.46**	**74.35**	**86.95**	**1.1**

The bold values in the table represented the main results/outcomes of the proposed model

Security analysis

The security analysis of the proposed EHIDF is based on the game-theoretic model [102,103]. Figure 12 shows the scenario on which the game model is developed. Table 16 describe the list of notation used in the game model.

Game theoretic model is designed as follows:-

The architecture of the deployment model is designed in Fig. 12. In this deployment model, three layers such as attack/end-user layer, mobile edge networking layer with IDF, and data storage layer are assumed. In the attack/end-user layer, the attacker tries to perform malicious activities. The attacker is a botmaster that generates and send malicious data packets to the edge devices to control the devices. The attack strategy fixed by the attacker is in the two ways to attain the highest payoff. In the first strategy, the attacker use knew regular attack to control the edge devices. In the second strategy, the attacker uses some unknown new sophisticated attack. The mobile edge networking layer with IDF consists of three different IDS as signature-based, anomaly-based and hybrid detection models. Here, decision model exists for alert generation. The SDM is responsible for detecting known attacks, and ADM is responsible for unknown attack detection. The hybrid model can handle both types of attack packets. The attacker tries to capture any edge resource that leads to resource exhaustion (R), which negatively impacts its payoff. Payoff (U) is a positive or negative score rewarded after every action played by a player. An equilibrium state will achieve when each player strategy leads to the maximum payoff for the other player's strategies. The third layer is the

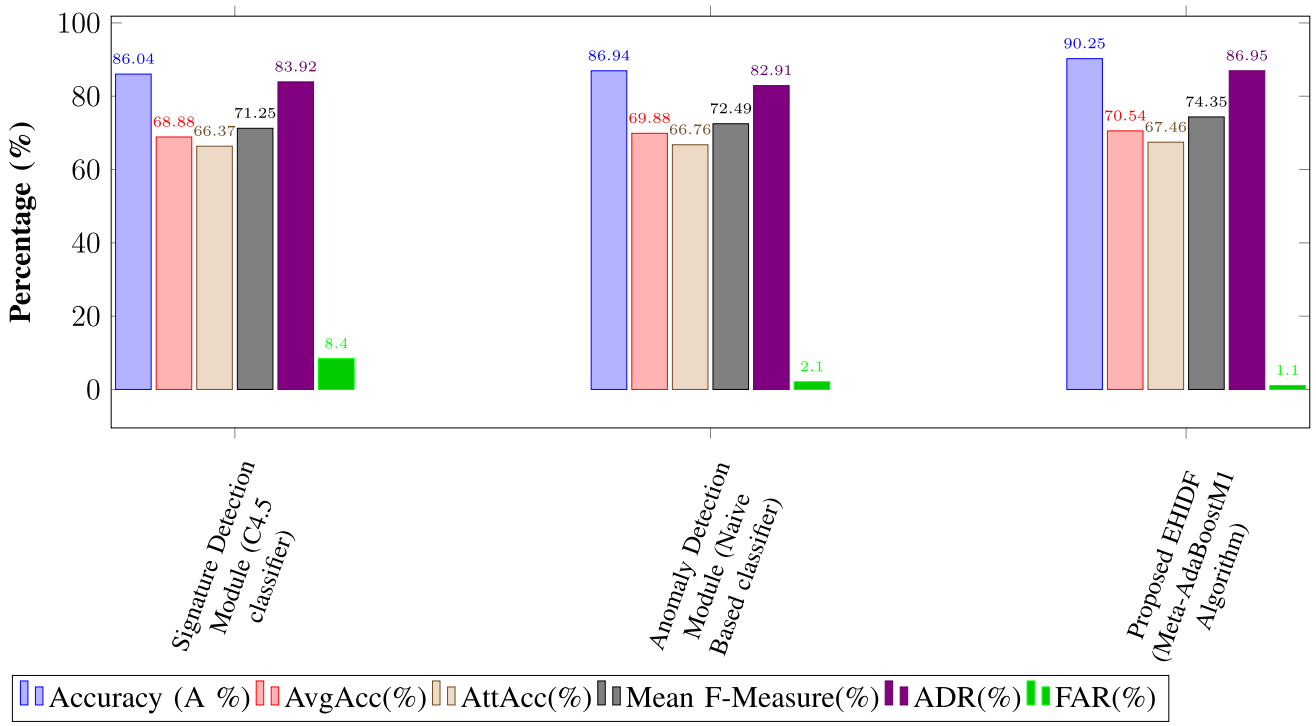

Fig. 8 Performance comparison of different modules used in our proposed EHIDF

data storage layer responsible for storing the edge resources and data.

