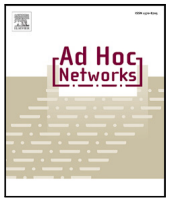
Ad Hoc Networks 156 (2024) 103404

Contents lists available at ScienceDirect

Ad Hoc Networks

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



SecureFlow: Knowledge and data-driven ensemble for intrusion detection and dynamic rule configuration in software-defined IoT environment

Amritpal Singh [a], Pushpinder Kaur Chouhan [b], Gagangeet Singh Aujla [c,*]

[a] Department of Engineering, Durham University, UK
[b] British Telecom, UK
[c] Department of Computer Science, Durham University, UK

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Intrusion detection system
Software-defined networking (SDN)
Cyber threats

ABSTRACT

There is a massive growth in the rate of heterogeneous devices configured in the Internet of Things (IoT) environment for efficient communication. The IoT devices are limited in resources, and there are no defined protocols in terms of security during communication in the IoT-based platforms. Several solutions are framed to make communication secure in the IoT ecosystem. However, the existing schemes need to be more reliable to handle the cyber threats and unwarranted incidents (such as intrusions, anomalies and attacks) coming from IoT endpoints owing to the unstructured patterns of IoT data and dynamic network conditions. Moreover, heavy cryptographic primitives have their deployment challenges due to the resource constraints of the IoT ecosystem. The dynamic nature of IoT traffic requires flexible and varied rules to handle the threats in different deployment scenarios. Therefore, a programmable interface enabled through Software-defined Networking (SDN) can handle heterogeneous threats and incidents in the IoT cyber world. Thus, in this paper, we have designed a novel framework, SecureFlow, an intrusion detection and dynamic rule configuration system based on the knowledge-based and data-driven ensemble. The proposed framework is robust and fault tolerant owing to dual-layer Intrusion Detection System (IDS) and rule configuration modules that can work without one of them. SecureFlow validated through several experiments performed through emulations in Mininet. The results depict that the proposed framework is effective and promising.

1. Introduction

Internet of Things (IoT) is a network of dedicated physical objects (things) that contain embedded technology to communicate and sense or interact with their internal states or the external environment [1]. Since its inception, IoT has expanded as a revolution, breaking all the barriers and entering the broader era of smart applications not limited to just healthcare or smart cities. The growth in IoT technologies led to expanding the research dynamics in this field. Fig. 1 shows how the research dynamics related to IoT technologies have matured since its inception. IoT encompasses a wide range of smart communities (like smart factories, power systems, etc.) wherein the generated data need to be guaranteed semantic-aware or logic-based or intent-based [2] processing and provisioning. Such applications are anticipated to generate data, send it to the processing systems, receive back the decision or outcome based on the generated data, and circulate it within the environment to materialise a specific goal or anticipated activity. It is anticipated that by 2025, around 75.44 billion things will connected across the globe to form a web of data generated at an unprecedented rate [3]. However, the generated data (i.e., big data) is expected to pose several challenges for the underlying network paradigms. The volume and dynamic patterns of the IoT data make it tedious for conventional network architectures to provide dynamic traffic management policies and broader flexibility.

The low latency requirements, unstructured patterns of IoT data, and dynamic network conditions pose a crucial challenge for providing ultra-reliable services in a connected IoT ecosystem [4]. Any delay or unwarranted incident or attack can result in functional incorrectness alongside failing to achieve the temporal guarantees [5,6]. There are many definitions of unwarranted incidents (or an attack), but we consider an incident as an event (any observable activity) that negatively affects environments/systems and impacts the business. An incident can be planned or unplanned and can cause an interruption to a service or a reduction in the quality of a service. Although there are several solutions to cope with the challenges caused by unwarranted

* Corresponding author.
E-mail addresses: amritpal.singh@durham.ac.uk (A. Singh), pushpinder.kaur.chouhan@bt.com (P.K. Chouhan), gagangeet.s.aujla@durham.ac.uk (G.S. Aujla).

https://doi.org/10.1016/j.adhoc.2024.103404
Received 14 November 2023; Received in revised form 25 December 2023; Accepted 10 January 2024
Available online 20 January 2024

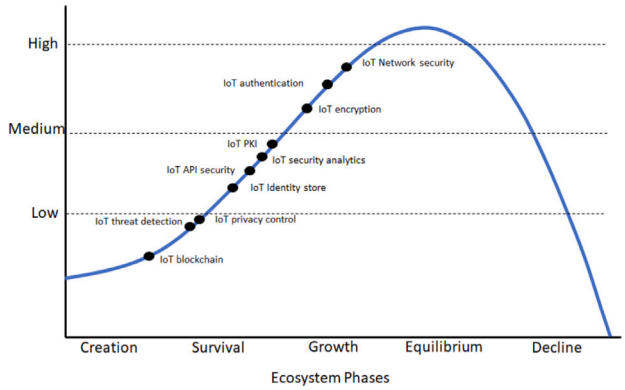


Fig. 1. Ecosystem phases of IoT technologies.

incidents or attacks in connected IoT systems, legitimate and malicious parties can access the open communication medium, the Internet. Thus, securing connected IoT systems is vital to the success of any other performance verticals.

Although IoT requires robust cyber-security primitives, the low latency requirements and limited computational resources make it challenging to rely on hard-coded or conventional cryptographic mechanisms. In line with this requirement, several existing researchers have proposed numerous intrusion detection systems (IDS) to handle unwarranted incidents or attacks in IoT-based environments. Most of these IDSs are categorised as knowledge-based systems or data-driven approaches. Knowledge-based IDS needs a knowledge repository that provides the legitimate network profile, and any activity (or action) deviating from the legitimate network profile is termed as an intrusion [7–9]. For instance, Kreibich and Crowcroft [10] proposed a mechanism to generate the attack signatures for network IDS automatically. The proposed system adopts pattern-matching methods and applies compliance checks at several layers in the honeypot system. Meiners et al. [11] proposed a hardware-based regular expression matching mechanism that uses Ternary Content Addressable Memories for deep packet inspection. However, given the modern-day IoT environment, the proposed systems were not designed to meet their requirements. Garg et al. [12] proposed a probabilistic approach using a set of data structure signatures to design an anomaly detection system for the Internet of Vehicles. However, this system uses signature verification to detect malicious nodes only.

The data-driven IDS use machine learning models to extract knowledge from a large dataset based on functions, rules or policies. These help to understand/extract varied data patterns and predict the malicious or anomalous behaviours in the data traffic [13]. In another work, Verma et al. [14] explored the benefits of integrating machine learning classification algorithms to secure the various attacks in the IoT environment. Eskandari et al. [15] introduced an intelligent IDS named *Passban* to safeguard the IoT devices configured in the smart cities. Similarly, Kumar et al. [16] proposed a novel unified IDS to protect the IoT-based environment from four attacks: exploit, Denial-of-Service, probe, and generic. The authors used the UNSW-NB15 data set to detect malicious activities. In another work, Ferrag et al. [17] introduced a new approach named RDTIDS for IoT based on classifier techniques such as decision tree and rule-based approaches. Singh et al. [18] proposed a deep learning-based approach to filter intrusions in IoT-edge ecosystems. The authors used deep belief networks to improve the accuracy of intrusion filtration. Garg et al. [19] adopted Restricted Boltzmann Machine and Unscented Kalman Filter to understand the profile of normal and abnormal behaviour. Yang et al. [20] proposed a neural network for an IDS that uses fast optimisation speed characteristics.

1.1. Research gaps and motivation

Existing proposals do not consider varied ranges and types of unwarranted incidents (or attacks). Most of the above-discussed approaches focused on small-scale networks and homogeneous traffic patterns. However, the increase in the number of IoT devices and heterogeneity of data patterns increases the chances of attacks or incidents in the network. Some approaches opted for data-driven solutions, while others opted for knowledge-driven methods. They have their benefits, but these systems are limited in their nature and scope, i.e., they focus only on detecting specific attacks or intrusions rather than on responding to them (once they are detected) through dynamic reconfiguration. It may be seen that a majority of the existing proposals do not provide a unified framework that can respond to unwarranted incidents (or attacks), help recover the network to pre-attack performance, and report/record the impact of recovery/mitigation policy or rule for future use. Thus, we need to devise a broader solution that can provide real-time and long-time attack (or intrusion) detection that uses a knowledge base and data capabilities while considering IoT traffic's dynamic patterns and characteristics. Moreover, it must ensure a deep intrusion analysis and root cause diagnosis to decide a remedial policy or a way to restore the performance of the underlying system to normal at the earliest.

