Internet of Things 23 (2023) 100843

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things

An intelligent context-aware threat detection and response model for smart cyber-physical systems

Zainab Noor [a], Sadaf Hina [b,*], Faisal Hayat [a], Ghalib A Shah [c]

[a] Department of Computer Engineering, University of Engineering and Technology, Lahore, Pakistan
[b] Department of Computer Science, School of Science, Engineering & Environment, University of Salford, Manchester, United Kingdom
[c] Department of Cybersecurity, Air University, Islamabad, Pakistan

ARTICLE INFO

Keywords:
Smart homes
Cyber-physical systems
Context-aware IoT security
Network traffic analysis
Machine learning

ABSTRACT

Smart cities, businesses, workplaces, and even residences have all been converged by the Internet of Things (IoT). The types and characteristics of these devices vary depending on the industry 4.0 and have rapidly increased recently, especially in smart homes. These gadgets can expose users to serious cyber dangers because of a variety of computing constraints and vulnerabilities in the security-by-design concept. The smart home network testbed setup presented in this study is used to evaluate and validate the protection of the smart cyber-physical system. The context-aware threat intelligence and response model identifies the states of the aligned smart devices to distinguish between real-world typical and attack scenarios. It then dynamically writes specific rules for protection against potential cyber threats. The context-aware model is trained on IoT Research and Innovation Lab - Smart Home System (IRIL-SHS) testbed dataset. The labeled dataset is utilized to create a random forest model, which is subsequently used to train and test the context-aware threat intelligence SHS model's effectiveness and performance. Finally, the model's logic is used to gain rules to be included in Suricata signatures and the firewall rulesets for the response system. Significant values of the measuring parameters were found in the results. The presented model can be used for the real-time security of smart home cyber-physical systems and develops a vision of security challenges for Industry 4.0.

1. Introduction

The exponential rise in IoT devices has also resulted in the enhancement of communication services [1]. Our way of life, including how we live, work, communicate, study, manage our health, and enjoy ourselves, has been revolutionized and simplified by IoT devices. It is predicted that there will be more than 13 trillion connected IoT devices by 2030 [2]. As a result of this exponential growth, more security challenges are emerging. One of the most well-known applications of the Internet of Things is the smart home, which poses a significant security problem in preventing threats from unidentified sources. Smart home gadgets are being used by over two-fifths of developed countries, which is more than nearly twice the percentage of developing countries, according to the latest survey [3]. Even though 98% of respondents are cautious about confidentiality linked with their gadgets, more than 50% have taken no action to safeguard them [4]. The most prevalent attacks on IoT devices are creating a center for bitcoin mining as well as cybercrime, data leaks, man-in-the-middle attacks, DDoS attacks, and spamming [5]. Furthermore, an IoT device in a smart home might be

* Corresponding author.
E-mail address: s.hina@salford.ac.uk (S. Hina).

https://doi.org/10.1016/j.iot.2023.100843

Available online 9 June 2023

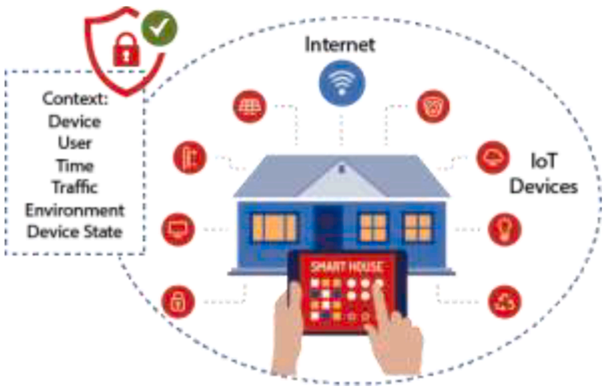


Fig. 1. Context-aware smart home system.

exploited to access and formulate a combination of actions to intrude into other devices of the house. Even without physical intrusion, silent surveillance through a hacked or malfunctioned gadget can result in a strategic campaign to compromise other vulnerable devices and run an intimate cyberstalk on the residents [6]. Smart home defense is an evolving notion with no well-defined threat detection and response model that fits different threat scenarios.

The context is any knowledge or information that may be used to comprehend the situation of the underlying environment in which the application is running [7]. Three crucial context factors, according to the authors [8] are where you are, who you are with, and what resources are close by. Location, time, identity, environment, network, history, and activity are the major context used for the development of any context-aware system.

A smart home is envisioned as a novel environment where the Internet of Things (IoT) is widely used. The Internet of Things (IoT), which is built on communication and information technology (ICT), fundamentally alters how we live by changing virtual interactions between people in a variety of scenarios, from the workplace to interpersonal connections [9]. Consequently, the creation of a smart home necessitates the seamless integration of user interactions, physical items, and human engagement. Particularly a smart house is thought to offer individuals a new kind of smart environment and lifestyle, greatly enhancing our quality of life [10]. Fig. 1 shows the context-aware smart home system.

The Internet of Things is a dynamic network with continually changing things and mobility of users and conventional fixed security solutions are found ineffective as a result. The context-aware defense has brought interesting aspects to classical security by making use of context information to make decisions [11]. Various machine learning methodologies have been proposed to control and expedite the process of creating context-specific access control for IoT devices [12]. The contextual information from the past records of ambient IoT access was used to determine whether to grant or deny a certain IoT direct access request in the future [13]. There were many network-based security solutions proposed but previously proposed technologies offer restricted assistance in deciphering the traffic in between the internet-connected system environment and they necessitate in-depth knowledge of the network standards and the unique infrastructure set-up that leads to the context-aware threat detection system [14]. Context-aware security models had to contend with the question of whether context features will generate more precise and determined security results [15]. Consequently, mapping the context of the device and the network into a model that delivers the best results in terms of security breach detection and decreasing false positive rates in alert generation is a huge challenge.

Numerous methods currently in use were designed primarily for static networks and are therefore unable to detect IoT devices mobile activity [16]. Most methods guarantee the accuracy of contextual data and the seamless transfer of that data between devices via the cloud, which is not assessed in a comprehensive scenario [17]. Some methods encrypt data using pricey cryptographic algorithms like Advanced Encryption Standard and Rivest-Shamir-Adleman RSA, however, these are inefficient for IoT devices with limited resources because they demand more computational power to function [18]. Moreover, most research studies tested and refined their proposed context-aware security models using publicly available datasets [19].

To address the deficiencies of current threat detection systems and to achieve the objectives, we proposed an effective context-aware threat intelligence and response model for smart home systems. The existing threat/ attack detection models lack real-time smart cyber-physical systems' datasets which challenges the advancement in the optimization of intelligent methods for context-aware threat detection and response. The development of state-of-the-art tracking and detection methods requires instantaneous access to sensors' data. To track deployed IoT devices' whereabouts and take prompt action in the smart home, this novel research study employed the idea of the context-aware system which collects network-related dynamic information from the equipped IoT devices. The dynamic rule writing technique is then used for the response model. This research also provides an innovative smart home network testbed architecture which is used to analyze and secure smart cyber-physical systems. With the deployed IoT Research and Innovation Lab - Smart Home System (IRIL-SHS) testbed, the proposed context-aware threat intelligence model identified the states of the connected smart devices and built a contextual model to differentiate between real-world normal and attack scenarios. With the contextual features extracted, the labeled dataset is later used to train and validate the efficiency and performance of the context-aware threat intelligence SHS model.

The remaining sections of the research are structured as follows: Section 2 presents a comprehensive overview of the available context-aware threat detection and response mechanisms literature along with a comparative analysis. Section 3 describes dataset generation using the topology of IoT devices. The context-aware threat detection and response model is presented in Section 4. Section 5 presents the performance analysis through results and a discussion of our suggested model with the comparison of previous studies. Finally, this paper outlines the conclusion in Section 6 and talks about future work.

2. Literature review

Many studies on context awareness have been conducted to build various context-aware systems that considerably improve people's daily lives. Starting with obtaining expertise in context, acquiring context, and defining the rules, the system must determine what adaptations are required [20]. When adopting and accessing context-aware systems, a lot of variables contribute to consumers developing substantial privacy concerns and potential security risks. Users are also exposed to a range of threats, such as processing enormous amounts of data, spending a lot of energy, and coping with data leakage. As a result, the need for adequate security solutions is growing [21]. Intrusion detection systems, intrusion prevention systems, context-aware security systems, firewalls, and other systems have been used to combat these threats. Context-aware security systems can adequately manage security mechanisms associated with constant context changes [22].

2.1. Background

A contemporary research area is context-awareness in the IoT for threat intelligence. Context is any information that may be used to comprehend the situation of the underlying environment in which the application is running [23]. Location, time, identity, environment, network, history, and activity are the major contexts used for the development of any context-aware system [24]. When adopting and accessing context-aware systems, many variables contribute to consumers developing substantial privacy concerns and potential security risks. As a result, the need for adequate security solutions is growing [9]. Context-aware security systems can adequately manage security mechanisms associated with constant context changes [25]. In recent research, context-aware security systems were introduced in smart grids, smart cities, smart industries, smart health care, smart home, and smart transportation systems to provide security against a variety of assaults, including data loss, phishing, service disruption (DoS/DDoS), power losses, unsecured ports, and other issues, utilizing various approaches such as machine learning, anomaly detection, and artificial intelligence. The term "cyber threat" describes the potential for a successful cyberattack with the intent of gaining unauthorized access to, destroying, disrupting, or stealing a computer network, proprietary information, or any other type of personal information. Cyber threats may originate from a company's own trusted employees or may come from distant, unidentified parties [26]. There are numerous detective strategies available to cope with cyber threats. These techniques can be roughly divided into host-based and network-based categories.

2.1.1. Host and network-based detection technique

To identify potential threats at the system level, the host-based detection technique gathers and analyses information about the internal operations of the computing device, such as log files, register data, API calling patterns, etc. [27]. The host-based detection software, as opposed to a network-based approach, provides the advantage of detailed insight because detection software is installed and running on the host. Numerous host-based tools and approaches have been created to detect bots, worms, and other hazardous threats [28]. Another method of identifying malicious software that might dodge the effects of packers, polymorphism, and deformation technologies is a dynamic behavioral analysis based on the API hook. The protected host's performance is impacted by host-based methods, which also provide incompatible network filtering signatures [29]. The network-based technique is a surveillance strategy that seeks to locate cybersecurity risks by monitoring network traffic for specific network portions or devices, analyzing the network, and using protocol activity to identify odd behavior [30]. The two primary categories of network-based approaches are those that rely on signatures and those that analyze network traffic [29].

2.1.2. Signature and traffic-based threat detection techniques

An attack or intrusion may be recognized by a signature-based IDS if the attack's fingerprint is currently stored in the prevailing database. These methods are often used in the field because they can reliably identify known assaults [31]. To identify cyber dangers, data transmission can be observed and analyzed. The two types of surveillance are active surveillance and passive surveillance [32]. Active monitoring detects activity from malicious traffic. This technique actively injects packets into the network or employs network scanners [33]. For instance, Nmap is a well-known tool that actively collects location and remote information from the Internet. It can do this automatically or manually and includes the domains and servers of this malicious software. The biggest drawback of this approach is how easy it is to discover a network scanner's IP address.