The game-theoretic model includes two main components (player), and both players have different strategies to gain the maximum benefits. One is Attacker (A), and another one is a defender (D). The attacker may perform a common attack (A1) or a new sophisticated attack (A2). For identifying these two types of attack, the defence has two types of system. Signature detections of common attack through defence strategy (D1) and behavior detection of new attack through defence strategy (D2). The Game function is designed as:

$$G1 = \{(D, A), (ST_D, ST_A), (Be_D, Be_A)\},$$

where (ST_D, ST_A) are the strategies of two participants. (Be_D, Be_A) are the benefits earn by two participants. Hence strategy Matrix is defined as: $\begin{bmatrix} D_1A_1 & D_1A_2 \\ D_2A_1 & D_2A_2 \end{bmatrix}$ For (D_1A_1) when the attacker performs a common attack and the defender uses a signature-based detection policy (SBD), the benefits Be_{11} are represented in Table 17.

Now the total benefit function Be_D and Be_A for both participants is defined with the help of Eq. 4 and 5 .

$$\begin{aligned} Be_D = {} & pq\ Be_{11}(d) + (1-\rho)\,(1-q)\ Be_{22}(d) \\ & + (1-\rho)\,q\ Be_{21}(d) + p\,(1-q)\ Be_{12}(d), \end{aligned} \quad (4)$$

$$\begin{aligned} Be_A = {} & pq\ Be_{11}(a) + (1-\rho)\,(1-q)\ Be_{22}(a) \\ & + (1-q)\,p\ Be_{12}(a) + q\,(1-p)\ Be_{21}(a). \end{aligned} \quad (5)$$

After putting the values of benefits from the table final benefit is defined by Eq. (6) and (7).

$$Be_D = U_t - C_t - P_+[1 - \sigma\ pq - (1-p)\,(1-q)\gamma], \quad (6)$$

$$Be_A = [1 - \sigma\ pq - (1-p)\,(1-q)\gamma]\ P_+Q_+. \quad (7)$$

The Nash equilibrium solution states that if the maximum benefit function Be_A is matched with the maximum benefit function Be_D. Then, the game strategy is purely successful. so $q = \frac{\gamma}{(\sigma+\gamma)}$, $p = \frac{\sigma\gamma}{(\sigma+\gamma)}$

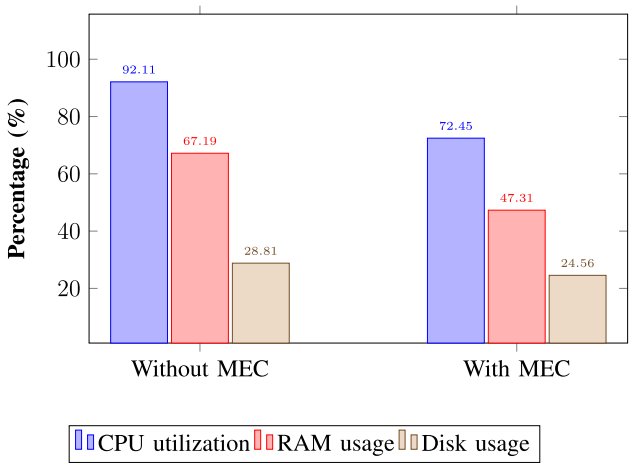

Fig. 9 System statistics

Table 13 Comparison of the performance metric values with the previous works

Research work	AvgAcc (%)	AttAcc (%)	Mean F-measure (%)	ADR (%)
Papamartzivanos et al. [98]	52.21	47.19	48.81	63.76
Kumar et al. [99]	65.21	57.01	68.13	90.32
Kumar et al. [100]	77.87	73.29	79.12	86.15
EHIDF	**70.54**	**67.46**	**74.35**	**86.95**

The bold values in the table represented the main results/outcomes of the proposed model