Most existing solutions are static and fixed-function and unsuitable for dynamic network paradigms. Conventional networks limit their capabilities due to stringent fixed-function rules and complex reconfiguration capabilities. Even if the solution supports dynamic reconfiguration for responding to unwarranted incidents (or attacks), the conventional network architecture limits its adaption and application. However, Software-defined Networking (SDN) has been widely anticipated as an alternative to the fixed-function conventional network technologies for ensuring dynamic reconfiguration and responses to unwarranted incidents (or attacks) scenarios [21]. SDN provides flexibility and openness to network management by decoupling the control from the forwarding devices. Most of the network management and control tasks are performed at the control plane, making it the network's brain. SDN has the potential to meet the dynamic requirement of processing the data packets generated from the IoT ecosystems [22]. The SDN's programmable characteristics help control the network as per requirement. The SDN provides real-time visibility of the configured network. To handle the unwarranted security incidents and attacks in the IoT environment, Wani et al. [23] proposed SDN-based IDS integrating deep learning approaches to detect the various anomalies. In another work, Ashraf et al. [24] presented an SDN-based anomaly detection system in an IoT-based environment. The authors focused on support vector machine, k-nearest neighbour, and multiplayer perception approaches for detecting anomalous traffic.

1.2. Contributions

For this reason, we have designed a novel system known as **SecureFlow** to handle the concerns related to IoT security by leveraging SDN as a core network architecture. The main contributions of this work are listed below.

- An ensemble IDS is designed that considers SNORT and Support Vector Machines (SVM) working in tandem to provide an alert regarding intrusions and anomalies in IoT traffic.
- An incremental host rating system is designed to forecast the status of malicious hosts in an IoT network.
- An SDN-based dynamic rule configuration (generation, verification and implementation) system is designed to respond to any unwarranted incidents or attack scenarios.
- A rule impact calculator is designed to score the effectiveness of the dynamic rule generated by the SDN controller.

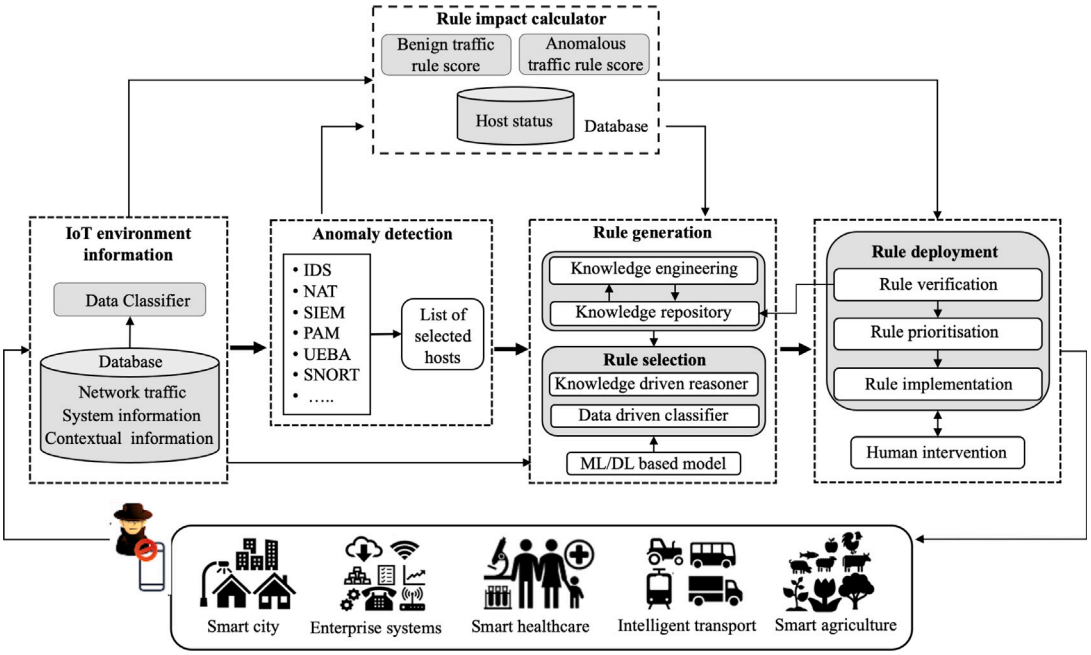


Fig. 2. A SDN-based flow management system.

2. Proposed SecureFlow system architecture

SecureFlow is a two-tier system designed to secure the IoT environment by responding to security incidents via SDN-based adaptive flow rule generation. The proposed approach leverages the benefits of knowledge-driven and data-driven approaches at two levels, i.e. IDS and rule generation levels, to ensure the designed system is robust and tolerant to faults. The critical idea leverages the benefits of both knowledge-driven and data-driven approaches to determine an appropriate response rule for security incidents once detected by IDS. The key advantages and unique features of this two-tier methodology are manifold. However, the top five advantages are listed below.

- We can incorporate domain knowledge in the knowledge-driven models whilst also taking advantage of the dynamic nature and continuous improvement of data-driven models. By not focusing on one specific approach, we can build a system that could take benefits of both approaches simultaneously.
- As two approaches can be used together, but they are not dependent on each other, we can produce results without one approach's needs. For example, in the absence of relevant data, response rules to security incidents can be decided by focusing on a knowledge-driven approach.
- This hybrid approach can resolve both known and unknown incidents. Known incidents can be resolved through a knowledge-driven approach that could implement signature-based, rule-based, and policy-based intrusion response methodologies, and a data-driven approach can resolve unknown incidents.
- We can automatically construct a large-scale knowledge base from the results of the data-driven approach, which can be used for incident response rule generation. A data-driven approach can help shorten the "Knowledge Gap" for new incidents and address the problem of ambiguous knowledge.
- Incidents informed from SDN-enabled environments can be analysed and resolved by a hybrid approach.

Looking into the above benefits, SecureFlow can address various intrusion types in SDN-IoT systems. SecureFlow could respond to two types of incidents: anomalies and attacks. This feature not only resolves the attacked systems but also helps to avoid any future attacks that may cause adverse effects on the system. Fig. 2 presents the architecture of SecureFlow, which consists of the following components:

- **IoT Environment Information:** Here, an organised collection of data related to IoT environment performance and device statistics is done. It consists of two subsystems:
  - **Database:** It stores information the IoT environment provides related to its devices and the network.
  - **Data Classifier:** It differentiates data per the requirements of various system components. For example, data required by a System Verifier differs from data required by a detection engine or rule generator.
- **Anomaly/Intrusion Detection:** This layer performs all the network intrusion and anomaly detection activities and generates the list of infected hosts or hosts with a high probability of being attacked. We can use log, traffic, and content analysis tools developed by researchers and Industry to detect intrusions and anomalies in the IoT environment. These systems include IDSs, Network Address Translation (NAT), Security Information and Event Management (SIEM), Privileged Access Management (PAM), User and Entity Behavior Analytics (UEBA), and Snort. In the proposed work, we have used two two-tier IDS that consider Snort-driven and SVM-driven IDS.
- **Rule Generation**: This component generates the rules for the informed malicious hosts. It takes information (and) or data about incidents and past responses and then analyses to identify relationships among them based on a knowledge-driven reasoner or data-driven classifier. It consists of the following components.
  - **Knowledge Engineering:** It is imitating how a human expert in a specific domain would act and make decisions. It refers to all technical, scientific, and social aspects of building, maintaining, and using knowledge.
  - **Knowledge Repository:** A knowledge repository is a facility that stores obtained knowledge. It systematically captures, organises, and categorises obtained knowledge.
  - **Rule Selection:** The appropriate rule is selected for the hosts informed as malicious by two-tier IDS using a hybrid

approach where knowledge-driven and data-driven methodologies can be used simultaneously and independently. It consists of two components:

  - **Knowledge-driven Reasoner:** This component is crucial and takes real-time incidents (converted to knowledge instances) as inputs and then performs causal reasoning – a decision-making process to derive an appropriate rule for a reported incident. The reasoner is a complex, computationally expensive software system that operates on knowledge within the knowledge repository and outputs recommended rule(s) to the rule deployment subsystem.
  - **Data-driven Classifier:** This module applies various models to select an appropriate rule for a reported incident from the anomaly detection subsystem. It correlates the information from ML/DL-based model training components and passes appropriate rules to the rule deployment subsystem.

– **Machine Learning (ML)/Deep Learning (DL)-based Model:** This module uses data to incrementally improve the model's ability to select a better response by training on available data.