One of the primary categories of passive monitoring is the anomaly-based threat detection technique, which is based on passively observing network traffic [34]. Based on network traffic anomalies, such as excessive network latency, large traffic volumes, traffic on unusual ports, unexpected DNS responses, and traffic behavior characteristics, anomaly-based detection tries to discover malicious code [35]. Three essential steps typically comprise this strategy: First off, some malicious software can be easily captured because it is used in a controlled setting (like a particular honeypot). Second, network security defense software analyses the network traffic generated by the malicious software, and both its static and dynamic characteristics are modeled using mathematical or statistical tools. Finally, models are used to carry out this identification process using data mining and machine learning approaches [28].

2.1.3. Machine learning techniques

In intrusion detection and prevention systems, malicious traffic can be detected using machine learning (ML) techniques [36]. A component of artificial intelligence (AI), machine learning (ML) aims to utilize algorithms to learn from data and produce predictions using that data. Deep learning is computationally expensive due to the massive amount of training data required, as well as sophisticated hardware and software [37]. The functioning of devices and the processes must be checked regularly in industry 4.0 settings in order to spot or anticipate errors or other circumstances that could lead to unfavorable outcomes. Machine learning algorithms can be developed on existing facts to comprehend the hidden aspect of integrated applications and then evaluated to forecast their new state provided by the data. Network threat analysis, which is the process of identifying dangers to the network, is one of the many uses of ML in the field of cybersecurity [38]. Due to its capability to track both incoming and outgoing traffic and identify possibly suspicious activity, machine learning can be useful in this endeavor [39].]. ML tasks can be carried out using a variety of ML models, each of which uses its mathematical equations to analyze the provided data. Different machine learning (ML) methods, such as K-nearest neighbor, XG boost, decision tree, and random forest, are frequently employed for threat detection. The K-nearest neighbor supervised learning model is one of the most basic ML models that are currently available. KNN is referred to as a lazy learner because it does not require training and instead makes predictions about the data to categorize it using the training data [40]. The supervised learning algorithm Decision Tree is beneficial for displaying a model's visual representation. A Decision Tree employs a hierarchical architecture with multiple connected nodes, much like a flowchart [41]. This network contains evaluations of the dataset's features, and each one has a split that either goes to another node or makes a classification judgment for the data. The predicted data is processed via the nodes of the tree that was constructed using the training data until the data can be categorized [42].

The Random Forest-supervised learning algorithm is thought to perform better than the DT model. The randomness of the model comes from two core concepts [43]. The first is that when the model is being trained, each tree is given a random sampling of the data, which can result in some trees using the same data more than once [44]. The goal is to narrow the gap between the scores for the expected outcomes by lowering the model's variance. The second concept suggests utilizing a sparse subset of the features to separate the nodes in the trees [45].

The XG Boost structure of the GBRT model is an enhancement. The sequential ensemble method is used to create the boosting model known as GBRT from a series of simple regression trees [46]. More trees can be adaptively added to the model to expand its capacity. Decision trees are used by XG and RF as classifiers in an ensemble [47]. Examples include using regularization, a technique for working with sparse and balanced data, and adopting a block structure for parallel learning [48].

2.2. Critical analysis

The smart home is among the contender's well-known applications on the Internet of Things dominant paradigm, with approximately 27 billion IoT devices in 2017, and this chain is intended to increase rapidly by 12% per year until it reaches more than 13 trillion devices by 2030. It also presents a significant security problem of escalating threats from anonymous sources that devastate customer experience [49]. Many researchers are working on this challenge and have modeled workable security systems for smart homes to overcome it and highlight the limitations of home automation in terms of detecting and responding to cunning attackers.

In [50], an approach for detecting intrusions in general, with a low false positive rate, is presented for Smart Home Systems (SHS). The authors dynamically modeled the SHS's variegated information into contextual arrays of location and time features for behavior analysis, which is centered on one-class learning to detect various forms of abnormal behaviors. A web interface was used to keep track of the daily usage of household equipment to collect 8110 normal records throughout the course of 90 days. Premised on the baseline model for authentic detection, they revealed that a context sequence of a length with 3 attributes (time slot, gateway availability, and physical location) yields a 2.1 percent false positive ratio. More than 98 percent detection accuracy was achieved for normal usage, involving various related behaviors, but for DoS, asset manipulation and break in attacks involving more than two behaviors, they obtained a detection ratio of more than 94 percent. In terms of future study, researchers suggested that for attacks involving only one behavior, the proposed technique should take detection accuracy into account and make it possible to operate properly for resources that don't require any context information to operate and can conduct complex state changes.

The researchers in [51] monitored the configurations of sensors and devices in a smart home for various user behaviors and utilization of patterns and constructed a context-specific model to distinguish between harmful and benign behavior. They consider context awareness as an understanding of variation in the statuses of sensors and devices as a result of continuing user behavior. The framework uses the machine learning technique of Markov Chain-based to analyze suspected malware by analyzing the present state of smart home assets and comparing it to previously learned user behavior. They tested Aegis in a variety of smart home environments, including with real-life individuals, genuine SHS devices (such as the Samsung Smart Things platform), and various day-to-day routines. By putting Aegis through its paces against a variety of malicious behaviors like impersonation, false data injection, denial-of-service and triggering a malicious app, they were able to obtain more than 95% accuracy with minimal overhead in a variety of smart home situations, detecting threats regardless of smart home designs, user counts, or imposed user regulations.

The authors of [52] emphasized trustworthy context manufacturers and customers, who should secure confidential material in smart surroundings from disclosure or inspection. They have used the attack scenarios of eavesdropping communications between sensors, system components, and applications. The upgraded Cerberus approach is proposed to provide context suppliers and recipients with authentication while also protecting the integrity of the context-specific data and its convenient movement among gadgets. Confidentiality is provided by a symmetric key cryptographic approach, and the integrity of data is ensured using a hash function based on a digital signature. After authentication, the new context providers and recipients were dynamically inserted by the proposed unique method. The strategy is based and evaluated in a genuine cloud platform with six actual devices, proving efficiency,

Fig. 2. v380 smart Wi-Fi camera.

Fig. 3. TUYA smart door lock.

authentication methods, negligible energy usage, and extensibility of multiple portable devices operating in perfect sync, and overall control of the system's privacy rights. The lowest power usage of around 0.35% is reported even without data transmission and while transferring 207.5 B/s toward the cloud.

Another research [32] focused on network-based tracking methodologies and proposed a fundamental structure HomeSnitch for increasing smart home control and visibility by categorization of IoT device exchange mechanisms based on semantic efficacy. Patterns in correlation datagram unit transfers that indicate application-layer interactions between network nodes are searched by HomeSnitch. The model, which was based on a typical wireless router and employed software-defined networks (SDN) primitives to impose connection restrictions, can characterize device behavior by employing destination and content-agnostic features. Using a Random Forest classifier, the model classified behavior from an independent data set of normal and man-in-the-middle attack pcap traffic with greater than 99 percent accuracy. Through all these initiatives, researchers proved the effectiveness of computer networks in classifying behaviors and imposing control on IoT devices.

Furthermore, by relying on IoT device consumption data, the authors in [53] developed a machine learning method to autonomously train contextual access policies from observed social behaviors in home automation. LoFTI is a federated multi-task learning system that identifies six main categories of attributes to record contextual access privileges and trains tailored context-aware rules from numerous smart homes. By designing a novel data augment strategy to handle the problem of outliers in learning, the model achieved a favorable trade between performance and computational power in distributed learning. The results showed that LoFTI may achieve a few false alarms; the false negative rate dropped by 24.2% while the false positive rate dropped by 49.5% when compared to the explicitly stated solo learning to all methods of learning.

Many researchers proposed a context-aware robust intrusion detection system using publicly available datasets such as NSL-KDD, UNSW-NB15, and AWID [54] having attacks scenarios of DoS, eavesdropping, MITM, false data injection, impersonation and triggering a malicious app but did not implement any preventative measures. For effective detection and categorization of innovative and complex assaults, multiple independent deep reinforcement learning agents should be dispersed over the network, producing a model with higher accuracy and a lower false-positive rate than existing systems [55]. The scarcity of real-time smart cyber-physical systems' datasets challenges the advancement in the optimization of intelligent methods for context-aware threat detection and response.

3. Dataset generation

The dataset generation comprises a few sections. In Section 3.1, the proposed testbed setup is discussed. In 3.2, benign and attack scenarios with tactics, techniques, and procedures (TTP) are explained, while Section 3.3 gives a comparison between the IRIL-SHS dataset and other popular publicly available datasets.

3.1. Testbed setup

A detailed description of the IRIL-SHS dataset, including the architecture design, smart devices, and security protocols used, is provided in this section. For this research study, eighteen IoT and non-IoT devices were connected to a Linksys smart Wi-Fi Router WRT1200AC in a star topology within a real testbed of a smart home. Four of them were IoT devices: a TUYA smart plug, a Things access smart hub, a TUYA smart door lock, and a v380 smart Wi-Fi camera that were controlled by homeowners using their mobile and

Fig. 4. Thingz access smart hub.

Fig. 5. TUYA smart plug.

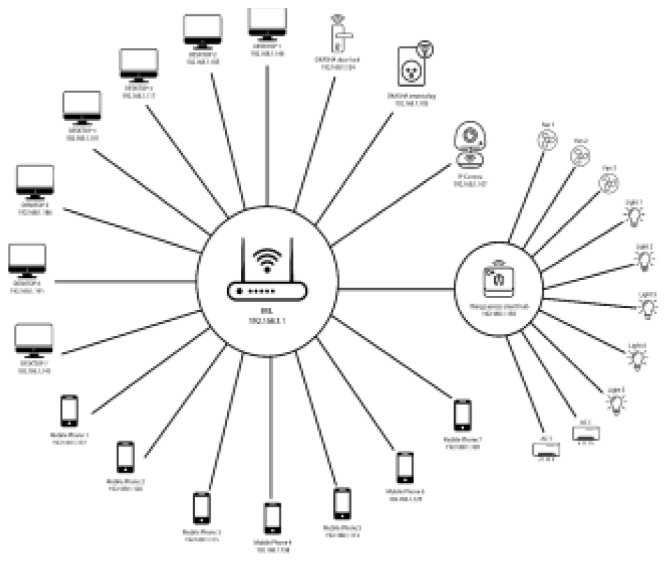


Fig. 6. Network topology.

Table 1
Devices deployed in the smart home.

Device Type	Devices	Protocol	Placement
DESKTOP	7	Ethernet	Close to the Router
Android Mobile	7	802.11 (Wi-Fi)	Wandering across the House
Smart Hub	1	Ethernet	Close to the Router
Smart Switch	1	802.11 (Wi-Fi)	Living Room
Smart Lock	1	802.11 (Wi-Fi)	Entrance Door
IP Camera	1	802.11 (Wi-Fi)	Entrance Door

Table 2
Malicious activities description.