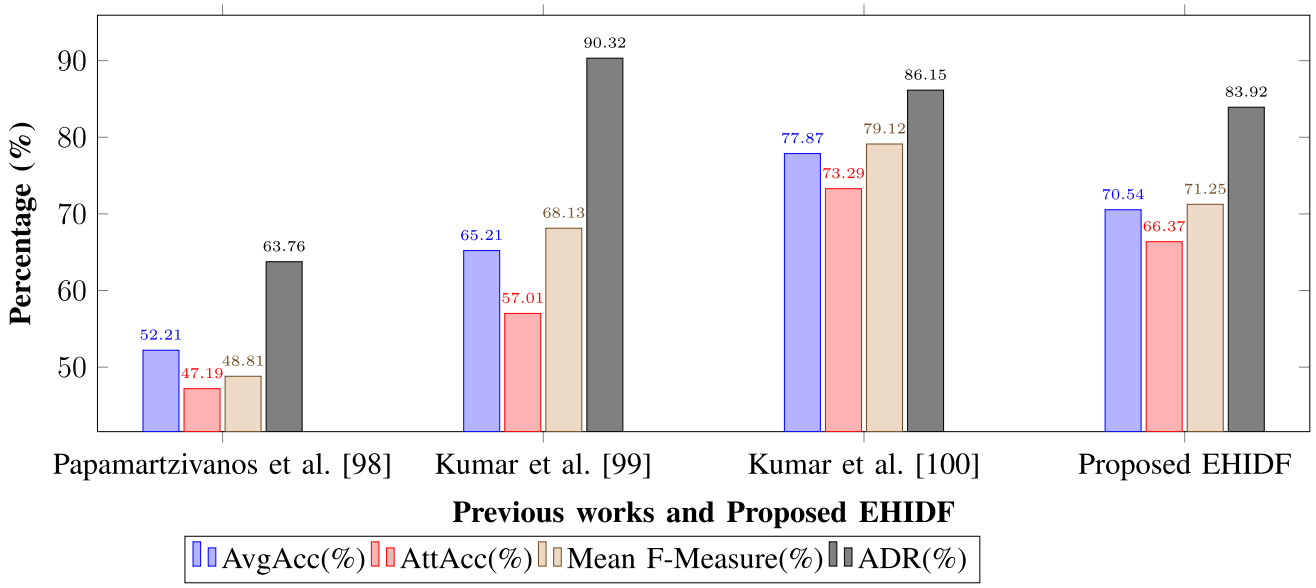

Fig. 10 Performance comparison of all the three previous works with the Proposed EHIDF

Table 14 Accuracy and FAR comparison with the previous works

Research work	Accuracy (A %)	FAR (%)
Papamartzivanos et al. [98]	84.33	2.61
Kumar et al. [99]	84.83	2.01
Moustafa et al. [97]	85.56	15.78
Kumar et al. [100]	88.92	3.80
Panda et al. [101]	81.47	12.85
Khraisat et al. [90]	83.24	8.3
Almogren [16]	85.73	2.98
EHIDF	**90.25**	**1.1**

The bold values in the table represented the main results/outcomes of the proposed model