- **Rule Deployment:** This component performs all the activities related to flow rules needed to direct/change the network flow according to the informed host status. It consists of three components: Rule verification, prioritisation, and implementation.
  - **Rule Verification:** Rule verification checks that the generated rule is appropriate for the reported host incident. It is a valuable feature that evaluates the rule effectiveness, calculated after response deployment as a rule impact score. The selected rule can be evaluated by an analyst for confirmation if required.
  - **Rule Prioritisation:** The rule prioritisation feature ranks response rules according to a function that depends on various factors, such as the incident severity, deployment time, and impact benefits, which help decide the rule's deployment time. This step also allows for human intervention so analysts can prioritise rules based on their experience.
  - **Rule Implementation:** This component launches the rules to the SDN controller. Various techniques can be used to implement the rule appropriate for the host incident reported as per 1 rule 1 host, 1 rule n hosts, n rule 1 host or n rule n host relation. The implementation technique can be checked by an analyst for confirmation if required.
- **Human Intervention:** It allows experts to check and alter the rules before deployment.
- **Rule Impact Calculator:** Rule Impact Calculator feature verifies the status of the environment through quality checks and counts impactful response rules for better rule selection for the future. This component calculates a value to evaluate the effectiveness of a rule deployed within an IoT environment. It can be calculated based on various factors defined by domain experts. The impact score helps to update the knowledge base, which in turn helps to improve the overall performance of the proposed hybrid system. For example, by checking if the system performance is improved/degraded after we have deployed the rule, we can determine the impact of that rule on that attack. The impact score is a supplementary input to our system. It consists of three subsystems:
  - **Benign Traffic Rule Score:** It calculates the score for the rules deployed for benign traffic.
  - **Rule Impact Score:** It calculates the score for the rules deployed for anomalous traffic, i.e., for informed attack.

Table 1
Notation table.

Notation	Abbreviation
$\mathbf{A_H}$	Anomaly header
$\mathbf{A_T}$	Traffic alert
$\mathbf{A_D}$	Domain
$\mathbf{B_D}$	Co-domain
BM	Buffer memory
$\mathbf{D_{KD}}$	Decision based on Knowledge-based Model
$\mathbf{D_{DD}}$	Decision based on Data-driven Model
$\mathbf{H_{MAC}}$	Host MAC
$\mathbf{H^R_{MAC}}$	MAC address of rated Host
MAC	Medium Access Control
$\mathbf{Q_{GAIN}}$	Quality Gain
$\mathbf{Q_{DI}}$, $\mathbf{Q_{AR}}$	Quality During Incident, After Recovery
$\mathbf{Q_{BI}}$	Quality Before Incident
$\mathbf{R^D}$	Vector Space with **D** dimensions
$\mathbf{R^M}$	Vector Space with **M** mapped dimensions
$\mathbf{S_R}$	Snort rule
T	Incoming Traffic
$\mathbf{T_H}$	Packet header
$\mathbf{T^{KD}_L}$	Malicious Traffic Label
$\|\|w\|\|^2$	Euclidean Distance
X	Feature Vector

  - **Host Status:** It records the status of the IoT devices (hosts). If the host was detected as attacked or had a high probability of being attacked, the status will be updated accordingly. Rules that were deployed and the score of each rule for the host are recorded.

3. Proposed ensemble IDS

An ensemble IDS forms the first major component of the proposed SecureFlow architecture. It forms a two-tier system that coordinates the Knowledge-based IDS and Data-driven IDS to detect anomalies, intrusions and attacks. The incoming traffic from different ports is passed through these two IDS and according to the collaborative ranking received from both IDS, the traffic is labelled as false or true. The working of Knowledge-based IDS and Data-driven IDS is described in the subsequent sections. The notations used throughout the proposed work are highlighted in Table 1.

3.1. Knowledge-based IDS

The knowledge-based IDS, also known as Signature-based mechanism helps to detects the anomalies and attacks by comparing the incoming traffic patterns with the stored patterns in the database (stored in the forms of rules). In the proposed work, Snort,[1] one of the most popular signature-based IDS is used to filter the incoming traffic using pre-defined set of rules. This model fetch the incoming traffic and forwards it to the detection engine that generates the alerts according to the identified malicious activity in the network traffic. The architecture of Snort-based IDS is shown in Fig. 3.

Snort-based IDS comprises of several components, i.e., packet decoder, packet pre-processors, detection engine, and alert GeL (generator and Logger). These components are described below.

- **Packet Decoder:** The packet decoder collects the incoming traffic packets from the network and forwards them to the other components for pre-processing. It determines the data packet profile including the underlying network protocols, size of the packet, and location. This packet profile is used bu other components of snort as required. Decoder also checks the packet header to identify any anomalies (e.g., invalid size) and raise an alert if required.

[1] https://www.snort.org.

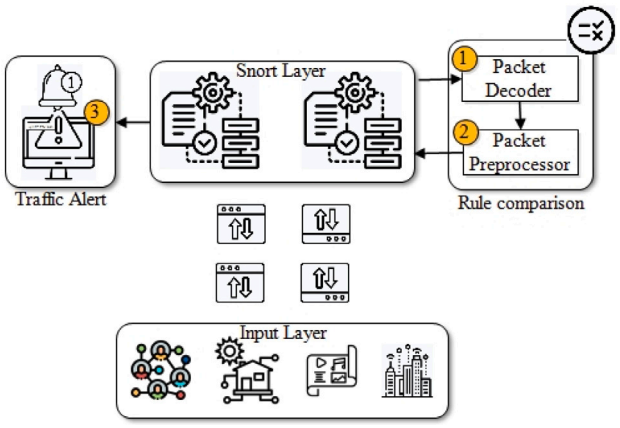


Fig. 3. Snort-driven IDS.

- **Packet Pre-processors:** The pre-processors receives the packets from the packet decoder and initiates packet sniffing. The pre-processor fragments the packets, arrange the data, and modifies the packets as desired. They enable various services with corresponding pre-processors that match and verify anomalies in the respective service.
- **Detection Engine:** The most critical component of Snort is the detection engine as it is responsible to identify if any unwarranted activity or intrusion signature exists in data packet. Detection engine applies a series of rules chained together in configuration files on each packet. If the rules are matched, the action corresponding to that rule is initiated or the suspicious packet is dropped. This may also impact the network performance in real time as it can result in additional latency.
- **Alert GeL (generator and Logger):** If the detection engine identifies an intrusion, i.e., a rule is matched, this component logs it and generates an alert that is configured through a configuration file.

The workflow of Knowledge-based IDS shown in Algorithm 1. The incoming traffic packet (**T**) from a specific host (**H**) is forwarded to snort configured environment to check if it is malicious or normal. Packet decoder generates the information profile of **T**. It extracts the packet header ($\mathbf{T_H}$) checks and verifies it for anomalies. If an anomaly is found in $\mathbf{T_H}$, an header anomaly alert ($\mathbf{A_H}$) is generated and a log is stored in $\mathbf{BM_S^{KD}}$. An intimation is sent to the SDN controller for executing dynamic configuration system (Section 5). If the $\mathbf{T_H}$ is normal, the information profile is forwarded to the corresponding pre-processor (based on the type of service). The pre-processor fragment **T**, arrange the data in **T** as required, and modify it so that the detection system cannot be fooled. The detection engine match profile of **T** with the configured snort rules ($\mathbf{S_R}$). If **T** matches with $\mathbf{S_R}$, an alert ($\mathbf{A_T}$) is generated and an intimation is sent to the SDN controller for executing dynamic configuration system (Section 5). The Medium Access Control (**MAC**) of the host and malicious traffic label ($\mathbf{T_L^{KD}}$) is stored in $\mathbf{BM_S^{KD}}$.

3.2. Data-driven IDS

Due to the rapid rate of data generation and varied patterns of the IoT environment, a regular updates are required to add new anomalous signatures, i.e., knowledge base. This process is time-consuming and tedious. Thus, a data-driven IDS that uses machine learning must develop a trustful way to label data packets as malicious or intrusive. Under this spectrum, the machine learning-based IDS comprises data pre-processing, classification of packets to find malicious data patterns

Algorithm 1 Snort-driven IDS

INPUT: Traffic packet (**T**), Packet header ($\mathbf{T_H}$), Rule: ($\mathbf{S_R}$)
OUTPUT: Alerts: (**A**, $\mathbf{A_H}$), Buffer Memory: ($\mathbf{BM_S^{KD}}$), Traffic Label ($\mathbf{T_L^{KD}}$)

```
while (T ≠ null) do
    DECODE → T
    CHECK → T_H
    if T_H == ANOMALOUS then
        GENERATE ALERT → A_H
        CALL SDN Controller
        EXECUTE → Dynamic Configuration System
        STORE Host MAC and T_L^KD → BM_S^KD
    else
        PROCEED to next step
    end if
    PRE-PROCESS → T
    MATCH T → S_R
    if (TRUE) then
        GENERATE ALERT → A_T
        CALL SDN Controller
        EXECUTE → Dynamic Configuration System
        STORE Host MAC and T_L^KD → BM_S^KD
    else
        FOLLOW Flow Table; FORWARD to next hop
    end if
end while
```

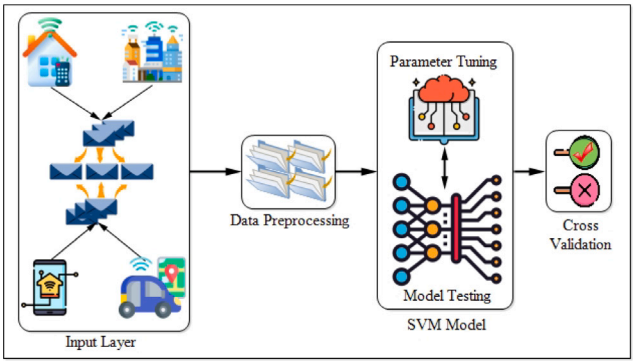


Fig. 4. SVM-based IDS.

and evaluation/recognition of abnormal behaviour. This work configures SVM to classify the traffic into two defined classes (Normal and malicious). Fig. 4 depicts a working model of SVM-based IDS.