Malicious activity	Tools	Duration	Pcap files
DoS/DDoS	HOIC	3 Days	8
Reconnaissance	Nmap	1 Day	2

web applications. These IoT devices are shown in Figs. 2–5. Seven desktop personal computers and mobile phones were connected to the router working in normal routine by the home users. The Thingz access smart hub was connected to and controlled the fans (3), lights (5), and ACs (2) in smart home architecture. The topology of connected devices in the smart home is shown in Fig. 6.

Table 1 shows the connected device types in the network topology with their protocols and placement. A brand-new collection of network traffic statistics from internet of things (IoT) devices is titled the IRIL-SHS dataset. The data were collected over six days in a medium-sized smart home setup, with 10 attacks captured pcap files and 16 benign captured pcap files. In the IoT Research and Innovation Lab (IRIL) at the Al-Khwarizmi Institute of Computer Science (KICS), University of Engineering and Technology, Lahore [56], this IoT network flow of smart home systems was recorded. Its objective was to provide researchers with a sizable dataset made up of real-time, labeled IoT attacks and IoT benign traffic for training and testing machine learning models. In particular, the residents of the smart homes behaved normally throughout the first two days of the collecting period, and the associated traffic records showed the system to be in a normal state. Three distinct anomalous behavioral scenarios of DoS, DDoS, and reconnaissance attacks, as explained in Sections 3.2.2 and 3.2.3, were put into action over the course of the experiment's last four days to imitate device failure or malicious attacker activity.

Researchers carried out a specific attack sample utilizing several tools that made use of various protocols and carried out various operations in each harmful situation. In a controlled network environment with an unrestricted internet connection, just like any other actual IoT device, both harmful and benign situations were tested.

3.2. *Benign and attack scenarios*

We had twenty-six scenarios for this dataset. The pcap files were recorded for normal and attack traffic scenarios. We had 16 scenarios for normal traffic captured at random timing of different days and 10 scenarios for attacks captured at random timing of different days. All devices in a network used different protocols. The dataset contains TCP, DNS, HTTP, TLS, QUIC, MDNS, UDP, and ARP protocols. DDoS attacks were captured from inside as well as outside the network performing HTTP and TCP flooding.

3.2.1. *Tactics, Techniques, and Procedures (TTP)*

DoS (Denial of Service), DDoS (Distributed Denial of Service), and reconnaissance attacks of three different sorts were carried out against the network. DoS and DDoS attacks were performed within the network as well as outside the network using the HOIC tool [57]. The reconnaissance was performed using the Nmap tool [58]. All the data were captured by the Wireshark tool and saved into pcap files. Table 2 shows the malicious activities performed.

3.2.2. *Denial and Distributed of Service (DoS/ DDoS) scenario*

Any fake flooding endeavor to compromise IoT/ IIoT network and service infrastructure is referred to as a DoS [59]. On several IoT and non-IoT devices in the setup SHS, attackers, including an IP address of 10.177.1.108, performed HTTP flooding and TCP flooding from both inside and outside the network for arbitrary lengths of time. As the smart home system connected with a router having IP 192.168.1.1, a DoS attack was performed from outside the router by port forwarding. The targeted systems were made the victim of many DDoS attacks. Network and IoT devices that were infected with malware turned into a zombie or bots [60]. The attacker then gained remote control of a network of automated devices known as a botnet [61]. Attackers including IP 10.177.1.108, 10.10.38.224, and 192.168.1.145 performed HTTP flooding and TCP flooding on different IoT and non-IoT devices in the SHS from outside the network as well as within the network at random duration.

3.2.3. *Reconnaissance scenario*

An attack known as a scanning assault, sometimes referred to as reconnaissance or probing, is the first step in the cyber death chain model or penetration testing [62]. This attack's goal was to gather victim systems' data, including system vulnerabilities and current IP addresses. On a variety of IoT and non-IoT devices in the SHS from outside the network, an attacker with IP 192.168.1.159 executed a Computer OS Fingerprint Probe, Network/Port scan, TCP Null scan, TCP SYNFIN scan, and TCP Xmas scan for a random amount of time.

3.2.4. *Validation of the IRIL-SHS dataset*

To validate the IRIL-SHS dataset developed in this study, it was evaluated with random forest, decision tree and K-nearest neighbor classification models with 12 selected network features. The same technique was utilized on the publicly available dataset like UNSW-NB15 and KDD99. The motive of adopting these datasets was that (i) they do not contain record duplication while collecting modern network traffic, and (ii) they resolve the issue of imbalances between normal and attack observations. The UNSW-NB15 and KDD99 datasets both contain normal and malicious network traffic activity and have 2,57,673 and 4,66,530 records in total, respectively. Each record is distinguished by 49 attributes, including class labels. The collected 12 attributes, which are common in IRIL-SHS, UNSW-

Table 3
Comparison of classification model results with publicly available dataset.

	Random forest			Decision tree			K-Nearest neighbour		
Dataset	Accuracy	**Precision**	**F1- Score**	Accuracy	**Precision**	**F1- Score**	Accuracy	**Precision**	**F1- Score**
IRIL-SHS	0.964	0.96	0.96	0.9239	0.93	0.92	0.9548	0.96	0.95
UNSW-NB15	0.953	0.95	0.95	0.9445	0.94	0.94	0.952	0.96	0.95
KDD99	0.955	0.95	0.95	0.952	0.95	0.95	0.945	0.95	0.95

Table 4
Comparison of popular datasets & IRIL-SHS dataset.

DS	RNC	RNT	HDS	FPC	ONA	MMS
KDD99	True	False	False	True	False	False
TUIDS ISCX	True	True	False	True	False	True
UNSW-NB15	True	True	False	True	False	False
CICIDS2017	True	True	False	True	True	True
IRIL-SHS	True	True	True	True	True	True

Dataset: DS; Realistic Network Configuration: RNC; Realistic Network Traffic RNT; Heterogeneous Data Sources: HDS; Full Packet Captures: FPC; Outside the Network Attack: ONA; Many Malicious Scenarios: MMS.

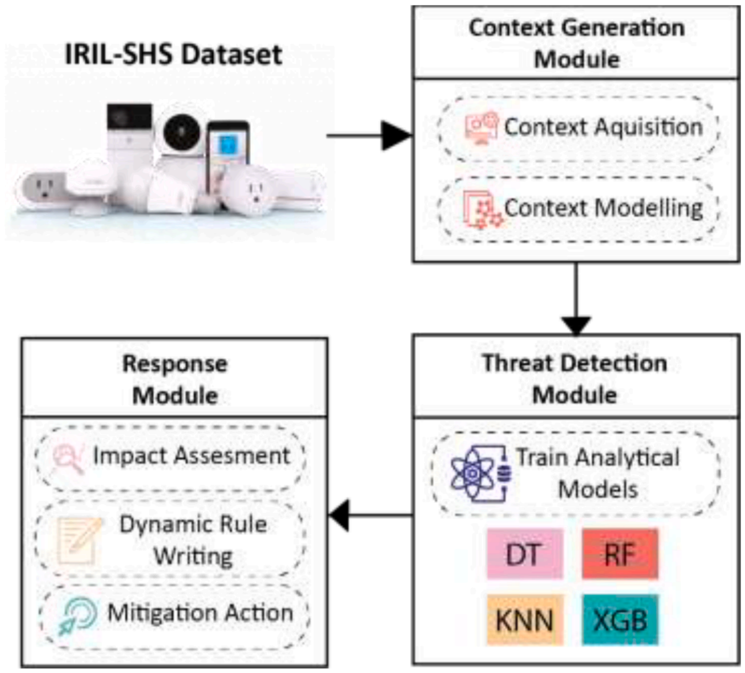


Fig. 7. Context-aware threat detection and response model.

NB15 and KDD99 datasets, were passed to the classification model after data was divided into a 70:30 ratio, with 70% utilized for training and 30% afterwards used for testing. The classification techniques are implemented using the Google collaborative environment in Python language. Table 3 depicts the key difference in the output of classification model between the IRIL-SHS dataset and publicly available datasets to provide usefulness and correctness of the IRIL-SHS dataset.

According to the results in Table 3, random forest model performed the best on all performance metrics, indicating that it is the most effective model for predicting attack traffic. When compared to baseline dataset performance, the IRIL-SHS dataset outperforms, demonstrating the dataset's usefulness and correctness for addressing the challenge at hand.

3.3. Comparison with popular security datasets

The design and development of credible datasets remain a key area of research to develop accurate intrusion detection methods able to identify and prevent threats/ attacks. Existing datasets, while useful in some contexts, present a number of problems including lack of consistently labelled data, lack of attack variety such as DDoS attack from outside the network, redundancy of traffic, and the

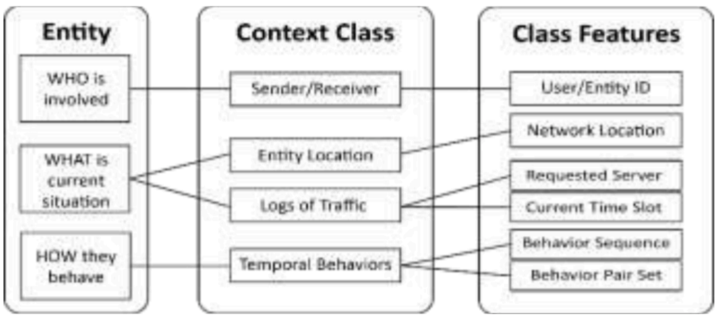


Fig. 8. SHS's context mapping hierarchical structure.

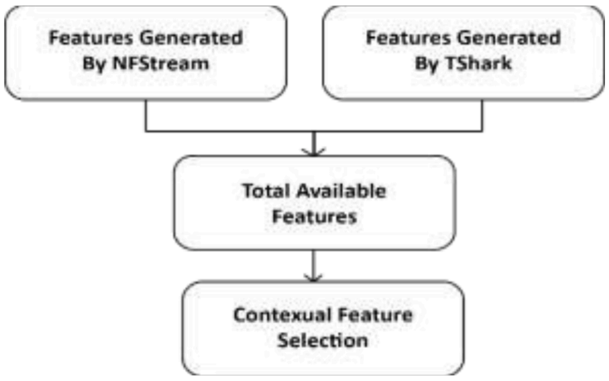

Features Generated By NFStream
Features Generated By TShark
Total Available Features
Contexual Feature Selection

Fig. 9. Flow diagram of contextual feature generation module.

absence of ground truth [63]. This research study has carefully compared the IRIL-SHS dataset with other popular datasets namely KDD99 [64], TUIDS ISCX [65,66], UNSW-NB15, and CICIDS2017 [67]. These datasets have been extensively used in many research studies to provide security solutions. The analysis showed that the KDD99 dataset is now obsolete and does not reflect current network traffic. The TUIDS ISCX, UNSW-NB15, and CICIDS2017 [67] datasets were collected from actual situations and contain comprehensive packet capture with a range of security events. However, they lack heterogeneous data sources and traffic (normal and attack) from outside the network in consideration. In contrast, the IRIL-SHS dataset provides heterogeneous data sources (Table 1) with many malicious activities (Table 2) from both internal and external networks. The generated dataset captures authentic network traffic from actual networks of IoT devices created by router configuration and consists of heterogeneous data sources gathered from IoT device telemetry datasets, Windows and Linux-based datasets, and network traffic datasets. Table 4 depicts the key differences between the IRIL-SHS dataset and other common datasets to provide a fair comparison.