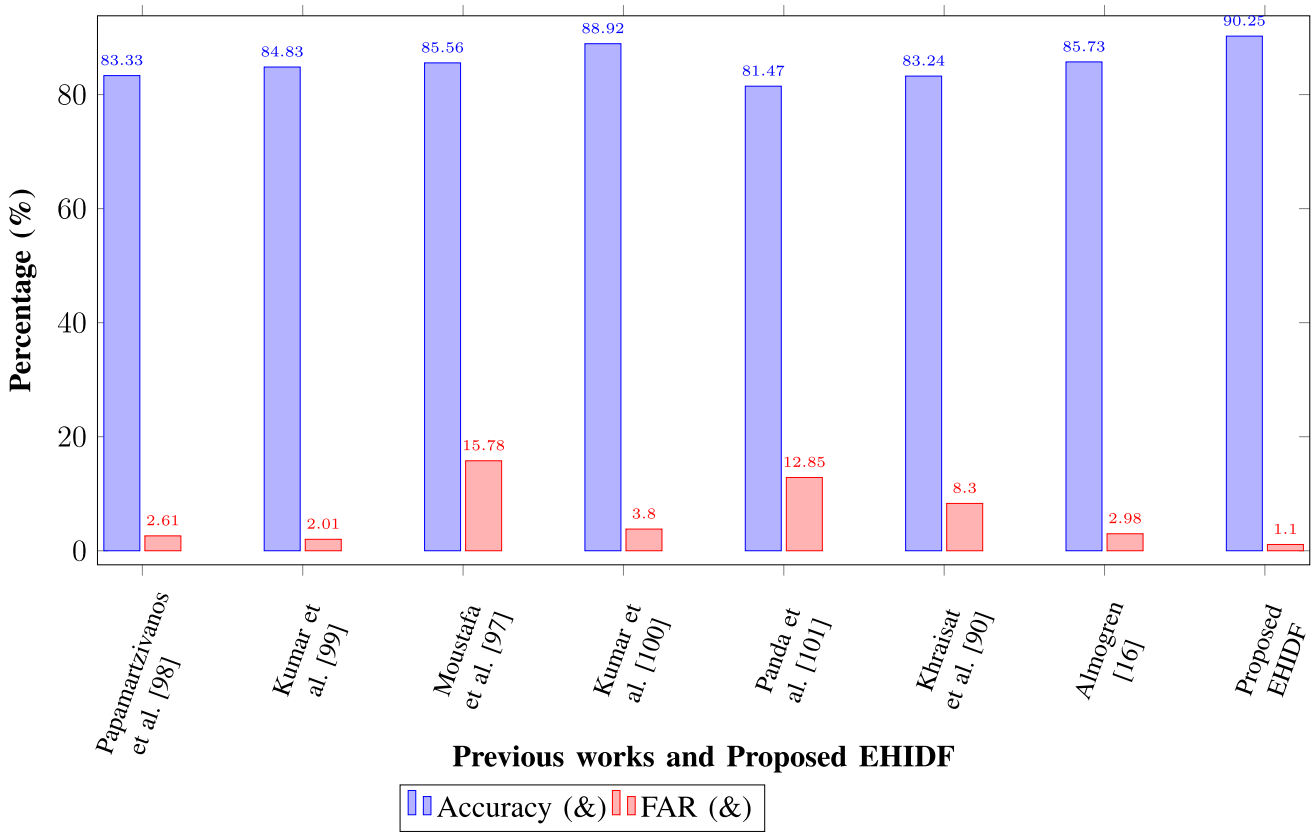

Fig. 11 Accuracy and FAR comparison of previous works with our proposed EHIDF

Table 15 Improvement of the proposed model accuracy with existing models

Research work	Accuracy improvement (%)	FAR reduction (-%)
Papamartzivanos et al. [98]	7.02	57.85
Kumar et al. [99]	6.39	45.27
Moustafa et al. [97]	5.48	93.03
Kumar et al. [100]	1.49	71.05
Panda et al. [101]	10.78	91.44
Ansam et al. [90]	8.42	86.75
Almogren [16]	5.27	63.09

Now for best strategies, the benefits will be maximized Be_D, Be_A and for that, it is differentiated with the probabilities, defined by Eqs. (8) and (9) .

$$\frac{dBe_D}{dp} = (\sigma + \gamma)\,(q - \gamma)\,P_+ = 0, \tag{8}$$

$$\frac{dBe_A}{dp} = (\gamma - (\sigma + \gamma))\,P_+ = 0. \tag{9}$$

So the strategy ST_D of the defender to defend attack through SBD and ABD is the equal probability with the strategy ST_A of the attacker to play common attack and new attack. Thus, Nash equilibrium is maintained when $ST_D = (p, 1\text{-}p)$ and $ST_A = (q, 1\text{-}q)$. If the average detection rate of a common attack is fixed, then for the high value of γ, i.e., average detection rate of a new attack will increase. So, the attacker will reduce to perform the new attack.

Conclusion and future scope

This work identified several intrusion detection problems that disturb the mobile edge networks by compromising availability, integrity, and confidentiality. The new or unknown intrusive traffic cannot be detected by a normal firewall as well as by using current ML-based approaches. This paper proposed an EHIDF for a mobile edge computing environment to overcome the current intrusion detection