SVM will help to sort the incoming data packets into two categories, i.e., normal and anomalous/incident, by differentiating between their similar features and characteristics. The proposed SVM-based IDS involves several phases. The work of each phase is discussed below.

3.2.1. Learning phase

The learning phase of the model is defined in the steps mentioned below.

- The domain ($\mathbf{A_D}$), i.e., the values that go into the function and co-domain ($\mathbf{B_D}$), i.e., the values that may possibly come out of a function, are identified as the possible inputs and outputs. A function (f) is defined to map $\mathbf{A_D} \rightarrow \mathbf{B_D}$ to extract the model's actual output, i.e., range.

$$f : \mathbf{A_D} \rightarrow \mathbf{B_D} \tag{1}$$

- The provided input data to be classified using SVM is represented with a feature vector, **X** as below.

$$\mathbf{X} \in \mathbf{R^D} \tag{2}$$

where, $\mathbf{R^D}$ is the vector space and **D** represents the dimension of the input domain.
Now, the transformed feature space for each input feature is mapped to a transformed vector. This is defined as below.

$$f(\mathbf{X}) : \mathbf{R^D} \mapsto \mathbf{R^M} \tag{3}$$

where **M** is the mapped dimension of the vector space.

- The points are represented in the space; now, they need to be separated according to their features using the hyperplane equation defined below.

$$\mathbf{H} : w^T(\mathbf{X}) + b = 0 \tag{4}$$

- For better segregation of the points in the space, the error rate of miss-classification must be minimal. Calculate the distance between the data points and the hyperplane line to minimise the error rate. The equation to find the distance is mentioned below:

$$d_{\mathbf{H}}(f(\mathbf{X}_0)) = \frac{|w^T(f(\mathbf{X}_0)) + b|}{\|w\|^2} \tag{5}$$

The euclidean distance $\|w\|^2$ for length w is calculated by:

$$\|w\|^2 = \sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2} \tag{6}$$

- Eq. (5) is iterated till the data points are not saturated and the generated data points are categorised.

3.2.2. Classification phase

- The generated data points in the learning phase are invoked with the testing file that contains the test data to check the accuracy of the trained model.
- This phase creates a prediction file used in post-processing activities.

3.2.3. Post-processing phase

- The generated prediction file contains information on the positive and negative results of the classification.
- By using positive and negative classifications, the precision and recall of the trained model are generated.

The working of SVM-driven IDS is depicted using Algorithm 2. After training the SVM model described in the learning phase, **T** (Traffic) from various hosts is forwarded to the trained model for verification. The trained model label **T** as Normal (**N**) or Malicious (**M**). If the label of the traffic is **N**, it follows the flow table and moves to the next hop. If the label of the traffic is **M**, an alert flag is raised, and the dynamic configuration system is triggered to generate an appropriate flow rule. The **MAC** address of the host and malicious traffic label ($\mathbf{T_L^{DD}}$) is stored in the buffer memory ($\mathbf{BM_S^{DD}}$).

Algorithm 2 Data-driven IDS

INPUT: Traffic: **T**
OUTPUT: Alert: (**A**), Buffer Memory: ($\mathbf{BM_S^{DD}}$), Traffic Label ($\mathbf{T_L^{DD}}$)

```
while (T ≠ null) do
    DECODE → T
    PRE-PROCESS → T
    MAPPING T ⇔ Trained model
    FETCH Class of T: {A, B}
    if (T == M) then
        GENERATE ALERT → A_T
        CALL SDN Controller
        EXECUTE → Dynamic Configuration System
        STORE Host MAC and T_L^DD → BM_S^DD
    else
        FOLLOW Flow Table; FORWARD to next hop
    end if
end while
```

4. Host status rating mechanism

In this section, the reliability of the traffic forwarding host is verified according to the rating provided by the two-tier IDS framework. After receiving the decisions from Snort-driven and Data-driven IDSs, three cases can be possible, i.e., (Malicious-Malicious, Normal-Malicious, and Inconclusive results). The ranking categories and severity labels are defined based on the decision from both categories of IDS. Table 2 shows the possible cases and severity defined accordingly. The non-conflicting decisions are highly severe, whereas the conflicting decisions are labelled suspicious, requiring further intervention from a security expert. These cases are described under three categories, i.e., Identical decisions, Conflicting decisions, and Inconclusive results. These three cases are described below.

Table 2
Ranking and severity based on decisions by IDS.

Snort-based IDS	SVM-driven IDS	Decision	Severity
M	M	Identical (M)	H
N	M	Conflict	S
M	N	Conflict	S
N	N	Identical (N)	L
INC	N	Inconclusive conflict	S
N	INC	Inconclusive conflict	S
INC	M	Inconclusive conflict	S
M	INC	Inconclusive conflict	S

M: Malicious; N: Normal; INC: Inconclusive; H: High; S: Suspicious; L: Low.

4.1. Case 1: Identical decisions

In this case, after analysing the traffic, the ensemble IDS declared the incoming traffic as malicious traffic or normal, i.e., knowledge-driven and data-driven IDS make the same decision. The host **MAC** ($\mathbf{H_{MAC}}$) and generated labels $\mathbf{T_L^{KD}}$ and $\mathbf{T_L^{DD}}$ are stored in the $\mathbf{BM_S^{KD}}$ and $\mathbf{BM_S^{DD}}$, respectively. For example, if $\mathbf{H}_{(00:1B:44:11:3A:B7)}$ host generated ten packets and they are forwarded to the configured network. The ensemble IDS declared all ten packets generated by $\mathbf{H}_{(00:1B:44:11:3A:B7)}$ are malicious; then it is blacklisted.

4.2. Case 2: Conflicting decisions

In this case, the decisions provided by both IDS are conflicting, i.e., one IDS declared the traffic as **M** (Malicious) and the other IDS declared it as **N** (Normal). The previous n packets from the host are inspected in this case. The final decision is taken based on the status of the previous n packets. Here, the host rating ($\mathbf{H_{MAC}^R}$) is calculated based on the following equation.

$$\mathbf{H_{MAC}^R} = \frac{\mathbf{D_{KD}} + \mathbf{D_{DD}}}{2} \tag{7}$$

If $\mathbf{H_{MAC}^R} > 0$, the selected $\mathbf{H_{MAC}}$ is suspicious, it is blocked and referred for further inspection by a security expert.

Let us take an example where we inspect the previous ten packets (n = 10) sent by $\mathbf{H}_{(00:1B:44:11:3A:B7)}$. Let us say that $\mathbf{IDS}_1$ declared all ten packets malicious, whereas $\mathbf{IDS}_2$ declared just three packets malicious. So, the $\mathbf{H_{MAC}^R}$ is calculated below.

$$\mathbf{H_{MAC}^R} = \frac{10 + 3}{2} = 6.5 \tag{8}$$

Now, both IDS are saying that at least some of the packets sent by $\mathbf{H}_{(00:1B:44:11:3A:B7)}$ are malicious and $\mathbf{H_{MAC}^R} > 0$. Thus, the decision will be that $\mathbf{H}_{(00:1B:44:11:3A:B7)}$ is malicious, should be blacklisted.

There can be another case in the conflicting decision where one IDS can provide an inconclusive output (e.g., an error). Let us say that $\mathbf{IDS}_1$ declared all ten packets as malicious, whereas $\mathbf{IDS}_2$ provided an inconclusive decision. Here, we consider the inconclusive decision as 0 and proceed based on Eq. (avg12). So, the $\mathbf{H_{MAC}^R}$ is calculated below.

$$\mathbf{H_{MAC}^R} = \frac{10 + 0}{2} = 5 \tag{9}$$

Now, $\mathbf{H_{MAC}^R} > 0$. Thus, the decision will be that $\mathbf{H}_{(00:1B:44:11:3A:B7)}$ is malicious, should be blacklisted. Thus, even if one IDS does not work, provide decisions or provide inconclusive decisions, the proposed SecureFlow framework can provide a final decision. This proves the robustness and fault-tolerant capabilities of SecureFlow.