4. Context-aware threat detection and response model

The IRIL-SHS dataset gathered from the communication of various IoT devices in the proposed testbed setup is used in our approach. Our context-aware threat detection and response model consists of three major modules as illustrated below in Fig. 7.

The contextual feature generation module contains the network and flow features extracted from the pcap to CSV files using NFStream and Tshark. After data preprocessing, we performed the feature selection to get the high-performance features. In the threat detection module, we used machine learning algorithms to validate the dataset and detect the threats accurately. The response module performed the dynamic rule writing against the detected attacked traffic.

4.1. Contextual features generation module

The notion of context modeling, which is to represent the data obtained from the IoT devices and resources that may explicitly explain the system's behavior, is the foundation for the construction of the contextual feature-generating module [50]. By tracing the key information into a high-level context that is highly specific and accurate, the diversity of IoT device information may be characterized. The context mapping for the smart home system's hierarchical structure is shown in Fig. 8 below.

The topmost hierarchy uses pairs in the following order to define the structure for the context [68]:

Who: is involved?
What: is current the situation?

Table 5
Extracted features using TShark.

Feature	Data type	Description
tcp.window_size	Integer	TCP flow Window Size
tcp.flags	String	TCP flags (SYN, FIN, URG, and RST)
tcp.len	Integer	TCP packet length
tcp.seq	Integer	TCP packet sequence
tcp.ack	Integer	TCP packet acknowledged
tcp.stream	Integer	TCP packet stream
ip.ttl	Integer	Time to live for a packet
http-request	Integer	Number of HTTP requests
udp.port	Integer	UDP port number

How: they behave?

Several context classes are formed in the second tier of SHS's context mapping to execute grouping methods on the IoT data. The classes created using grouping techniques, such as the sender context class, are set up to accept the input of the following data: 1) the IP address, authentication data, and HTTP cookies of the sender, all of which are taken from the packet stream. 2) account info for individuals, obtained from the configuration log. 3) The sender's precise location, as determined by a GPS signal [50]. It is essential to note that if a device's position is maliciously changed, like an IP camera captures traffic from a restricted database or a server room, its location context can be utilized in determining the threat or attack scenario. Furthermore, a trucking company that transports items throughout the country with a fleet of vehicles. Each vehicle is fitted with a GPS device and a telematics system that tracks the vehicle's location, speed, and other critical data. This contextual data can be utilized to determine the malicious scenario, such as sudden changes in the vehicle's route, unexpected stops or detours, or unusual driving behavior. The information from a real-world situation allows human-oriented decision-making by utilizing a number of machine-learning approaches such as feature extraction, learning, and inference. Finally, all class features containing network flow information are stored in a low-level hierarchy. Behavior sequence and behavior set are formed by the pair set of information captured in the traffic related to each other like Close Door number 1 D1C, Turn off Light number 1 L1O etc. If length of behavior sequence is 3 then we can get different pair set as {"D1C", "L2O", "D3O"}. Fig. 9 shows the flow diagram for the contextual feature generation module. Fig. 9 shows the flow diagram for the contextual feature generation module. This research used NFStream [65] and Tshark [66] to construct the features for the IRIL-SHS dataset.

All features obtained from NFStream and Tshark were then combined to get complete possible information about the proposed context-aware smart home system. To get high-performance contextual features, a feature selection technique, explained in Section 4.1.2, was utilized by consuming only relevant data, getting rid of the noise in the data, and passing them to the model. To make live and unconnected network data more comprehensible, NFStream, an open-source Python API library, allows simple and customizable feature conversion [67]. The library should act as a common network packet analytics platform for academics, enabling data repeatability between studies, according to the authors' main goal. The following advantages are provided by NFStream:

1 Extraction of statistical features:

Regarding feature engineering, NFStream provides both early flow features and postmortem statistical features (such as the minimum, mean, standard deviation, and maximum packet size and inter-arrival time) as well as a sequence of the first n packet sizes, inter-arrival times, and directions [67].

1 Flexibility:

Extending NFStream is simple. The work is open-sourced, and the feature selection technique can be done via NFPlugins.

Based on the configurations defined by NFStreamer (a driver process), new flow characteristics were developed. The driver's main responsibility is to set the entire workflow, which is mostly an arrangement of concurrent metering tasks. The features that were extracted using NFStreamer are listed in Appendix A, Table A1.

From the network pcap files, we retrieved 86 flow features/attributes using NFStream and stored them as a CSV file. Tshark is a function of Wireshark that is used to monitor network protocols and analyze traffic. Although Wireshark has limited export abilities, it can examine pcap files that record network data [63]. Besides static feature extraction, TShark offers a more versatile, potent export capability that can produce analytical, calculated data. We used the TShark discussions export option to retrieve several fundamental, traffic- and connection-based features [68]. We also used Tshark for extracting TCP, HTTP, and UDP information. The following command on the Kali Linux extracted TCP, HTTP, and UDP features.

[tshark -r input.pcap.pcapng -T fields -e tcp.window_size -e tcp.flags -e tcp.len -e tcp.seq -e tcp.stream -e tcp.ack -e ip.ttl -e http.request -e udp.port -E header=y -E separator=, > output.csv]

The features that were retrieved using TShark are listed in Table 5.

A total of 10 features, extracted by Tshark, and all files of these features were merged with the NFStreamer files having 86 features of different data types containing a total of 96 features set. These features were good enough for extracting context-aware features to provide a heterogeneous security system and model for SHS. All CSV files were labeled manually and merged into one file using a

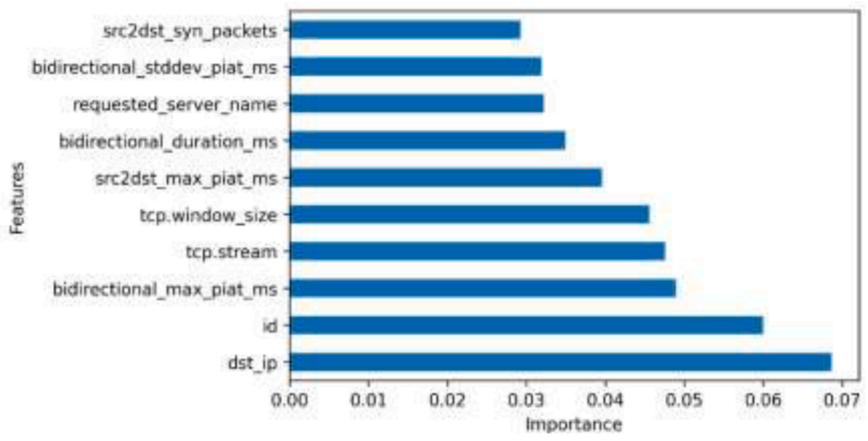


Fig. 10. Comparison of feature importance.

Table 6
Contextual features importance of IRIL-SHS dataset.

Selected features	Importance
id	0.060025
dst_ip	0.069980
tcp.window_size	0.048890
tcp.stream	0.047596
bidirectional_max_piat_ms	0.050000
src2dst_max_piat_ms	0.042230
src2dst_syn_packets	0.030000
bidirectional_duration_ms	0.037870
requested_server_name	0.031120
bidirectional_stddev_piat_ms	0.031120

python script. It is important to highlight that each record, whether normal or attack, was tagged using an authentic tagging procedure. Manual tagging/labeling was done by stamping 0 to Normal traffic, 1 to DoS Attack traffic, 2 to DDoS attack, and 3 to Reconnaissance attack traffic.

4.1.1. *Preprocessing of labeled dataset*

The IRIL-SHS dataset contains 608,500 records of real-time traffic, including both attacks and normal events, with far more normal records than anomalous records. To avoid overfitting and examine the generalization capacity of each model, all redundant records were eliminated from the dataset to reduce the imbalance impact. To address the imbalance issue in the dataset, an experiment was conducted where the performance of the proposed model was compared using the original dataset and a balanced dataset by utilizing oversampling and under-sampling techniques. Specifically, the synthetic minority over-sampling technique (SMOTE) was used for oversampling and random under-sampling (RUS) for under-sampling. For oversampling, SMOTE was applied to the minority class in the dataset to create synthetic samples, resulting in a balanced dataset. For under-sampling, some samples were randomly removed from the majority class in the dataset to balance the dataset. The model was then trained using both the original and the balanced datasets and evaluated the performance using the same metrics.

The experiment showed that the performance of the model improved significantly when using the balanced dataset compared to the original dataset. Notably, the accuracy improved from 88% to 99%, and the F1-score improved from 0.7 to 0.85. These results indicate that dataset imbalance can have a significant impact on the performance of the model and that addressing this issue using over-sampling and under-sampling techniques can improve the overall performance. Converting categorical data into numerical information and removing any anomalies and incorrect data from the dataset are the first steps in the data pre-processing process. As a result, we encoded the categorical features using an ordinal encoder, thereby increasing the feature count.

4.1.2. *Contextual features selection*

Throughout this step, we selected a set of attributes that yielded the greatest performance. By lowering the number of features and removing unwanted or loud characteristics, feature selection speeds up training [69]. The Extra Tree classifier approach, an ensemble learning feature selection methodology also known as the Extremely Randomized Trees Classifier, is employed in this study as feature selection [70]. It aggregates the results from various pattern decision trees from a forest to display the outcomes of its classification. Random samples from the training dataset are used to construct each decision tree in the Extra Trees Forest. Then, a random number of K-featured samples are distributed to each decision tree test node. Using the GINI index or Information Gain, also known as feature importance, each decision tree selects the best features to differentiate between meaningful and irrelevant aspects of the data. This forest layout's features are presented in declining order of feature importance [71]. Each feature from this forest layout is arranged in descending order of feature relevance. The top K features are then chosen from this feature order, with the other features being ignored [72]. The following formula can be used to determine the entropy of a feature:

$$Entropy(F) = \sum_{i=1}^{n} -p_i log_2(p_i) \tag{1}$$

Where n is the number of distinct class labels and p_i is the probability that a class i exists in the dataset. In this study, the top 10 dataset features were chosen using information gain, shown in Fig. 10.

According to the figure, 10 features were chosen for the IRIL-SHS datasets, which represents an 86% reduction in the size of the feature set for the entire dataset. This results from the Extra Tree classifier strategy selecting the pertinent features like packet TCP stream and window size values that indicate the propensity of cyberattacks to have distinct packet dimensions when contrasted to usual traffic, offering the most details about the class [73]. Table 6 illustrates the 10 contextual features that were chosen for all attack types with their importance.