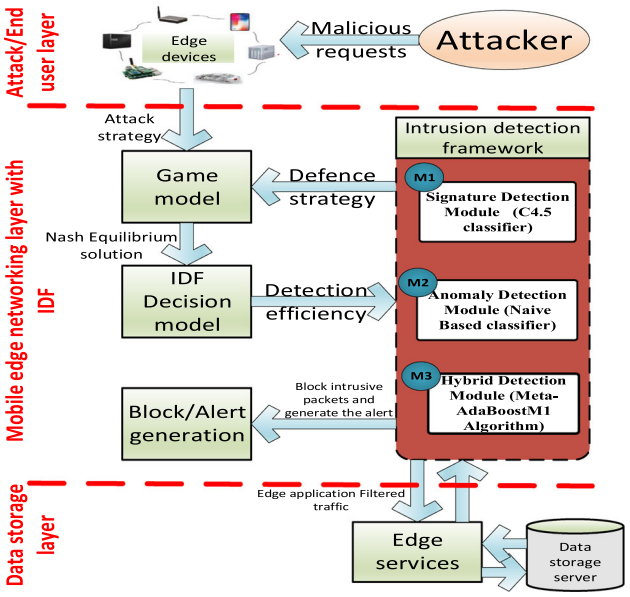

Fig. 12 Game theoretical-based security analysis model

problems. The proposed detection framework consists of different detection modules with different classifiers. It is capable of detecting unknown or new attacks with low FAR into mobile edge infrastructure. The achieved implementation results demonstrated that SDM accuracy is 86.04% and FAR is 8.4%, ADM accuracy is 86.94% and FAR is 2.1%, and EHIDF accuracy is 90.25% and FAR is 1.1%. These results are compared with previous works and found improved performance (accuracy is improved up to 10.78% and FAR is reduced up to 93%). A game-theoretical-based security analysis is also discussed. The proposed framework is limited to only 15 feature vectors. The considered feature vectors may be independent of each other, which leads high error rate. Thus, selecting more than 15 dependent feature vectors may give more accurate results. In the future, the framework's performance will be improved by including other ML-based techniques. The work can be implemented on different datasets such as UNB ISCX 2012, IDEVAL

Table 16 Notations and its description

Symbol	Description
U_t	The utility of operating system of server
C_t	The protection cost of the node
V_t	The utility of edge-based IDF
Q_t	The attack cost of the attacker
P_t	The utility of attacker
W_t	Wait time for next attack
γ	The average detection accuracy for new attack
σ	The average detection accuracy for common attack
ρ	The problem of common attack detection of defender
(1- ρ)	The problem of new attack detection of defender
q	The problem of enforcing common attack by the attacker
(1-q)	The problem of performing new attack by the attacker