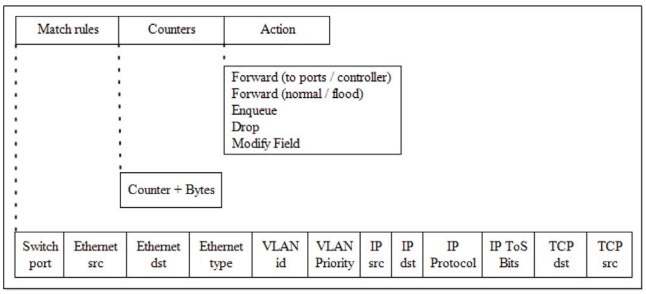


Fig. 5. Flow table architecture.

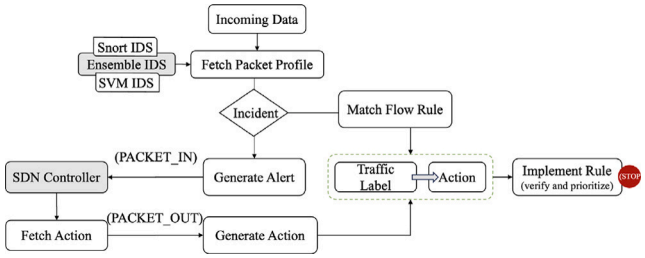


Fig. 6. Flowchart of dynamic rule generator.

5. Dynamic rule configuration system

The flow rules are the backbone of SDN architecture and help to route incoming traffic to the desired destination. The flow rules are generated, verified, prioritised and installed on the flow table by the SDN controller. The Flow table contains the information for packet lookup, forwarding, and blocking. Each flow entry in the flow table has twelve tuple components related to the packet. These flow entries are shown in Fig. 5. Whenever a data packet from IoT devices is forwarded, the packet header matches the flow entry in the flow table to find a feasible action. The configured switch activates the lookup key to match the flow entry in the switch flow table. The action key value directs the packet to the underlying network if a match is found. If a match is not found, the **PACKET_IN** key is generated and forwarded to the SDN controller to generate a suitable flow rule. SDN controller generates a new flow entry/rule and forwards **PACKET_OUT** to the configured switches. Accordingly, the packet is directed to the destination. The proposed dynamic rule configuration system comprises several steps: rule generation, verification, prioritisation and implementation. The proposed system is described in the following sections.

5.1. Hybrid and dynamic rule generation

The incoming traffic from various input devices is forwarded to the connected switches in the configured network. The integrated knowledge-based and data-driven IDSs analyse the incoming traffic and update the traffic label parameter accordingly. The packet matches the flow rules stored on the configured switches in the underlying network. The mandatory elements of the flow rule are mentioned in Table 3.

The traffic label parameter can depict several anomalies/intrusions or unwarranted incidents. If a specific type of anomaly/intrusion or unwarranted incident is found, an ensemble flow rule generator (knowledge-based and data-driven flow rule generator) generates a new flow rule. The working of the proposed model is highlighted in Fig. 6.

The two types of flow rule generators proposed in this work are described below.

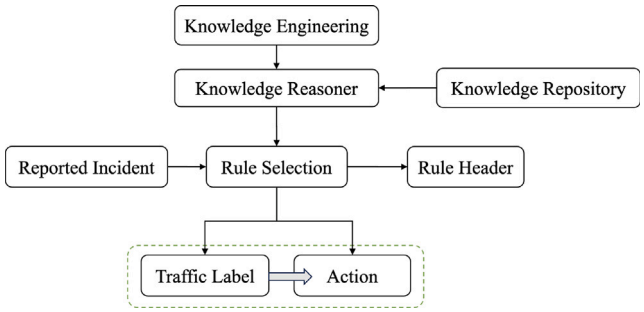


Fig. 7. Knowledge-based rule generation workflow.

Table 3
Flow rule header format.

Category	Elements
Rule header	Action Protocol Source address Source port Direction Destination address Destination port

Table 4
Knowledge-based dynamic rule generator.

Traffic label ($\mathbf{T_L}$)	Action ($\mathbf{A_c}$)
Suspicious payload size ($\mathbf{T}_1$)	*Port number of Honeypot* ($\mathbf{A}_{c1}$)
$R \geq 80\%$ ($\mathbf{T}_2$)	*Block* ($\mathbf{A}_{c2}$)
Denial of service ($\mathbf{T}_3$)	*Enqueue* ($\mathbf{A}_{c3}$)
TT = Multi-cast ($\mathbf{T}_4$)	*Modify field* ($\mathbf{A}_{c4}$)

5.1.1. Knowledge-based rule generation

The traditional rule generation method is based on the knowledge base. There are several components involved in knowledge-based rule generation. The rule engineering provides domain knowledge, and the knowledge repository (Tables 3 and 4) integrates to select an appropriate rule for all the reported incidents. The reported incident is matched with the action field based on the traffic label. These components are shown in Fig. 7.

If a match is found, the **PACKET_IN** key is generated and forwarded to the configured controller to generate the new rule as per the traffic label. The **Action** value is fetched from the Table 4 according to the traffic label, and **PACKET_OUT** is generated to define the newly generated rule. It is forwarded to the configured switches in the underlying network. For example, if the traffic analyser finds an unexpected packet size, the traffic is forwarded to the connected Honeypot. In case of flooded messages from the same host, the messages are en-queued for further investigation, and the payload of the packets is matched. Accordingly, suitable action is generated to avoid the denial of service attack in the network. Suppose the same message is multi-casted on several destinations. In that case, the packets are put on hold, and the priority of en-queue messages is verified, and accordingly, the traffic is forwarded to the configured network.

5.1.2. Data-driven rule generation

The modern-day method is driven by data collected from previous incident scenarios. This data is used to train a machine/deep learning model. The trained models provide the suitable flow rule entry for any reported intrusion or unwarranted incident. The proposed data-driven rule generation approach is shown in Fig. 8.

Once the ML/DL model is trained, the reported incidents (by ensemble IDS) are used to predict new flow rules in the flow table. The action about the selected flow entry is selected, and after that, further processing is initiated.

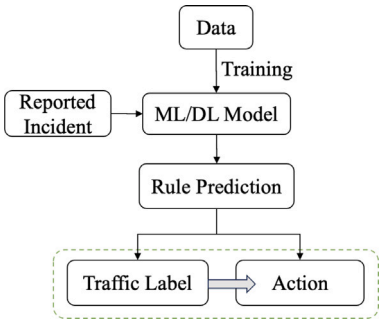


Fig. 8. Data-driven rule generation workflow.

Table 5
Rule prioritisation categories.

Severity level	Priority
Critical	High priority
Minor	Guaranteed
Low	Best effort

5.2. Rule verification

In the proposed scheme, the rules defined in Table 4 are verified before implementation. First, the syntax and semantics of the generated rules are checked and verified if they meet the required criteria and comply with the proper structure. The traffic label ($\mathbf{T_L}$) and Action ($\mathbf{A_c}$) shown in Table 4 are labelled as $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4$ and $\mathbf{A}_{c1}, \mathbf{A}_{c2}, \mathbf{A}_{c3}, \mathbf{A}_{c4}$, and they are mapped against each other. A wrong match can also lead to unsuccessful verification. Initially, the flow rule header ($\mathbf{FR_H}$) is fetched to ensure that mandatory header credentials are verified. The traffic label is mapped with the action label if the header is verified and compliant. If mapping is correct, the selected rule is labelled as *"Verified Successfully"* ($\mathbf{V_S}$). If mapping is incorrect, the output says *"Verification Failed"* ($\mathbf{V_F}$). If a rule fails verification, the SDN controller is notified, and the said rule is removed from the record. The SDN controller initiates the rule implementation process if the rule clears verification. The above described working of the verification process is depicted in Algorithm 3.

Algorithm 3 Rule Verification Algorithm

INPUT: $\mathbf{T}$, $\mathbf{T_L}$, $\mathbf{A}$, $\mathbf{FR_H}$.
OUTPUT: $\mathbf{V_S}$, $\mathbf{V_F}$

```
while (T ≠ null) do
    FETCH T_L → (T1,T2,T3,T4)
    if (FR_H ⊃ (∀ Elements)) then          ▷ (Refer Table 3)
        if T_L ↦ A then
            RETURN V_S
        else
            RETURN V_F
        end if
    else
        RETURN V_F
    end if
end while
```

5.3. Rule prioritisation and implementation

The packet forwarding decision using SDN is a flow-based rather than a destination-based approach used in the traditional schemes. The sensitivity of the flow decides the priority of the flow rule implementation according to the severity of intrusion/incident reports by IDS. If the severity level is "CRITICAL", the flow is labelled as *"High Priority"*. If the severity level is "AVERAGE", the flow is labelled as *"Guaranteed"*. In case the level is "LOW", the flow is labelled as *"Best Effort"*. The criteria for rule prioritisation is described in Table 5.