Table 6 lists the top 10 features that have an impact on our model. Identification, Destination IP, TCP window size, TCP stream, source to destination and bidirectional Inter-packet maximum time, Syn packets from source to destination, duration in both directions, Bidirectional standard deviation packet inter-arrival time and the requested server name which is most useful for indicating DoS, DDoS and Reconnaissance traffic classification.

4.2. Threat detection module

The second module of our research trains the various machine learning models that are utilized to identify malicious behaviors in SHS using contextual characteristics provided in the contextual features generation module as input. Many learning approaches are used to identify cybersecurity threats, including decision trees, random forests, naive Bayes, support vector machines, K-nearest neighbors, deep belief networks, artificial neural networks, and XG-Boost [73]. Decision Trees, Random Forests, K-Nearest Neighbors, and XG Boost are the four techniques that have been considered. Each tactic is briefly discussed below. This approach aims to assess the effectiveness and performance of different machine-learning approaches against distinct attack types. The data was divided into a 70:30 ratio, with 70% utilized for training and 30% afterward used for testing. The classification techniques are implemented using the Google collaborative environment in Python language. The libraries employed in this work include sklearn, pandas, matplotlib, and NumPy.

4.2.1. Decision tree classifier

The decision tree functions by splitting the data points into a representation of (D), and each split is carried out in a way that maximizes the related features while minimizing informational entropy (E). The splits are known as leaves (L), and the terminal leaf is the last split [38]. We chose a minimum sample split of two, a maximum depth of ten, and a random state of zero for our decision tree classifier. Consider G to be the sample split metric that needs to be maximized. R is the possible range of values. Let ∂ be a user-defined confidence value state and let n be the total number of training samples that are available. The classifier may be calculated as follows:

$$\sum^{=}\sqrt{\frac{R^2 ln\left(\frac{1}{\partial}\right)}{2^{*}n}} \tag{2}$$

4.2.2. Random forest classifier

The output of several machine learning algorithms is combined in ensemble learning, which improves predicted performance and makes use of additional machine learning algorithms. As a result, the random forest method incorporates many decision tree methods. Implementation of random forest involves the following steps:

Step1: Choose B arbitrary data points from the practice set.
Step 2: Building the decision tree connected to these B data points.
Step 3: Decide how many trees to construct and repeat steps 1 and 2.
Step 4: Forecast the category for each branch of the decision tree for a new data point, and then predict the category that received the majority of votes [39]. Class prediction of the random forest includes:

$$\mathrm{C}_{rf}^{B} = majorityvote\{\mathrm{C}_b(x)\}_1^B \tag{3}$$

4.2.3. K nearest neighbor classifier

The KNN algorithm determines how to assign a new data point (D) to one of the categories by using the subsequent steps. Step 1: Decide how many neighbors there will be; we chose $K = 7$. Step 2: Using the Euclidean distance, get the K-nearest neighbors of the new data point. Step 3: Count the number of data points in each category among these K neighbors. The fourth step is to assign the new data

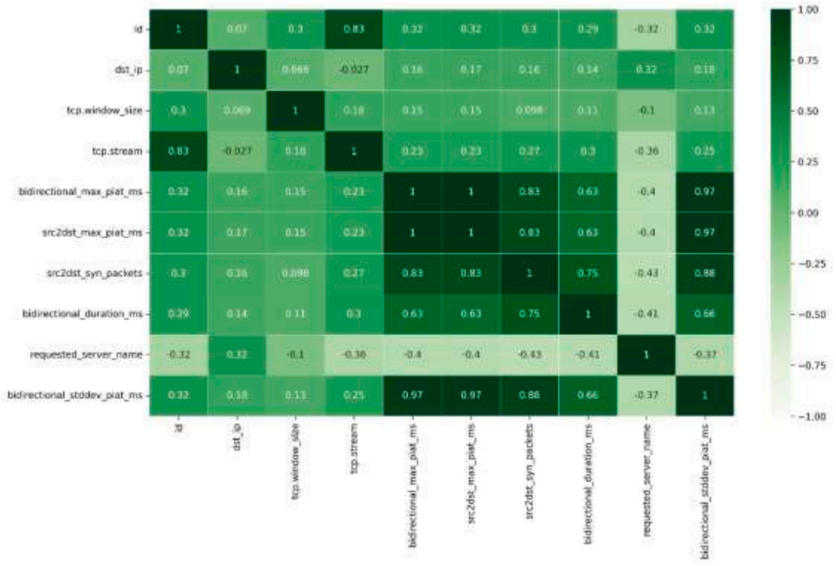


Fig. 11. Pearson correlation of contextual features.

point to the category with the greatest number of neighbors [36]. Euclidean distance (d) between sample $x_i$and x_l ($l = 1, 2, 3, \ldots$, n) is defined as

$$d(x_i, x_l) = \sqrt{(x_{i1}, x_{l1})^2 + (x_{i2}, x_{l2})^2 + \ldots + (x_{in}, x_{\mathrm{ln}})^2} \tag{4}$$

4.2.4. XG boost classifier

Accuracy problems, data loss, and resulting discrepancies can all be managed via XG Boost. The following steps make up the XG Boost process. Step 1: A decision tree is initially created. Step 2: The majority decision is then examined in a Bagging operation to produce predictions. Step 3: Trees are independent; hence random forest is used to form forests. Step 4: Boosting is carried out to find losses (L). Step 5: Gradient boosting is used to overfit a dataset. Step 6: XG Boost offers parallel tree boosting (GBDT, GBM) [43].

After applying all classifiers, we got the Random Forest classifier classifying the attack traffic and normal traffic 99.177211% accurately. We got the forecast model that could distinguish between various sorts of traffic thanks to the machine learning analysis. We could now carry out several add-on tasks for a model trained to distinguish between good and bad traffic e.g.:

- Based on the outcomes of the random forest, create extra firewall rules to block offending traffic.
- Create IDS rules based on the outcomes of the random forest to identify malicious traffic.
- Review logs or traffic captures regularly, reporting any high-priority malicious traffic that is found [74].

4.3. Response module

The final section of our study offers a straightforward set of rules for finding correlations in the information collected through our Random Forest model. By integrating these rules into already-in-use defenses like IDS, firewalls, specially-written detection scripts, or classification software, malicious traffic or data can be prevented from entering our SHS [74].

For rules, we used the WEKA tool which is a collection of machine-learning techniques for data mine. Obtained data modeling, categorization, grouping, mining, and presentation techniques are all included. Weka is simple to use and can be expanded using Java programming [75].

JRip is indeed a WEKA version of RIPPER, a classifier method that tends to make use of its categories as they increase in size before generating basic rules set for a subclass. Every decision event in the learning dataset is treated by JRip as a class, and it then generates a set of rules that apply to group individuals of that class. Once all classes have already been completed, it moves on to the next one and performs necessary operations [76]. This research work used JRip due to its prominence and widespread use in earlier studies for rule set generation [77,78]. To differentiate between DoS, DDoS, Reconnaissance, and normal traffic, we obtained a total of 9 rules. Simple rules in the form of a tree outlined the metrics values that may be used to identify the sort of attack in the traffic. These rules aided in establishing the precise values to be included in the Suricata rules while creating an SHS prevention system.

5. Results and discussion

Based on contextual features, the categorization model forecasts whether the incoming traffic will be harmful or benign. As the

Table 7
Metrics definitions.

Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$

Table 8
Confusion matrix comparison of threat detection module.

Decision tree classifier Actual/Predicted	Normal	DoS	DDoS	Reconnaissance
Normal	17,787	223	47	81
DoS	480	71,044	239	5
DDoS	617	1055	85,308	3
Reconnaissance	486	67	21	5087
Random Forest Classifier				
Actual/Predicted	**Normal**	**DoS**	**DDoS**	**Reconnaissance**
Normal	17,968	86	52	32
DoS	306	71,340	107	15
DDoS	471	108	86,385	19
Reconnaissance	259	25	22	5355
K Nearest Neighbor Classifier				
Actual/Predicted	**Normal**	**DoS**	**DDoS**	**Reconnaissance**
Normal	17,362	247	195	334
DoS	436	71,071	252	9
DDoS	662	206	86,105	10
Reconnaissance	461	26	39	5135
XG Boost Classifier				
Actual/Predicted	**Normal**	**DoS**	**DDoS**	**Reconnaissance**
Normal	18,040	12	12	74
DoS	601	70,958	196	13
DDoS	760	763	85,453	7
Reconnaissance	755	6	10	4890

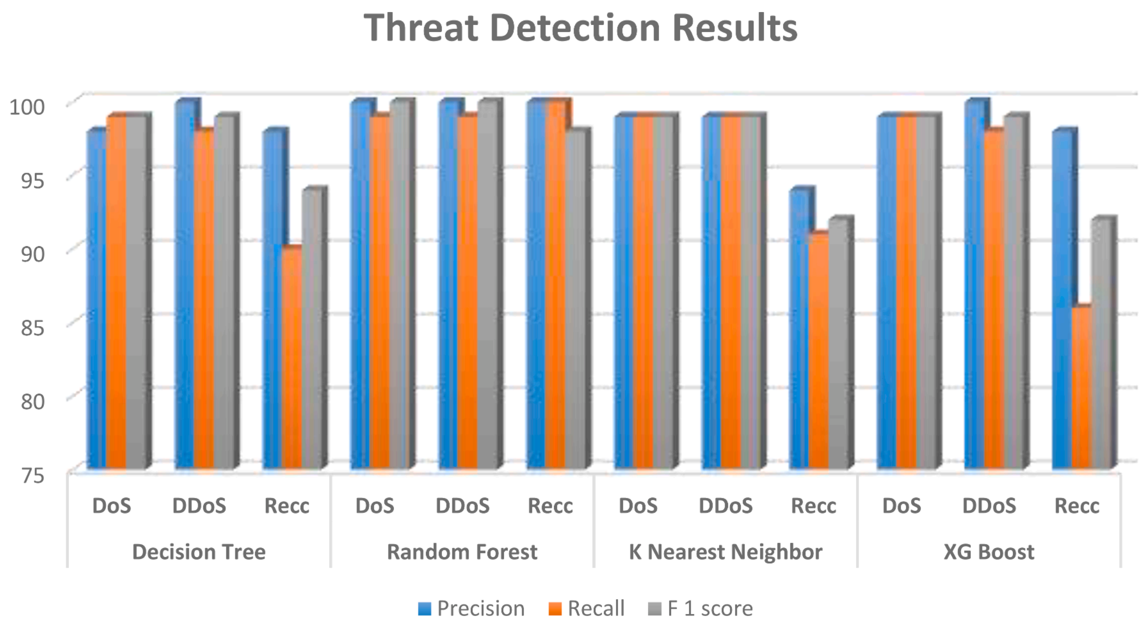


Fig. 12. Threat detection model comparison.

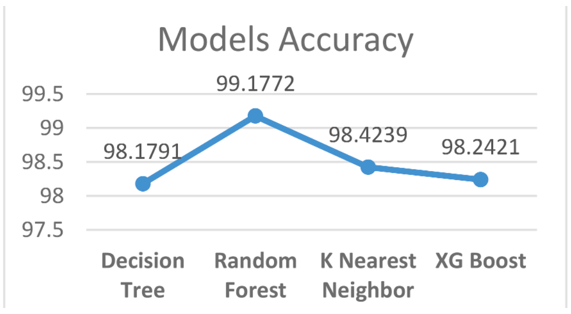


Fig. 13. Comparison of the different machine learning algorithms.