Table 17 Benefits of Be_{11}

Participants	Be_{11}	Be_{12}	Be_{21}	Be_{22}
Attacker	$(1\text{-}\rho)P_+\text{-}Q_+$	$P_+\text{-}Q_+$	$P_+\text{-}Q_+$	$(1\text{-}\gamma)P_+\text{-}Q_+$
Defender	$U_t\text{-}C_t\text{-}(1\text{-}\rho)P_+$	$U_t\text{-}C_t\text{-}P_+$	$U_t\text{-}C_t\text{-}P_+$	$U_t\text{-}C_t\text{-}(1\text{-}\gamma)P_+$

datasets, and ADFA, which may improve the framework's accuracy.

Declarations

Conflict of interest There is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abbas N, Zhang Y, Taherkordi A, Skeie T (2017) Mobile edge computing: a survey. IEEE Internet Things J 5(1):450–465
2. Khan WZ, Ahmed E, Hakak S, Yaqoob I, Ahmed A (2019) Edge computing: a survey. Future Gener Comput Syst 97:219–235
3. Siriwardhana Y, Porambage P, Liyanage M, Ylianttila M (2021) A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects. IEEE Commun Surv Tutor 23(2):1160–1192
4. Cao H, Wachowicz M, Cha S (2017) Developing an edge computing platform for real-time descriptive analytics. In: 2017 IEEE International Conference on Big Data (Big Data), IEEE, pp 4546–4554
5. Sabella D, Vaillant A, Kuure P, Rauschenbach U, Giust F (2016) Mobile-edge computing architecture: the role of MEC in the Internet of Things. IEEE Consum Electron Mag 5(4):84–91
6. Yueyue DD, Maharjan S, Qiao G, Zhang Y (2019) Artificial intelligence empowered edge computing and caching for internet of vehicles. IEEE Wirel Commun 26(3):12–18
7. Asif-Ur-Rahman M, Afsana F, Mahmud M, Kaiser MS, Ahmed MR, Kaiwartya O, James-Taylor A (2018) Toward a heterogeneous mist, fog, and cloud-based framework for the internet of healthcare things. IEEE Internet Things J 6(3):4049–4062
8. Farhin F, Shamim KM, Mahmud M (2020) Towards secured service provisioning for the Internet of Healthcare Things. In: 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), IEEE, pp 1–6
9. Farhin F, Shamim KM, Mahmud M (2021) Secured smart healthcare system: blockchain and bayesian inference based approach. Proceedings of international conference on trends in computational and cognitive engineering. Springer, Berlin, pp 455–465
10. Shamim KM, Zenia N, Tabassum F, Mamun SA, Arifur RM, Shahidul IM, Mahmud M (2021) 6G Access network for intelligent internet of healthcare things: opportunity, challenges, and research directions. Proceedings of international conference on trends in computational and cognitive engineering. Springer, Berlin, pp 317–328
11. Furnell S (2004) Enemies within: the problem of insider attacks. Comput Fraud Secur 2004(7):6–11
12. Sharma P, Sengupta J, Suri PK (2019) Survey of intrusion detection techniques and architectures in cloud computing. Int J High Perform Comput Netw 13(2):184–198
13. Mahesh Yadav YR (2019) Effective analysis of malware detection in cloud computing. Comput Secur 83:14–21
14. Roman R, Lopez J, Mambo M (2018) Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. Future Gener Comput Syst 78:680–698
15. Vimal S, Suresh A, Subbulakshmi P, Pradeepa S, Kaliappan M (2020) Edge computing-based intrusion detection system for smart cities development using IoT in urban areas. Internet of things in smart technologies for sustainable urban development. Springer, Berlin, pp 219–237
16. Almogren AS (2020) Intrusion detection in Edge-of-Things computing. J Parallel Distrib Comput 137:259–265
17. Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961
18. Liang C, Shanmugam B, Azam S, Karim A, Islam A, Zamani M, Kavianpour S, Idris NB (2020) Intrusion detection system for the Internet of Things based on blockchain and multi-agent systems. Electronics 9(7):1120
19. Mudgerikar A, Sharma P, Bertino E (2020) Edge-based intrusion detection for IoT devices. ACM Trans Manag Inform Syst (TMIS) 11(4):1–21
20. Cao X, Fu Y, Chen B (2020) Packet-based intrusion detection using Bayesian topic models in mobile edge computing. Secur Commun Netw. https://doi.org/10.1155/2020/8860418
21. Eskandari M, Haider Janjua Z, Vecchio M, Antonelli F (2020) Passban IDS: an intelligent anomaly-based intrusion detection system for IoT edge devices. IEEE Internet Things J 7(8):6882–6897
22. Mendonça RV, Teodoro AAM, Rosa RL, Saadi M, Carrillo MD, Nardelli PHJ, Rodríguez DZ (2021) Intrusion detection system based on fast hierarchical deep convolutional neural network. IEEE Access 9:61024–61034
23. Abid Salih A, Mohsin AA (2021) Evaluation of classification algorithms for intrusion detection system: a review. J Soft Comput Data Min 2(1):31–40
24. Ramaiah M, Chandrasekaran V, Ravi V, Kumar N (2021) An intrusion detection system using optimized deep neural network architecture. Trans Emerg Telecommun Technol 32(4):e4221
25. Singh BN, Khari M (2021) A survey on hybrid intrusion detection techniques. Research in intelligent and computing in engineering. Springer, Berlin, pp 815–825
26. Shahraki A, Abbasi M, Haugen O (2020) Boosting algorithms for network intrusion detection: a comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. Eng Appli Artif Intell 94:103770
27. Sha K, Yang TA, Wei W, Davari S (2020) A survey of edge computing-based designs for IoT security. Digit Commun Netw 6(2):195–202
28. Besharati E, Naderan M, Namjoo E (2019) LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. J Ambient Intell Humaniz Comput 10(9):3669–3692
29. Bakshi A, Dujodwala YB (2010) Securing cloud from DDOS attacks using intrusion detection system in virtual machine. In: Communication Software and Networks, 2010. ICCSN10. Second International Conference, IEEE, pp 260–264
30. Schapire RE (2003) The boosting approach to machine learning: an overview Nonlinear estimation and classification. Springer, Berlin, pp 149–171
31. Li Y, Xia J, Zhang S, Yan J, Ai X, Dai K (2012) An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Syst Appl 39(1):424–430