Table 6
Rule impact score.

Gain ($\mathbf{Q_{GAIN}}$)	Impact	Score ($\mathbf{R_{IS}}$)
≥0.75	Very high	1
≥0.5 & <0.75	High	0.75
≥0.25 & <0.5	Average	0.5
<0.25	Low	0.25

According to the priority of the flow rule, the SDN controller implements the rule as per 1 rule 1 host, 1 rule n hosts, n rule 1 host or n rule n host relation.

5.4. Rule impact calculator

Introducing new rules in the existing system may affect the system's performance. Quality of Service (QoS) helps verify the underlying system's performance. In the proposed work, the QoS is calculated at three stages, i.e., before the incident, during the incident and after implementing the rule response. For example, the QoS for a specific metric related to network performance was 100% before an incident. However, after being subject to the incident, QoS dropped to 10%. Now, the proposed scheme ensures a response to the incident by generating a new flow rule. So, the SDN controller generates a new rule and implements it to restore the network performance. After using the new rule is implemented, the QoS is re-recorded. If the QoS witnessed a gain compared to its value before the incident, then the generated rule has an impact. The performance gain ($\mathbf{Q_{GAIN}}$) is calculated as below.

$$\mathbf{Q_{GAIN}} = \frac{\mathbf{Q_{AR}} - \mathbf{Q_{DI}}}{\mathbf{Q_{BI}}} \tag{10}$$

where $\mathbf{Q_{DI}}$, $\mathbf{Q_{AR}}$, and $\mathbf{Q_{BI}}$ represent quality metric value (in %) during the incident, after rule response, and before the incident, respectively.

For example, if QoS before the incident was 90%, during the incident it dropped to 10%, and after response, it raised to 80%, then the performance gain is calculated as below.

$$\mathbf{Q_{GAIN}} = \frac{80 - 10}{90} = 77.78 \tag{11}$$

Based on the value of $\mathbf{Q_{GAIN}}$, an impact score ($\mathbf{R_{IS}}$) is provided to the rule under consideration. The category of rule impact score is highlighted in Table 6.

The effectiveness of the generated rule is calculated to check the impact of the rule in future decisions. The newly gained impact score is added to the overall impact score of the rule and varies as per its performance over its lifetime. The overall impact score ($\mathbf{R^O_{IS}}$) for a specific rule is calculated as below.

$$\mathbf{R^O_{IS}} = \mathbf{R^C_{IS}} \pm \mathbf{R_{IS}} \tag{12}$$

where $\mathbf{R^C_{IS}}$ represents the current impact score of a rule.

6. Results and discussion

The proposed **SecureFlow** framework helps to detect any unwarranted incidents in the incoming IoT traffic using ensemble IDS. Then, it generates a flow rule through the SDN controller to reconfigure the network to restore performance. So, to validate the effectiveness, various experiments have been performed for different subsystems of **SecureFlow**. We have segregated the experiments into three parts: (a) Knowledge-based IDS, (b) Data-driven IDS, and (c) Rule configuration system. The details about the simulation setup, dataset, and obtained results are provided in the subsequent section. The proposed system uses the Linux-based platform to evaluate the performance of **SecureFlow**. Table 7 details the tools/systems used to design the simulated scenario.

A floodlight controller is deployed in the system design to extract the real-time outcomes. The topology design included five connected

Table 7
Simulation design elements.

Platform/Tools	Description
Linux (Ubuntu)	18.04
java	1.8.0_312
Apache ant	1.10.5
Apache Maven	3.6.0
Snort	2.9.7.0
Controller	Floodlight
Python	3.9.1
Honeypot Server IDS	–

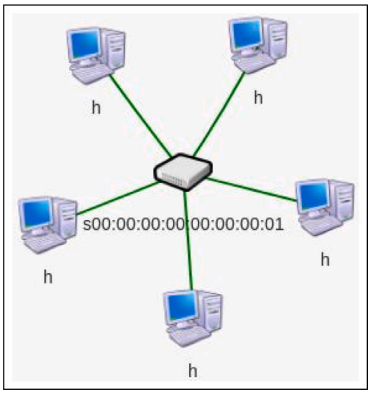


Fig. 9. Topology layout.

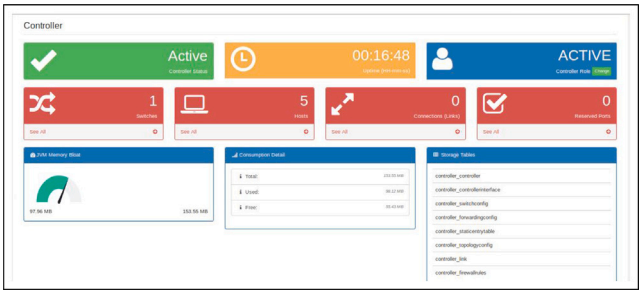


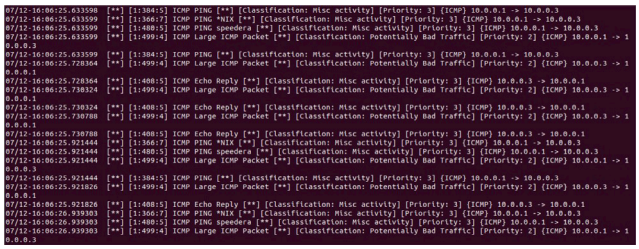
Fig. 10. Experimental dashboard.

IoT endpoints and one centralised controller, as shown in Fig. 9. The IoT endpoints connect with the OpenFlow switch for efficient centralised network supervision in the configured topology. The experimental dashboard of controller accessed on the local-host address is shown in Fig. 10.

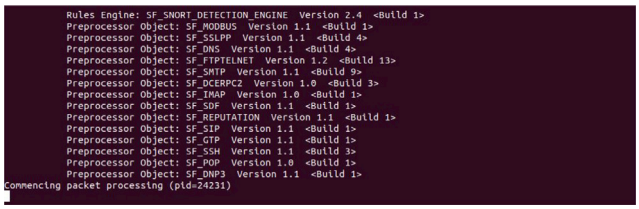
6.1. Knowledge-based IDS

In the first phase of the experimentation, the knowledge-based IDS uses Snort to detect unwarranted incidents in the incoming IoT traffic. The signatures/rules of the particular IoT traffic are defined and logged in a specific file. Further, as per the type of application, the specified rule files are included in the main snort configuration file for the filtration process. After this phase, the alerts for intrusion detection are generated and forwarded to the SDN controller for further action based on the rule configuration system.

The Snort-based IDS system has been visualised when it is activated and deactivated. Fig. 11(a) and (b) depict the command system visualisation showing the snort-enabled and standard environments. Here, we computed the execution time for the proposed Snort-based IDS, depicting minimum, maximum, and average times for the deployed scenario. Moreover, the snort-based IDS and standard environments are compared based on the computation time, as shown in Fig. 12. Fig. 12(a)



(a) Activation of Snort for Traffic Filtration



(b) Deactivating Snort for Traffic Filtration

Fig. 11. Console-based filtration.

Table 8
Dataset statistics.

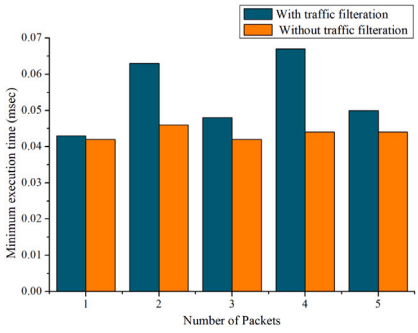
Features	Max.	Min.	Mean	Median	Count
duration	57 715	1	218.86	0	22 577
protocol_type	tp	icmp	–	–	22 577
service	whois	IRC	–	–	22 577
src_bytes	62 825 648	20	10 395.45	54	22 577
dst_bytes	1 345 927	20	2056.02	46	22 577
urgent	3	0	0.0007	0	22 577
num_root	878	1	0.11	0	22 577
srv_count	511	1	211.212	0	22 577
dst_host_count	255	1	101.2356	0	22 577
dst_host_srv	255	1	132.75	0	22 577

shows the minimum execution time the configured environment takes to handle the generated IoT traffic. In Fig. 11(a), the minimum execution time (msec) of different packets in both environments is compared, and the execution time of the snort-based environment is higher as the extra filtration phase is included. The Snort-enabled environment consumes additional time to execute the provided traffic, as the traffic header must be matched with the defined rules to analyse the type of incoming traffic (Normal/Anomalous). Thus, Fig. 12(b) reflects the maximum time to evaluate the incoming IoT traffic. The configured rules in the Snort-enabled environment for anomalous traffic alert generation increase the overall time consumption. Fig. 12(c) and (d) show the average time consumption and maximum deviation in time consumption to process the incoming traffic. Fig. 12(b) and (c) reflect higher maximum and average execution times for snort-based IDS as compared to the standard environment.