Table 9
Comparison of classification rates with existing approach.

Paper	Dataset	Feature selection	Random forest model results
[69]	UNSW NB15	Feature importance technique using RF	97.37% accuracy
IRIL-SHS Dataset	IRIL-SHS	Feature importance technique using RF	97.99% accuracy
Proposed Context-aware Threat Detection Model	IRIL-SHS	GINI index or Information Gain using DT	99.1772% accuracy

proposed model predicts the type of attacks and determines the correlation between the 10 variables. The chosen attributes showed a high correlation with a value near to or equal to −1 or 1. Fig. 11 shows the Pearson correlation matrix of the selected features for our dataset.

The evaluation of performance is typically an important factor in machine learning. Numerous indicators have been considered when evaluating the model used in this research [79]. To measure the model's performance in the multi-class classification, we employed metrics for precision, recall, F1 score, and accuracy. Precision is the number of positive predictions divided by the total number of positive class values predicted. Recall is the number of positive predictions divided by the number of positive class values in the test data. F1 score is the weighted average of precision and recall while accuracy is defined as the number of correct predictions, including true positive (TP) and true negative (TN) predictions, divided by the total number of predictions. A low recall could signify a high proportion of false negative (FN) predictions, whereas a low accuracy could signify a high proportion of false positive (FP) predictions. A high F1 score could suggest low FP and low FN predictions because it combines precision and recall [80].

True Positive (TP) is the number of anomaly traffic that has been identified after an attack. True Negative (TN) is the number of detectable normal traffic that is considered to be benign traffic (normal). False Positive (FP) is the number of anomaly traffic (attacks) that have been identified as benign traffic (normal). False Negative (FN) is the quantity of detectable benign traffic that is identified as attack traffic (attack). Table 7 provides definitions of the measures in terms of positives and negatives.

This research study represented multi-class classification using the confusion matrix of all models applied to the dataset. Table 8 shows the confusion matrix comparison of the threat detection machine learning module.

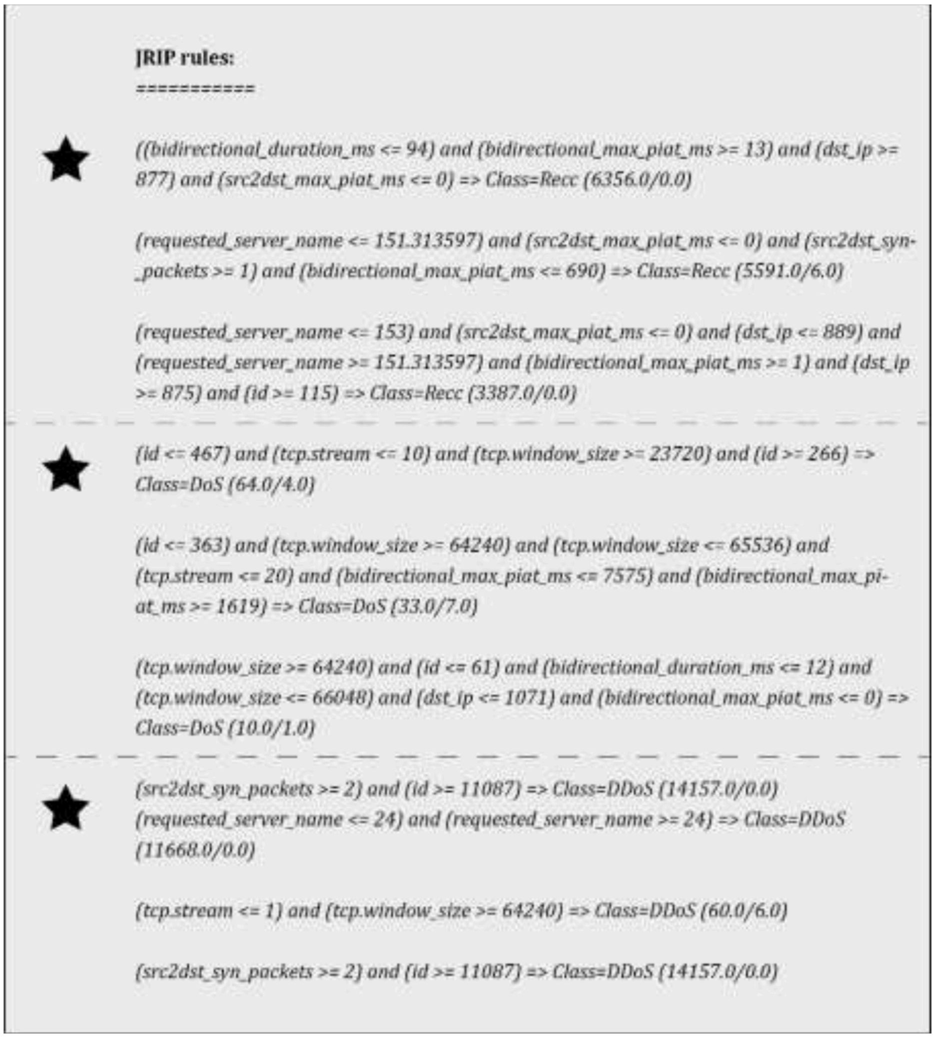
The classification of legitimate packets and malicious packets from three different attack groups is accomplished using the random forest model with high accuracy of 99.177%. The model can detect 99% of the genuine regular packets in the usual stream. The model can detect more than 98% of malicious packets as detection of the relevant type for DoS, DDoS, and reconnaissance attacks, but it cannot discriminate packets of DDoS attacks, as 10.9% of DDoS traffic was classified as normal traffic,) because of sufficient training packets for DDoS attacks. So, enhancing the number of packets will help detect attacks accurately. Precision, Recall, and F1 Score comparison of our threat detection module is shown in Fig. 12.

It is evident from the trials and outcomes above that there is a trade-off between being able to identify destructive behaviors and upholding a low percentage of false alarms (FPR). Most of the time, Random Forest obtained the lowest false positive rates, which explained its successful F1-score performance. The comparison of the accuracy we achieved in our models is shown in Fig. 13.

When compared to these machine learning techniques, Random Forest's performance exhibited a notable improvement. The best KNN reached 98.4239% accuracy, whereas the Decision Tree classifier and XG Boost obtained 98.1791% and 98.2421% accuracy, respectively. The Random Forest model, however, has the highest accuracy of 99.1772%.

5.1. Comparison with the previous study

The most pertinent research in the field of machine learning is described in [69], where the authors used the UNSW NB15 data set to apply supervised machine learning (ML) methods such Random Forest (RF), Support Vector Machine, and Artificial Neural Networks to the clusters. A dataset for networking attacks is UNSW-NB15 and there are 9 distinct assaults in it, including malware, open ports, DoS attacks, and packet sniffing. Crude datagrams are included in the collection. The training set has 175,341 data, whereas the test set contains 82,332 data of the two main types, attacked and ordinary. They used 37 features and divided them into clusters based on the MQTT and TCP protocol types. Using the random forest classifier, they were able to classify several classes with 97.37% accuracy. We



```
JRIP rules:
===========

((bidirectional_duration_ms <= 94) and (bidirectional_max_piat_ms >= 13) and (dst_ip >=
877) and (src2dst_max_piat_ms <= 0) => Class=Recc (6356.0/0.0)

(requested_server_name <= 151.313597) and (src2dst_max_piat_ms <= 0) and (src2dst_syn-
_packets >= 1) and (bidirectional_max_piat_ms <= 690) => Class=Recc (5591.0/6.0)

(requested_server_name <= 153) and (src2dst_max_piat_ms <= 0) and (dst_ip <= 889) and
(requested_server_name >= 151.313597) and (bidirectional_max_piat_ms >= 1) and (dst_ip
>= 875) and (id >= 115) => Class=Recc (3387.0/0.0)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(id <= 467) and (tcp.stream <= 10) and (tcp.window_size >= 23720) and (id >= 266) =>
Class=DoS (64.0/4.0)

(id <= 363) and (tcp.window_size >= 64240) and (tcp.window_size <= 65536) and
(tcp.stream <= 20) and (bidirectional_max_piat_ms <= 7575) and (bidirectional_max_pi-
at_ms >= 1619) => Class=DoS (33.0/7.0)

(tcp.window_size >= 64240) and (id <= 61) and (bidirectional_duration_ms <= 12) and
(tcp.window_size <= 66048) and (dst_ip <= 1071) and (bidirectional_max_piat_ms <= 0) =>
Class=DoS (10.0/1.0)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(src2dst_syn_packets >= 2) and (id >= 11087) => Class=DDoS (14157.0/0.0)
(requested_server_name <= 24) and (requested_server_name >= 24) => Class=DDoS
(11668.0/0.0)

(tcp.stream <= 1) and (tcp.window_size >= 64240) => Class=DDoS (60.0/6.0)

(src2dst_syn_packets >= 2) and (id >= 11087) => Class=DDoS (14157.0/0.0)
```

Fig. 14. JRip rules of IRIL-SHS dataset.

have applied their techniques of data preprocessing and feature selection on the IRIL-SHS dataset to validate and compare their results with the proposed context-aware threat detection model. Following Table 9 describe the comparison of context-aware and simple feature selection for Random Forest Model.

By applying the developed IRIL-SHS dataset to the straightforward machine learning model, as in the prior study without integrating context aware features, 97.99% accuracy was obtained against the random forest model. However, the proposed context-aware threat detection model was improved by achieving 99.1772% accuracy.

For the response module, nine rules were gained using JRip classifier for the developed dataset. The best feature set was obtained with defined values against each feature using the JRip classifier [70], which is beneficial for generating Suricata signatures. For instance, if the bidirectional duration is less than or equal to 94 and the bidirectional maximum packet inter-arrival time is greater than or equal to 13, and the destination IP is 877, and the source to destination maximum packet inter-arrival time is less than or equal to 0, the traffic is reconnaissance and should be blocked from entering the smart home system.

Using these metrics, we made some Suricata emerging threats rules for our prevention system as shown in Fig. 14. The IRIL-SHS dataset is based on the same benchmark datasets, KDD99, TUIDS ISCX, UNSW-NB15, and CICIDS2017. It offers a variety of data sources and traffic, including a lot of malicious actions from both internal and external networks. The conclusion section discusses the stages of Random Forest model deployments on the IRIL-SHS dataset and how contextual feature generation and transfer learning improve threat detection accuracy to 99.177%. Based on the generated Random Forest model, the conversion rules will be developed in our response module to automatically generate Suricata signatures.

6. Conclusion and future work

For diverse varieties of online threats, various learning models are applied. In contrast, limited researchers have sought to draw

Table A1
Extracted features using NFStreamer.