32. Roschke S, Cheng F, Meinel C (2009) An extensible and virtualization-compatible IDS management architecture. In: 2009 Fifth International Conference on Information Assurance and Security, volume 2, IEEE, pp 130–134
33. Toosi AN, Kahani M (2007) A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Comput Commun 30(10):2201–2212
34. Aljawarneh S, Aldwairi M, Bani Yassein M (2018) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J Comput Sci 25:152–160
35. Sazzadul HM, Mukit M, Bikas M, Naser A (2012) An implementation of intrusion detection system using genetic algorithm. Int J Netw Secur Appl (IJNSA) 4(2):109–120
36. Subramanian U, Ong HS (2014) Analysis of the effect of clustering the training data in naive bayes classifier for anomaly network intrusion detection. J Adv Comput Netw 2(1):85–88
37. Zhengbing H, Jun S, Shirochin VP (2007) An intelligent lightweight intrusion detection system with forensics technique. In: 2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IEEE, pp 647–651
38. Mohammad IL (2010) Anomaly network intrusion detection system based on distributed time-delay neural network (DTDNN). J Eng Sci Technol 5(4):457–471
39. Chen W-H, Hsu S-H, Shen H-P (2005) Application of SVM and ANN for intrusion detection. Comput Oper Res 32(10):2617–2634
40. Hajimirzaei B, Jafari NN (2019) Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Express 5(1):56–59
41. Horng S-J, Su M-Y, Chen Y-H, Kao T-W, Chen R-J, Lai J-L, Perkasa CD (2011) A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Syst Appl 38(1):306–313
42. Varshovi A, Rostamipour M, Sadeghiyan B (2014) A fuzzy Intrusion detection system based on categorization of attacks. In: 2014 6th Conference on Information and Knowledge Technology (IKT), pp 50–55
43. Akashdeep, Manzoor I Kumar N (2017) A feature reduced intrusion detection system using ANN classifier. Expert Syst Appl 8:249–257
44. Dovom ME, Azmoodeh A, Dehghantanha A, Newton DE, Parizi RM, Karimipour H (2019) Fuzzy pattern tree for edge malware detection and categorization in IoT. J Syst Archit 97:1–7
45. Hassan MMM (2013) Network intrusion detection system using genetic algorithm and fuzzy logic. Int J Innov Res Comput Commun Eng 1(7):1435–1445
46. Raja S, Ramaiah S (2017) An efficient fuzzy-based hybrid system to cloud intrusion detection. Int J Fuzzy Syst 19(1):62–77
47. Keegan N, Ji S-Y, Chaudhary A, Concolato C, Yu B, Jeong DH (2016) A survey of cloud-based network intrusion detection analysis. Hum Centric Comput Inform Sci 6(1):1–16
48. Hamad Hatem, Al-Hoby Mahmoud (2012) Managing Intrusion Detection as a Service in Cloud Networks. International Journal of Computer Applications 41(1):35–40
49. Xuren W, Famei H, Rongsheng X (2006) Modeling intrusion detection system by discovering association rule in rough set theory framework. In: 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA06), IEEE, p 24
50. Gauthama RMR, Somu N, Kirthivasan K, Liscano R, Shankar SSVS (2017) An efficient intrusion detection system based on hypergraph: genetic algorithm for parameter optimization and feature selection in support vector machine. Knowl-Based Syst 134:1–12
51. Houmansadr A, Zonouz SA, Berthier R (2011) A cloud-based intrusion detection and response system for mobile phones. In: 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), IEEE, pp 31–32
52. Li H, Liu D (2010) Research on intelligent intrusion prevention system based on snort. Int Conf Comput Mechatron Control Electron Eng 1:251–253
53. Li L, Yang D-Z, Shen F-C (2010) A novel rule-based Intrusion Detection System using data mining. Int Conf Comput Sci Inform Technol (CSIT) 6:169–172
54. Senthilnayaki B, Venkatalakshmi K, Kannan A (2013) An intelligent intrusion detection system using genetic based feature selection and Modified J48 decision tree classifier. In: 2013 Fifth International Conference on Advanced Computing (ICoAC), IEEE, pp 1–7
55. Vieira K, Schulter A, Westphall C, Westphall C (2010) Intrusion detection for grid and cloud computing. IT Prof 12(4):38–43
56. Xia T, Qu G, Hariri S, Yousif M (2005) An efficient network intrusion detection method based on information theory and genetic algorithm. In: PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, IEEE, pp 11–17
57. Pradeep MKK, Saravanan M, Thenmozhi M, Vijayakumar K (2021) Intrusion detection system based on GA-fuzzy classifier for detecting malicious attacks. Concurr Comput 33(3):e5242
58. Botha M, Von Solms R, Perry K, Loubser E, Yamoyany G (2002) The utilization of artificial intelligence in a hybrid intrusion detection system. In: SAICSIT 02: Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology, South African Institute for Computer Scientists and Information Technologists, pp 149–155
59. Shamshirband S, Fathi M, Chronopoulos AT, Montieri A, Palumbo F, Pescapè A (2020) Computational intelligence intrusion detection techniques in mobile cloud computing environments: review, taxonomy, and open research issues. J Inform Secur Appl 55:102582
60. Mugabo E, Zhang Q-Y (2020) Intrusion detection method based on support vector machine and information gain for mobile cloud computing. IJ Netw Secur 22(2):231–241
61. Basavaraj D, Tayeb S (2021) Limitations and challenges of Fog and edge-based computing. In: 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), IEEE, pp 1–6
62. Sun B, Osborne L, Xiao Y, Guizani S (2007) Intrusion detection techniques in mobile ad hoc and wireless sensor networks. IEEE Wirel Commun 14(5):56–63
63. Lai S, Zhao R, Tang S, Xia J, Zhou F, Fan L (2021) Intelligent secure mobile edge computing for beyond 5G wireless networks. Phys Commun 45:101283
64. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in Mobile Edge Computing: a deep learning-based hybrid approach. J Netw Comput Appl 178:102974
65. Abdenacer N, Hangxing W, Nabil AN, Dhelim S, Ning H (2021) A novel framework for mobile edge computing by optimizing task offloading. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3064225
66. Chen J, Ran X (2019) Deep learning with edge computing: a review. Proc IEEE 107(8):1655–1674
67. Lin F, Zhou Y, An X, You I, Choo K-KR (2018) Fair resource allocation in an intrusion-detection system for edge computing: ensuring the security of Internet of Things devices. IEEE Consum Electron Mag 7(6):45–50
68. Yao H, Gao P, Zhang P, Wang J, Jiang C, Lu L (2019) Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection. IEEE Netw 33(5):75–81