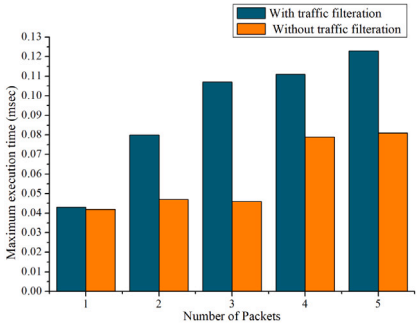
6.2. Data-driven intrusion detection

Data-driven IDS uses a supervised learning approach, i.e., SVM, to analyse unwarranted incoming traffic packets' incidents. The statistical information about the considered dataset[2] is used during training the SVM model. There are 22 577 entries in the dataset. The details of the selected parameters during training of the SVM model and the statistical representation of the dataset are represented in Table 8. The SVM-based model is trained for anomalous traffic detection on the Jupyter Notebook platform. The performance of the SVM-based IDS is measured based on various performance metrics (accuracy, recall,

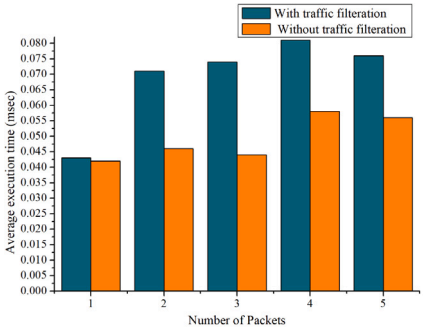
[2] https://www.unb.ca/cic/datasets/nsl.html.



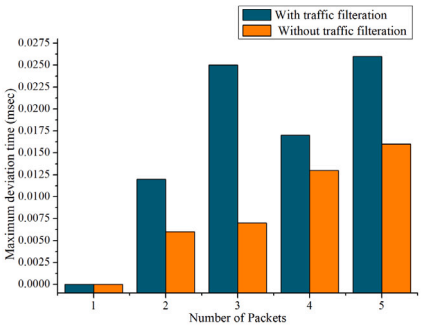
(a) Minimum Execution Time



(b) Maximum Execution Time



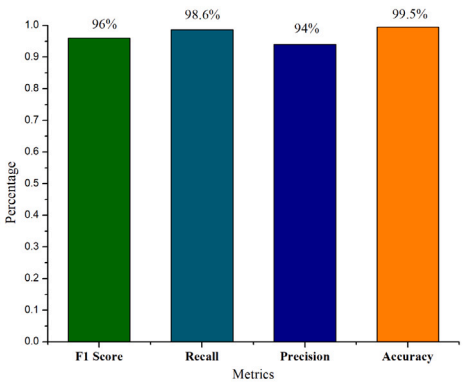
(c) Average Execution Time



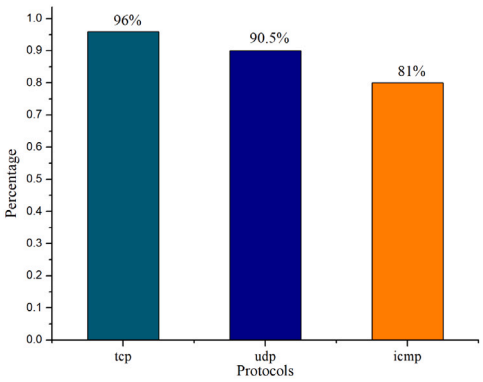
(d) Maximum Deviation Time

Fig. 12. Experimental results.

precision, and F1 score). The accuracy comes out to be 99.5% for the deployed model. Similarly, the F1 score, precision and recall are calculated as 96%, 94% and 98.6%, respectively. Fig. 13(a) depicts the results obtained for the deployed scenario for SVM-based IDS. We have performed a protocol-based traffic analysis (*tcp*, *udp*, and *icmp*) by segregating the dataset for further validation. The accuracy measured



(a) Results obtained for SVM-based IDS



(b) Protocol-wise Classification of Accuracy

Fig. 13. Results obtained for SVM model.

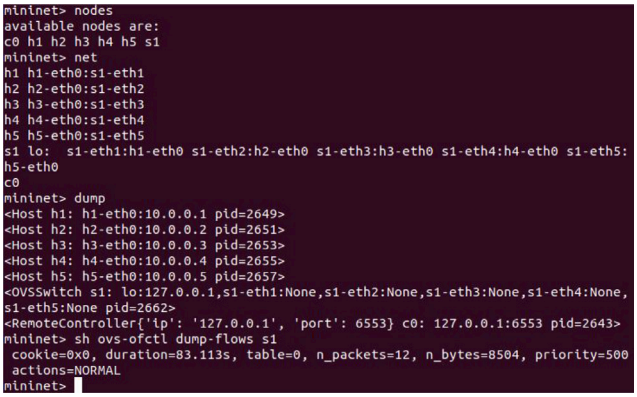
while detecting intrusions in the protocol-wise traffic is shown in Fig. 13(b).

6.3. Flow rule generation

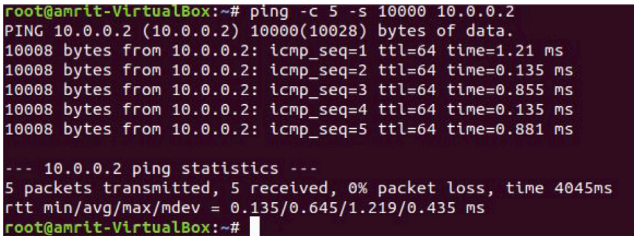
The ensemble IDS analyses the incoming traffic and alerts the SDN controller if any incident is detected. The SDN controllers respond to the incident by generating a flow rule. The flow rule is selected, verified, prioritised, and implemented for action. The proposed rule configuration system was set up on Mininet emulator.[3] The proposed topology is deployed, and all the host statistics are implemented as depicted in Fig. 14(a). After this, the packet transmission is initiated based on various traffic packets. Fig. 14(b) depicts the *icmp* packets transmitted over the deployed scenario. After this, when the incidents are detected, the alert is generated. For instance, Fig. 14(c) depicts the alert "BAD TRAFFIC". The priority is set, and a rule is generated based on which the further action or destination for the packet is decided. The generated rule is implemented, and the results show that four packets were transmitted. The rule suggested dropping the packets, and all four packets were dropped. Fig. 14(c) shows 100% packet loss. The computation time was 3054 ms, as shown in Fig. 15.

The proposed rule configuration system was also compared for its performance to the conventional controller (Floodlight). The time consumption is measured in contrast to an increase in the number of flow rules generated by the controller. The results (as shown in Fig. 15) depict an increase in time consumption, but this increase is negligible. This proves the effectiveness of the proposed system.

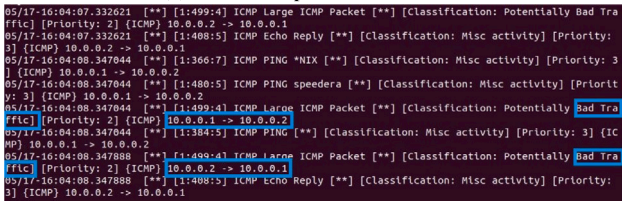
[3] http://mininet.org.



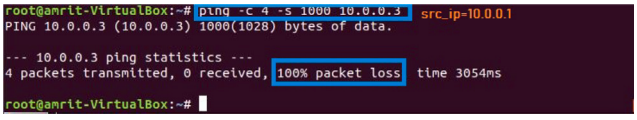
(a) Topology nodes and interfaces



(b) Successful packet transmission



(c) Alert generation of bad traffic



(d) Packet loss due to host blockage

Fig. 14. Implementation of flow rule configuration system.

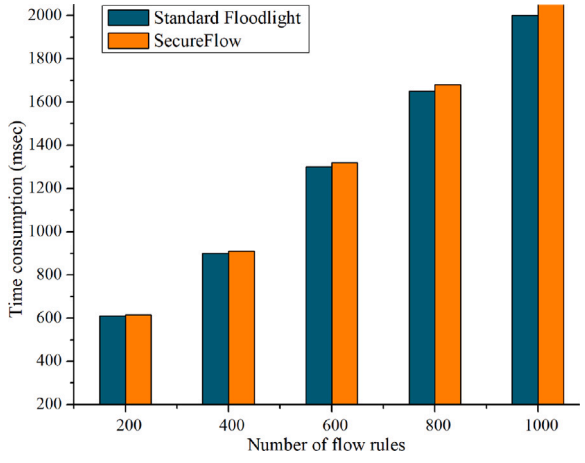


Fig. 15. Time consumption.