NFStream Features	Data types	Explanation	NFStream features	Data types	Explanation
id	Data	Flow indicator	expiration_id	Data	Flow expiration trigger identifier.
src_ip	String	String representation of the source IP address.	src_mac	String	String representation of source MAC address.
src_oui	String	String representation of the source organization unique identifier.	src_port	Integer	Port used for the transport layer.
dst_ip	String	String representation of the destination IP address.	dst_mac	String	String representation of destination MAC address.
dst_oui	String	String of destination organization unique identifier.	dst_port	Integer	Destination Port used for the transport layer.
protocol	Integer	Protocol for the transport layer.	ip_version	Integer	IP address version
vlan_id	Integer	Identifier for a virtual LAN.	biD_first_seen_ms	Integer	Millisecond timestamp on the first bidirectional flow packet.
biD_last_seen_ms	Integer	Millisecond timestamp on the last bidirectional flow packet.	biD_duration_ms	Integer	Milliseconds of flow in both directions.
biD_packets	Integer	Flow accumulator for bidirectional packets.	biD_bytes	Integer	Bi-directional bytes accumulator flow.
s2d_first_seen_ms	Integer	Timestamp on the initial flow src2dst packet in milliseconds.	s2d_last_seen_ms	Integer	Timestamp on the last flow src2dst packet in milliseconds
s2d_duration_ms	Integer	Duration of the flow src2dst in milliseconds.	s2d_packets	Integer	Src2dst flow packet accumulation.
s2d_ bytes	Integer	Flow src2dst bytes accumulator.	d2s_first_seen_ms	Integer	Millisecond timestamp on the initial flow dst2src packet.
d2s_last_seen_ms	Integer	Millisecond timestamp on the last flow dst2src packet.	d2s_duration_ms	Integer	Milliseconds of flow between dst2src.
d2s_packets	Integer	Accumulator for dst2src packet flow.	d2s_bytes	Integer	Flow dst2src bytes accumulator.
app_name	String	Application name found by nDPI.	app_category_name	String	App class name found by nDPI.
app_is_guessed	Integer	Set if detect result relied on a port-based guess or just on dissection.	req_server_name	String	Requested server name (SSL/TLS, DNS, HTTP).
client_fingerprint	String	Client fingerprint (DHCP fingerprint for DHCP, JA3 for SSL/TLS and HASSH for SSH).	server_fingerprint	String	User Agent Identifier for QUIC or extracted user agent for HTTP.
content_type	String	HTTP content type extracted.	biD_min_ps	Integer	Flow bidirectional min packet size.
biD_mean_ps	Float	Flow bidirectional mean packet size.	biD_stdev_ps	Float	Flow bidirectional packet size sample standard deviation.
biD_max_ps	Integer	Flow bidirectional max packet size.	s2d_min_ps	Integer	Flow src2dst min packet size.
s2d_mean_ps	Float	Flow src2dst mean packet size.	s2d_stdev_ps	Float	Flow src2dst packet size sample standard deviation.
s2d_max_ps	Integer	Flow src2dst max packet size.	d2s_min_ps	Integer	Flow dst2src min packet size.
d2s_mean_ps	Float	Flow dst2src mean packet size.	d2s_stdev_ps	Float	Flow dst2src packet size sample standard deviation.
d2s_max_ps	Integer	Flow dst2src max packet size.	biD_min_piat_ms	Integer	Flow bidirectional min packet inter arrival time.
biD_mean_piat_ms	Float	Flow bidirectional mean packet inter arrival time.	s2d_stdev_piat_ms	Float	Flow src2dst packet inter arrival time standard deviation.
biD_stdev_piat_ms	Float	Flow bidirectional packet inter arrival time sample standard deviation.	s2d_max_piat_ms	Integer	Flow src2dst maximum packet inter arrival time.
biD_max_piat_ms	Integer	Flow bidirectional maximum packet inter arrival time.	d2s_min_piat_ms	Integer	Flow dst2src minimum packet inter arrival time.
s2d_min_piat_ms	Integer	Flow src2dst minimum packet inter arrival time.	d2s_mean_piat_ms	Float	Flow dst2src mean packet inter arrival time
s2d_mean_piat_ms	Float	Flow src2dst mean packet inter arrival time.	d2s_stdev_piat_ms	Float	Flow dst2src packet inter arrival time standard deviation.
d2s_max_piat_ms	Integer	Flow dst2src maximum packet inter arrival time.	s2d_cwr_packets	Integer	Flow src2dst cwr packet accumulators.
biD_syn_packets	Integer	Flow bidirectional syn packet accumulators.	s2d_ece_packets	Integer	Flow src2dst ece packet accumulators.
biD_cwr_packets	Integer	Flow bidirectional cwr packet accumulators.	s2d_urg_packets	Integer	Flow src2dst urg packet accumulators.
biD_ece_packets	Integer	Flow bidirectional ece packet accumulators.	s2d_ack_packets	Integer	Flow src2dst ack packet accumulators.
biD_urg_packets	Integer	Flow bidirectional urg packet accumulators.	s2d_psh_packets	Integer	Flow src2dst psh packet accumulators.
biD_ack_packets	Integer	Flow bidirectional ack packet accumulators	s2d_rst_packets	Integer	Flow src2dst rst packet accumulators.
biD_psh_packets	Integer	Flow bidirectional psh packet accumulators	s2d_fin_	Integer	Packets Flow src2dst fin packet accumulators.
biD_fin_packets	Integer	Flow bidirectional fin packet accumulators.	d2s_syn_packets	Integer	Flow dst2src syn packet accumulators.
s2d_syn_packets	Integer	Flow src2dst syn packet accumulators.	d2s_cwr_packets	Integer	Flow dst2src cwr packet accumulators.
d2s_urg_packets	Integer	Flow dst2src urg packet accumulators.	d2s_ece_packets	Integer	Flow dst2src ece packet accumulators.
d2s_ack_packets	Integer	Flow dst2src ack packet.	d2s_rst_packets	Integer	Flow dst2src rst packet.
d2s_psh_packets	Integer	Flow dst2src psh packet.	d2s_fin_packets	Integer	Flow dst2src fin packet.

attention to the limitations that machine learning techniques encounter. To test the most recent developments in machine learning for cyber-attacks, we observed and advised that an inclusive benchmark dataset which would not only include data from the heterogeneous smart IoT devices but also attack data from internal as well as external network should be created. This dataset was then used for context-aware feature generation in the random forest model, which conclusively showed that the accuracy of threat identification on the IRIL-SHS dataset was increased. When context-aware results are considered, the system obtained a high rate of detection and level of precision. Out of the four ML algorithms, RF performed better than the others by consistently achieving excellent detection performance and F1-score with low false positive rates. To prevent malicious traffic from accessing the smart home system, this study suggested rule-set generation from the open-source tool, Suricata. To add emerging threat rules into Suricata for system defense, JRip rules helped to get the metrics values. This research study acknowledges the limitations of the performance of the developed machine learning model against a limited set of attack types. While the researchers believe that the results demonstrate the robustness of the model against the attacks that were tested, they also recognize that there may be other types of attacks that were not included in the intensive experiments. Additionally, as they rely on 4-tuple information, consisting of source and destination IP addresses and ports, the model may not be able to extract additional features like payload or application layer information from the encrypted traffic. Further research is needed to fully evaluate the performance of the proposed model in real-world scenarios where a variety of attack types may be present along with advancements in encryption and decryption technologies to improve the ability to extract features from encrypted traffic for critical analysis. Based on the real-time IRIL-SHS dataset using IoT devices, future studies can also concentrate on generating the intended Industry 4.0 dataset which could include data from industrial IoT devices like distributed control systems (DCS), programmable logic control (PLCs) and gateways with attack data like Modbus protocols attacks etc., and from an encrypted traffic the feature extraction would be a unique task. As the presented model performs well for smart homes and small offices, it can also perform effectively for industry 4.0 dataset. In future, based on various threat scenarios and zero-day attacks, we also intend to enhance our contextually aware model for the industry 4.0 dataset.

Declaration of Competing Interest

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

Data availability

Data will be made available on request.

Appendix A

Table A1.

References

[1] N. Sharma, M. Shamkuwar, I. Singh, The history, present and future with IoT. Internet of Things and Big Data Analytics for Smart Generation, Springer, 2019, pp. 27–51.

[2] M. Khan, S. Din, S. Jabbar, M. Gohar, H. Ghayvat, S. Mukhopadhyay, Context-aware low power intelligent SmartHome based on the Internet of Things, Communist Chin. Sci. Abstr. 52 (2016) 208–222.

[3] S.M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of Internet of Things (IoT): a survey, J. Netw. Comput. Appl. 161 (2020), 102630.

[4] S. Zheng, N. Apthorpe, M. Chetty, N. Feamster, User perceptions of smart home IoT privacy, Proc. ACM Hum. Comput. Interact. 2 (2018) 1–20.

[5] P. Gupta, A. Sinha, A. Perti, A.K. Singh, Security implementations in IoT using digital signature. Innovations in Electrical and Electronic Engineering, Springer, 2021, pp. 523–535.

[6] A.D. TUB, R.K. UM, B. Schmid, M.S. TUB, and F. Fahy, "Deliverable 2.4.".

[7] S. Gollagi, M. Math, A.A. Daptardar, A survey on pervasive computing over context-aware system, CCF Trans. Pervasive Comput. Interact. 2 (2) (2020) 79–85.

[8] H.J.T. Manaligod, M.J.S. Diño, S. Ghose, J. Han, Context computing for Internet of Things, J. Ambient Intell. Humaniz. Comput.: Springer 11 (2020) 1361–1363.

[9] B. Ospan, N. Khan, J. Augusto, M. Quinde, K. Nurgaliyev, Context aware virtual assistant with case-based conflict resolution in multi-user smart home environment, in: 2018 International Conference on Computing and Network Communications (coconet), IEEE, 2018, pp. 36–44.

[10] D.W. Seo, H. Kim, J.S. Kim, J.Y. Lee, Hybrid reality-based user experience and evaluation of a context-aware smart home, Comput. Ind. 76 (2016) 11–23.

[11] E. de Matos, et al., Context information sharing for the Internet of Things: a survey, Comput. Networks Chem. Lab., Symp. 166 (2020), 106988.

[12] L. Tan, K. Yu, F. Ming, X. Cheng, G. Srivastava, Secure and resilient artificial intelligence of things: a HoneyNet approach for threat detection and situational awareness, IEEE Consum. Electron. Mag. 11 (3) (2021) 69–78.

[13] A. Tundis, M. Uzair, M. Mühlhäuser, An IoT-based context-aware model for danger situations detection, Communist Chin. Sci. Abstr. 96 (2021), 107571.

[14] Z. Pan, S. Hariri, J. Pacheco, Context aware intrusion detection for building automation systems, Comput. Security 85 (2019) 181–201.

[15] E.A. Shams, A. Rizaner, A.H. Ulusoy, A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems, Neural Comput. Appl. 33 (20) (2021) 13647–13665.

[16] J. Indumathi, N. Asha, J. Gitanjali, Smart security system using IoT and mobile assistance. Emerging Research in Data Engineering Systems and Computer Communications, Springer, 2020, pp. 441–453.