69. Pešić S, Ivanović M, Radovanović M, Bădică C (2020) CAAVI-RICS model for observing the security of distributed IoT and edge computing systems. Simul Model Pract Theory 105:102125
70. Li Z, Chen J, Zhang J, Cheng X, Chen B (2020) Detecting advanced persistent threat in edge computing via federated learning. International conference on security and privacy in digital economy. Springer, Berlin, pp 518–532
71. Pacheco J, Benitez VH, Felix-Herran LC, Satam P (2020) Artificial neural networks-based intrusion detection system for Internet of Things fog nodes. IEEE Access 8:73907–73918
72. Aravamudhan P, Kanimozhi T (2021) A survey on intrusion detection system and prerequisite demands in IoT networks. J Phys 1916:012179
73. Naseer Qureshi K, Jeon G, Piccialli F (2021) Anomaly detection and trust authority in artificial intelligence and cloud computing. Comput Netw 184:107647
74. Zhang C, Chen Y, Meng Y, Ruan F, Chen R, Li Y, Yang Y (2021) A novel framework design of network intrusion detection based on machine learning techniques. Secur Commun Netw. https://doi.org/10.1155/2021/6610675
75. Markham T, Payne C (2001) Security at the network edge: a distributed firewall architecture. In: Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, volume 1, IEEE, pp 279–286
76. Muna AL-H, Moustafa N (2018) Identification of malicious activities in industrial Internet of Things based on deep learning models. J Inform Secur Appl 41:1–11
77. Meng W, Wang Y, Li W, Liu Z, Li J, Probst CW (2018) Enhancing intelligent alarm reduction for distributed intrusion detection systems via edge computing. Australasian conference on information security and privacy. Springer, Berlin, pp 759–767
78. Li Q, Hou J, Meng S, Long H (2020) GLIDE: a game theory and data-driven mimicking linkage intrusion detection for edge computing networks. Complexity. https://doi.org/10.1155/2020/7136160
79. Kozik R, Choraś M, Ficco M, Palmieri F (2018) A scalable distributed machine learning approach for attack detection in edge computing environments. J Parallel Distrib Comput 119:18–26
80. Aloqaily M, Otoum S, Al Ridhawi I, Jararweh Y (2019) An intrusion detection system for connected vehicles in smart cities. Ad Hoc Netw 90:101842
81. Sudqi Khatar B, Abdul WAWB, Idris IMYIB, Abdulla HM, Ahmed IA (2019) A lightweight perceptron-based intrusion detection system for fog computing. Appl Sci 9(1):178
82. Sharma R, Aun CC, Leckie C (2020) Evaluation of centralised vs distributed collaborative intrusion detection systems in multi-access edge computing. In: 2020 IFIP Networking Conference (Networking), IEEE, pp 343–351
83. Almiani M, AbuGhazleh A, Al-Rahayfeh A, Atiewi S, Razaque A (2020) Deep recurrent neural network for IoT intrusion detection system. Simul Model Pract Theory 101:102031
84. Liu H, Zhang S, Zhang P, Zhou X, Shao X, Pu G, Zhang Y (2021) Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing. IEEE Trans Veh Technol. https://doi.org/10.1109/TVT.2021.3076780
85. Gong Y, Liu Y, Yin C (2021) A novel two-phase cycle algorithm for effective cyber intrusion detection in edge computing. EURASIP J Wirel Commun Netw 2021(1):1–22
86. Idrissi I, Mostafa AM, Moussaoui O (2021) A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for IoT. Int J Comput Digit Syst:1–8
87. Alghamdi R, Bellaiche M (2021) A deep intrusion detection system in lambda architecture based on edge cloud computing for IoT. In: 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), IEEE, pp 561–566
88. Hajisalem V, Babaie S (2018) A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. Comput Netw 136:37–50
89. Aydın MA, Zaim AH, Ceylan KG (2009) A hybrid intrusion detection system design for computer network security. Comput Electr Eng 35(3):517–526
90. Khraisat A, Gondal I, Vamplew P, Kamruzzaman J, Alazab A (2020) Hybrid intrusion detection system based on the stacking ensemble of C5 decision tree classifier and one class support vector machine. Electronics 9(1):173
91. Al-Yaseen WL, Othman ZA, Nazri MZA (2017) Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. Expert Syst Appl 67:296–303
92. Cepheli Ö, Büyükçorak S, Karabulut K (2016) Hybrid intrusion detection system for DDoS attacks. J Electr Comput Eng. https://doi.org/10.1155/2016/1075648
93. Alghayadh F, Debnath D (2021) A hybrid intrusion detection system for smart home security based on machine learning and user behavior. Adv Internet Things 11(1):10–25
94. Yang L, Moubayed A, Shami A (2021) MTH-IDS: a multi-tiered hybrid intrusion detection system for internet of vehicles. IEEE Internet Things 9:1–18
95. Moustafa N, Creech G, Slay J (2017) Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models. Data analytics and decision support for cybersecurity. Springer, Berlin, pp 127–156
96. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, pp 1–6
97. Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inform Secur J 25(1–3):18–31
98. Papamartzivanos D, Gómez Mármol F, Kambourakis G (2018) Dendron: genetic trees driven rule induction for network intrusion detection systems. Future Gener Comput Syst 79:558–574
99. Kumar V, Sinha D, Das AK, Pandey SC, Goswami RT (2020) An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset. Clust Comput 23(2):1397–1418
100. Kumar V, Das AK, Sinha D (2019) UIDS: a unified intrusion detection system for IoT environment. Evolut Intell 14:47–59
101. Panda M, Abraham A, Patra MR (2010) Discriminative multinomial Naïve Bayes for network intrusion detection. In: 2010 Sixth International Conference on Information Assurance and Security, IEEE, pp 5–10
102. Singh Gill K, Saxena S, Sharma A (2020) GTM-CSec: game theoretic model for cloud security based on IDS and honeypot. Comput Secur 92:101732
103. Han L, Zhou M, Jia W, Dalil Z, Xingbo X (2019) Intrusion detection model of wireless sensor networks based on game theory and an autoregressive model. Inform Sci 476:491–504

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.