7. Conclusion

This paper introduces SecureFlow, an ensembled framework for intrusion detection and dynamic rule configuration for IoT environments. The proposed framework integrates knowledge-based and data-driven IDS to provide a dual-level detection system that generates alerts and rates the malicious hosts accordingly. After this, the dual-layer rule configuration module generates a response in the form of new rules, verifies the generated rules, prioritises them, and finally implements them to restore the affected IoT network. The SDN-enabled environment allows customising the rules according to the detected incidents and responding dynamically. The proposed framework was validated in a simulated environment, and the performance was measured for various sub-systems: Snort-based IDS, data-driven IDS, and rule configuration module. The proposed scheme provides promising results based on standard performance metrics (such as execution time, deviation time, accuracy, and many more). The scenario to depict the bad traffic in the configured IoT network and the working of the rule configuration modules is also depicted. This work includes limited attack scenarios and rules for validation. So, the future work can include validations at scale considering diverse range of attacks/incidents and data quality concerns for real-time applications.

CRediT authorship contribution statement

Amritpal Singh: Methodology, Validation, Writing – original draft, Writing – review & editing. **Pushpinder Kaur Chouhan:** Conceptualization, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Gagangeet Singh Aujla:** Conceptualization, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing.

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

No data was used for the research described in the article.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) for project CHEDDAR: Communications Hub For Empowering Distributed ClouD Computing Applications And Research [Grant number EP/X040518/1].

References

[1] U. Demirbaga, G.S. Aujla, MapChain: A blockchain-based verifiable healthcare service management in IoT-based big data ecosystem, IEEE Trans. Netw. Serv. Manag. 19 (4) (2022) 3896–3907.

[2] A. Singh, G.S. Aujla, R.S. Bali, Intent-based network for data dissemination in software-defined vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (8) (2020) 5310–5318.

[3] P.K. Chouhan, S. McClean, M. Shackleton, Situation assessment to secure IoT applications, in: 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, 2018, pp. 70–77.

[4] C. Qiu, G.S. Aujla, J. Jiang, W. Wen, P. Zhang, Rendering secure and trustworthy edge intelligence in 5G-enabled IIoT using proof of learning consensus protocol, IEEE Trans. Ind. Inform. 19 (1) (2022) 900–909.

[5] R. Bedi, M. Marwaha, T. Singh, H. Singh, A. Singh, Analysis of different privacy preserving cloud storage frameworks, 2012, http://dx.doi.org/10.48550/ARXIV.1205.2738, URL https://arxiv.org/abs/1205.2738.

[6] F. Habeeb, K. Alwasel, A. Noor, D.N. Jha, D. AlQattan, Y. Li, G.S. Aujla, T. Szydlo, R. Ranjan, Dynamic bandwidth slicing for time-critical IoT data streams in the edge-cloud continuum, IEEE Trans. Ind. Inform. 18 (11) (2022) 8017–8026.
[7] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (2021) 1–27.
[8] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, A survey of intrusion detection techniques in cloud, J. Netw. Comput. Appl. 36 (1) (2013) 42–57.
[9] H. Debar, M. Dacier, A. Wespi, A revised taxonomy for intrusion-detection systems, Ann. Télécommun. 55 (7–8) (2000) 361–378.
[10] C. Kreibich, J. Crowcroft, Honeycomb: creating intrusion detection signatures using honeypots, ACM SIGCOMM Comput. Commun. Rev. 34 (1) (2004) 51–56.
[11] C.R. Meiners, J. Patel, E. Norige, E. Torng, A.X. Liu, Fast regular expression matching using small {tcams} for network intrusion detection and prevention systems, in: 19th USENIX Security Symposium (USENIX Security 10), 2010.
[12] S. Garg, A. Singh, G.S. Aujla, S. Kaur, S. Batra, N. Kumar, A probabilistic data structures-based anomaly detection scheme for software-defined internet of vehicles, IEEE Trans. Intell. Transp. Syst. 22 (6) (2021) 3557–3566.
[13] A. Khraisat, I. Gondal, P. Vamplew, An anomaly intrusion detection system using C5 decision tree classifier, in: Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2018 Workshops, BDASC, BDM, ML4Cyber, PAISI, DaMEMO, Melbourne, VIC, Australia, June 3, 2018, Revised Selected Papers 22, Springer, 2018, pp. 149–155.
[14] A. Verma, V. Ranga, Machine learning based intrusion detection systems for IoT applications, Wirel. Pers. Commun. 111 (4) (2020) 2287–2310.
[15] M. Eskandari, Z.H. Janjua, M. Vecchio, F. Antonelli, Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices, IEEE Internet Things J. 7 (8) (2020) 6882–6897.
[16] V. Kumar, A.K. Das, D. Sinha, UIDS: a unified intrusion detection system for IoT environment, Evol. Intell. 14 (1) (2021) 47–59.
[17] M.A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, H. Janicke, Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks, Future Internet 12 (3) (2020) 44.
[18] P. Singh, A. Kaur, G.S. Aujla, R.S. Batth, S. Kanhere, Daas: Dew computing as a service for intelligent intrusion detection in edge-of-things ecosystem, IEEE Internet Things J. 8 (16) (2021) 12569–12577.
[19] S. Garg, K. Kaur, S. Batra, G.S. Aujla, G. Morgan, N. Kumar, A.Y. Zomaya, R. Ranjan, En-ABC: An ensemble artificial bee colony based anomaly detection scheme for cloud environment, J. Parallel Distrib. Comput. 135 (2020) 219–233.
[20] A. Yang, Y. Zhuansun, C. Liu, J. Li, C. Zhang, Design of intrusion detection system for internet of things based on improved BP neural network, IEEE Access 7 (2019) 106043–106052.
[21] R. Chaudhary, G.S. Aujla, N. Kumar, P.K. Chouhan, A comprehensive survey on software-defined networking for smart communities, Int. J. Commun. Syst. (2022) e5296.
[22] G.S. Aujla, A. Singh, N. Kumar, Adaptflow: Adaptive flow forwarding scheme for software-defined industrial networks, IEEE Internet Things J. 7 (7) (2019) 5843–5851.
[23] A. Wani, R. Khaliq, SDN-based intrusion detection system for IoT using deep learning classifier (idsIoT-SDL), CAAI Trans. Intell. Technol. 6 (3) (2021) 281–290.
[24] J. Ashraf, N. Moustafa, A.D. Bukhshi, A. Javed, Intrusion detection system for SDN-enabled IoT networks using machine learning techniques, in: 2021 IEEE 25th International Enterprise Distributed Object Computing Workshop, EDOCW, IEEE, 2021, pp. 46–52.

Amritpal Singh is a postdoctoral research associate at Durham University, UK since October 2023. Prior to this, he was working on several academic positions for 10 years in different Universities in India. He received his Ph.D. from Chandigarh University, India in 2022. His research interests are computer networks, software-defined networks, edge-cloud computing, and digital twins.

Pushpinder Kaur Chouhan is a senior research scientist at British Telecom UK, brings over 15 years of experience in research and development across diverse fields, including distributed high-performance computing, data science, and cybersecurity. She obtained a PhD in Computer Science from ENS-Lyon, France in 2006. After completing her doctoral studies, she delved into research and development, dedicating her expertise to solving complex security challenges. In the realm of cybersecurity, she has focused on network malware detection within IoT and virtualized environments. She incorporate deception technology to anticipate and mitigate potential zero-day threats for cybersecurity. Additionally, her research extends beyond conventional cybersecurity measures. She actively explores continuous authentication based on biometrics, incorporating cutting-edge technologies to enhance security protocols.

Gagangeet Singh Aujla is an assistant professor of computer science at Durham University, United Kingdom, and a Fellow of Durham Energy Institute. Prior to this, he was a postdoctoral research associate with the School of Computing, Newcastle University, United Kingdom. He received his Ph.D. in computer science from Thapar University, India, in 2018. He received the 2022 IEEE TCSC Award for Excellence (Early Career Researcher), 2022 TEMS TC on Blockchain and Distributed Ledger Technologies Early-Career Award 2022 (Runner-up), 2021 IEEE Systems Journal Best Paper Award, and 2018 IEEE TCSC Outstanding Ph.D. Dissertation Award. The main themes of his research are energy-efficient, reconfigurable, resilient, and intelligent surfaces (smart city, smart grid, IoT-Edge-Cloud systems, healthcare systems, transportation systems). He is leading several workshop series (BlockCPS, BlockSecSDN) in conjunction with top conferences like IEEE Infocom, IEEE Globecom, IEEE ICC, IEEE/ACM UCC, and IEEE/ACM CCGrid. He is the Co-Secretary, IEEE UK and Ireland Diversity, Equality, and Inclusion Committee. He is an Area Editor for Adhoc Networks (Elsevier), Associate Editor of Concurrency and Computation: Practice and Engineering, IET Smart Grid and Frontiers in Internet of Things. He is Senior Member of the IEEE and a Member of the ACM.