[17] N.H. Sultan, V. Varadharajan, L. Zhou, F.A. Barbhuiya, A role-based encryption (RBE) scheme for securing outsourced cloud data in a multi-organization context, IEEE Trans. Serv. Comput. (2022).

[18] A. Chorti, et al., Context-aware security for 6 G wireless: the role of physical layer security, IEEE Commun. Standards Mag. 6 (1) (2022) 102–108.

[19] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, Y. Venu Madhav, A context-aware robust intrusion detection system: a reinforcement learning-based approach, Int. J. Inf. Secur. 19 (6) (2020) 657–678.

[20] S. van Engelenburg, M. Janssen, B. Klievink, Designing context-aware systems: a method for understanding and analysing context in practice, J. Log. Algebr. Methods Program. 103 (2019) 79–104.

[21] S.T. Park, G. Li, J.C. Hong, A study on smart factory-based ambient intelligence context-aware intrusion detection system using machine learning, J Ambient Intell. Humaniz. Comput. 11 (4) (2020) 1405–1412.
[22] T. Sylla, M.A. Chalouf, F. Krief, K. Samaké, Towards a context-aware security and privacy as a service in the Internet of Things, in: IFIP International Conference on Information Security Theory and Practice, Springer, 2019, pp. 240–252.
[23] S. Gollagi, M. Math, A.A.J.C.T. o. P. C. Daptardar, and Interaction, "A survey on pervasive computing over context-aware system," vol. 2, pp. 79–85, 2020.
[24] H.J.T. Manaligod, M.J.S. Diño, S. Ghose, J.J.J.o.A.I. Han, H. Computing, Context Computing For Internet of Things, 11, Springer, 2020, pp. 1361–1363.
[25] D.W. Seo, H. Kim, J.S. Kim, and J.Y.J.C.I.I. Lee, "Hybrid reality-based user experience and evaluation of a context-aware smart home," vol. 76, pp. 11–23, 2016.
[26] M. Lehto, Cyber-attacks against critical infrastructure. Cyber Security, Springer, 2022, pp. 3–42.
[27] P. Deshpande, S.C. Sharma, S.K. Peddoju, S. Junaid, HIDS: a host based intrusion detection system for cloud computing environment, Int. J. Syst. Assurance Eng. Manag. 9 (3) (2018) 567–576.
[28] S. Yang, J. Wang, J. Zhang, H. Li, Cyber threat detection and application analysis, in: 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, 2016, pp. 46–49.
[29] A. Kumar, M.A. Albreem, M. Gupta, M.H. Alsharif, S. Kim, Future 5 G network based smart hospitals: hybrid detection technique for latency improvement, IEEE Access 8 (2020) 153240–153249.
[30] M. Masdari and H.J.A.S.C. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," vol. 92, p. 106301, 2020.
[31] Y. Zhu, Y. Zheng, Retracted article: traffic identification and traffic analysis based on support vector machine, Neural Comput. Appl. 32 (7) (2020) 1903–1911.
[32] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, A.R. Sadeghi, HomeSnitch: behavior transparency and control for smart home IoT devices, in: Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, 2019, pp. 128–138.
[33] R.R. Chowdhury, S. Aneja, N. Aneja, E. Abas, Network traffic analysis based IoT device identification, in: Proceedings of the 2020 the 4th International Conference on Big Data and Internet of Things, 2020, pp. 79–89.
[34] S. Boudabous, J. Garbiso, B. Leroy, S. Clémençon, H. Labiod, Traffic analysis based on bluetooth passive scanning, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, 2019, pp. 1–6.
[35] M. Husák, M. Čermák, T. Jirsík, P. Čeleda, HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting, EURASIP J. Inf. Secur. 2016 (1) (2016) 1–14.
[36] J.D. Ndibwile, A. Govardhan, K. Okada, Y. Kadobayashi, Web Server protection against application layer DDoS attacks using machine learning and traffic authentication, in: 2015 IEEE 39th Annual Computer Software and Applications Conference 3, IEEE, 2015, pp. 261–267.
[37] G. Kocher and G.J.S.C. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," vol. 25, no. 15, pp. 9731–9763, 2021.
[38] J. Hegde, B. Rokseth, Applications of machine learning methods for engineering risk assessment–a review, Stem Cells Int. 122 (2020), 104492.
[39] A. Churcher, et al., An experimental analysis of attack classification using machine learning in IoT networks, Sensors 21 (2) (2021) 446.
[40] Z. Zhang, Introduction to machine learning: k-nearest neighbors, Ann. Transl. Med. 4 (11) (2016).
[41] A. Merghadi, et al., Machine learning methods for landslide susceptibility studies: a comparative overview of algorithm performance, Earth Sci. Rev. 207 (2020), 103225.
[42] H. Sharma, S. Kumar, A survey on decision tree algorithms of classification in data mining, Int. J. Sci. 5 (4) (2016) 2094–2097.
[43] M. Schonlau, R.Y. Zou, The random forest algorithm for statistical learning, Stata J. 20 (1) (2020) 3–29.
[44] Y. Mishina, R. Murata, Y. Yamauchi, T. Yamashita, H. Fujiyoshi, Boosted random forest, IEICE Trans. Inf. Syst. 98 (9) (2015) 1630–1636.
[45] Q. Ren, H. Cheng, H. Han, Research on machine learning framework based on random forest algorithm, in: AIP Conference Proceedings 1820, AIP Publishing LLC, 2017, 080020.
[46] D. Elavarasan, D.R. Vincent, Reinforced XGBoost machine learning model for sustainable intelligent agrarian applications, J. Intellig. Fuzzy Syst. 39 (5) (2020) 7605–7620.
[47] A. Gupta, S. Sharma, S. Goyal, M. Rashid, Novel xgboost tuned machine learning model for software bug prediction, in: 2020 International Conference on Intelligent Engineering and Management (ICIEM), IEEE, 2020, pp. 376–380.
[48] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[49] N. Ghosh, S. Chandra, V. Sachidananda, Y. Elovici, SoftAuthZ: a context-aware, behavior-based authorization framework for home IoT, IEEE Internet Things J. 6 (6) (2019) 10773–10785.
[50] Z. Pan, J. Pacheco, S. Hariri, Y. Chen, B. Liu, Context aware anomaly behavior analysis for smart home systems, Int. J. Inf. Commun. Eng. 13 (5) (2019) 257–270.
[51] A.K. Sikder, L. Babun, H. Aksu, A.S. Uluagac, Aegis: a context-aware security framework for smart home systems, in: Proceedings of the 35th Annual Computer Security Applications Conference, 2019, pp. 28–41.
[52] J. Al-Muhtadi, K. Saleem, S. Al-Rabiaah, M. Imran, A. Gawanmeh, J.J. Rodrigues, A lightweight cyber security framework with context-awareness for pervasive computing environments, Sustain. Cities Soc. 66 (2021), 102610.
[53] T. Yu, et al., Learning context-aware policies from multiple smart homes via federated multi-task learning, in: 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), IEEE, 2020, pp. 104–115.
[54] Y.F. Hsu, M. Matsuoka, A deep reinforcement learning approach for anomaly network intrusion detection system, in: 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), IEEE, 2020, pp. 1–6.
[55] M. Lopez-Martin, A. Sanchez-Esguevillas, J.I. Arribas, B. Carro, Network intrusion detection based on extended RBF neural network with offline reinforcement learning, EEE Access 9 (2021) 153153–153170.
[56] (2002). Al-Khwarizmi Institute of Computer Science (KICS). Available: https://kics.edu.pk/web/.
[57] HOIC Tool. Available: https://www.imperva.com/learn/ddos/high-orbit-ion-cannon/.
[58] "Nmap Tool.".
[59] J.M. Vidal, A.L.S. Orozco, L.J.G. Villalba, Adaptive artificial immune networks for mitigating DoS flooding attacks, Swarm Evol. Comput. 38 (2018) 94–108.
[60] A. Furfaro, P. Pace, A. Parise, Facing DDoS bandwidth flooding attacks, in: Simulation Modelling Practice Theory, 98, 2020, 101984.
[61] Y. Lu, M. Wang, An easy defense mechanism against botnet-based DDoS flooding attack originated in SDN environment using sFlow, in: Proceedings of the 11th International Conference on Future Internet Technologies, 2016, pp. 14–20.
[62] T. Zaware, "Cybersecurity automation using cyber kill chain.".
[63] N. Koroniotis, N. Moustafa, E. Sitnikova, and B.J.F.G.C.S. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: bot-iot dataset," vol. 100, pp. 779–796, 2019.
[64] G. Meena, R.R. Choudhary, A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA, in: 2017 International Conference on Computer, Communications and Electronics (Comptelix), IEEE, 2017, pp. 553–558.
[65] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of network-based intrusion detection data sets, Comput. Security 86 (2019) 147–167.
[66] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, Inf. Security J.: Global Perspect. 25 (1–3) (2016) 18–31.
[67] R. Panigrahi, S. Borah, A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems, Int. J. Eng. Technol. 7 (3) (2018) 479–482, 24.
[68] H.T. Nguyen, Q.D. Ngo, V.H. Le, A novel graph-based approach for IoT botnet detection, Int. J. Inf. Secur. 19 (5) (2020) 567–577.
[69] M. Ahmad et al., "Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set," vol. 2021, no. 1, pp. 1–23, 2021.
[70] P. Nimbalkar and D.J.I.E. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," vol. 7, no. 2, pp. 177–181, 2021.

[71] M. Injadat, et al., Multi-stage optimized machine learning framework for network intrusion detection, IEEE Trans. Netw. Service Manag. 18 (2) (2020) 1803–1816.
[72] M.R.C. Acosta, et al., Extremely randomized trees-based scheme for stealthy cyber-attack detection in smart grid networks, IEEE access 8 (2020) 19921–19933.
[73] M. Ahsan, et al., Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review, Journal of Cybersecurity Privacy 2 (3) (2022) 527–555.
[74] J. Markey, A. Atlasis. Using decision tree analysis for intrusion detection: a how-to guide, SANS Institute, 2011.
[75] A. Basharat, M.M.B. Mohamad, A. Khan. Machine Learning Techniques for Intrusion Detection in Smart Healthcare Systems: A Comparative Analysis, IEEE, 2022.
[76] P.A. Barraclough, G. Fehringer, J. Woodward, Intelligent cyber-phishing detection for online, Comput. Secur. 104 (2021) 102123.
[77] S. Kumar, A. Viinikainen, T. Hamalainen. Evaluation of ensemble machine learning methods in mobile threat detection, IEEE, 2017.
[78] R.S. Abdullah, et al., Analysis of IoT Botnets using Machine Learning Technique 4 (1) (2022) 18–30.
[79] B.S. Khater, et al., Classifier performance evaluation for lightweight IDS using fog computing in IoT security, Electronics 10 (14) (2021) 1633.
[80] M. Ge, et al.. Deep learning-based intrusion detection for IoT networks, IEEE, 2019.