Computers, Materials & Continua
DOI: 10.32604/cmc.2022.028285
Article

Tech Science Press

R-IDPS: Real Time SDN-Based IDPS System for IoT Security

Noman Mazhar[1,2], Rosli Saleh[1,*], Reza Zaba[1,3], Muhammad Zeeshan[4], M. Muzaffar Hameed[1] and Nauman Khan[1]

[1]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, 50603, Malaysia
[2]Centre for Research in Industry 4.0, University of Malaya, Kuala Lumpur, 50603, Malaysia
[3]MIMOS Berhad, National Applied R&D Centre, Kuala Lumpur, 57000, Malaysia
[4]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, 44000, Pakistan
*Corresponding Author: Rosli Saleh. Email: rosli_salleh@um.edu.my

Received: 06 February 2022; Accepted: 07 April 2022

Abstract: The advent of the latest technologies like the Internet of things (IoT) transforms the world from a manual to an automated way of lifestyle. Meanwhile, IoT sector open numerous security challenges. In traditional networks, intrusion detection and prevention systems (IDPS) have been the key player in the market to ensure security. The challenges to the conventional IDPS are implementation cost, computing power, processing delay, and scalability. Further, online machine learning model training has been an issue. All these challenges still question the IoT network security. There has been a lot of research for IoT based detection systems to secure the IoT devices such as centralized and distributed architecture-based detection systems. The centralized system has issues like a single point of failure and load balancing while distributed system design has scalability and heterogeneity hassles. In this study, we design and develop an agent-based hybrid prevention system based on software-defined networking (SDN) technology. The system uses lite weight agents with the ability to scaleup for bigger networks and is feasible for heterogeneous IoT devices. The baseline profile for the IoT devices has been developed by analyzing network flows from all the IoT devices. This profile helps in extracting IoT device features. These features help in the development of our dataset that we use for anomaly detection. For anomaly detection, support vector machine has been used to detect internet control message protocol (ICMP) flood and transmission control protocol synchronize (TCP SYN) flood attacks. The proposed system based on machine learning model is fully capable of online and offline training. Other than detection accuracy, the system can fully mitigate the attacks using the software-defined technology SDN technology. The major goal of the research is to analyze the accuracy of the hybrid agent-based intrusion detection systems as compared to conventional centralized only solutions, especially under the flood attack conditions generated by the distributed denial of service (DDoS) attacks. The system shows 97% to 99% accuracy in simulated results with no false-positive alarm. Also, the system shows notable improvement in terms of resource utilization and performance under attack scenarios. The R-IDPS is scalable, and the system is suitable for heterogeneous IoT devices and networks.

Keywords: Machine learning; Internet of things; software defined networking; distributed denial of service attacks

1 Introduction

Plethora of new application based on IoT devices overwhelmed the market, However the conventional technologies are not able to cope the challenges posed by the low power devices. This force the development of new or customization of existing conventional technologies. This makes the IoT devices prone to cyber-attacks; and has become very challenging these days to guard IoT devices from attacks. The reason is that most IoT devices have limited resources (e.g., computation capacity, storage, and power); This scarce resource is a big constraint for the use of many conventional and complicated intrusion detection systems. It has been estimated that global mobile data traffic is increased by 71% in 2017. However, by the end of 2022, big chunk of the mobile traffic data will originate from smart devices [1]. Similarly, the number of IoT malware has been increased around seven times compared to 2013. Mirai-infected IoT devices are the biggest source of launching cyberattacks, and traffic size reached 620 Gbps [2]. Botnets are another form of malicious devices around 10,263 botnets are hosted in different IoT networks, as identified in 2018 [3]. One of the biggest hits was in the financial sector as the compromised devices initiate a DDoS attack known as "IoTroop," this was discovered in 2017. This powerful attack involves more than 13,000 IoT devices [4].

Because of resource limits in terms of storage, processing power, and energy consumption, traditional intrusion detection systems (IDSs) are not suited for implementation in IoT devices. Thus, the possible solution is to use network-based IDS to implement the detection and prevention system for the IoT environment. Signature and anomaly-based techniques are the primary detection approaches in IDSs. Signatures are developed using predefined attacks patterns. One of the limitations of this approach is to detect new kinds of attacks [5]. However anomaly detection systems using statistical techniques are more effective to detect the new attacks, the problem of the high false-positive rate is one of the main challenges [5,6]. Software-Defined network technology transforms the traditional network management and network applications altogether. SDN has been used in many IDS systems to address IoT security due to its centralized control and global network transparency. SDN approach decouples the data and control planes. SDN uses the open flow (OF) protocol that enables the controller and the forwarding devices to communicate the OF tables among each other. OpenFlow protocol is device-independent and allows the SDN to operate in a heterogeneous environment [7,8].

This work develops an intrusion detection and prevention system based on an SDN network for IoT devices. The initial work has been discussed with brevity in our previous work [9]. The focus of our proposed system R-IDPS based on SDN technique is to monitors the network traffic. The basic modules of R-IDPS are detection and mitigation modules. The detection module uses machine learning for the detection of attacks based on the baseline network profile. The baseline network profile is developed using normal network traffic statistics. The mitigation module uses the OpenFlow rules to mitigate the attack and block the compromised devices in the network. For attack testbed, we use bots to launch the DDoS attacks. The novelty of the work includes the development of lite weight agents with the ability to work with vendor independent IoT devices, further training of machine learning algorithm online without shutting down the system, additional using the time series lite wight database for real-time data storage and retrieval for analysis. Finally, mitigating the attack using the SDN technology without affecting the normal network traffic by minimizing the latency for the IoT node's response during the DDoS attack. The contribution of this study is as below:

- Design and development of real-time agent-based distributed intrusion detection architecture for DDoS attack based on Machine learning approach.
- Algorithm for attack mitigation using open flow rules.
- Dataset creation for ICMP and TCP SYN flood attack types.
- Comparing the performance of proposed R-IDPS with the existing state-of-the-art IDS systems.
- Analyzing the IoT device utilization after R-IDPS implementation.

The paper is structured as follows: Section 2 illustrate the related work, Section 3 explains in detail the R-IDPS along its modules, Section 4 discuss the experimental testbed and its topology, Section 5 analyzes initial the results with the limitations and explore future research direction based on the experimental outcome and Section 6 concludes the research by summarizing the results of the work and potential future research opportunities.

2 Related Work

Pattern matching and signature-based techniques are at the heart of well-known IDS systems like Suricata, Snort, and Bro. Snort is a popular intrusion detection system developed by Cisco Systems [10], and version 2.9 now supports multi-threading architecture, as opposed to single-threaded versions previously. Suricata, an IDS/IPS open-source solution [11], is another IDS system that uses a multi-threading architecture. It is, nevertheless, more useful in large-scale network infrastructure. A study [12] discusses in detail the capability of Suricata. These IDS systems support most of the conventional network systems, and real-time processing becomes a challenge for them. However, the Snort requires fewer resources as compared to the Suricata, but still quite an issue when implemented in the resource constraints devices. it still has a limitation on the number of rules exploited for attack detection when it is implemented on resource constraint devices [13].

In our previous work [9], we discuss the implementation of R-IDPS and present the initial results against DDoS attacks with recall 97%–98% and f1 98% to 99%. In this extension we compare R-IDPS with SNORT, the results show a 4% improvement in terms of detection accuracy. Also, the IoT network traffic latency has been analyzed as compared to R-IDPS and SNORT. R-IDPS performs efficiently under the DDoS attack with minimum regular traffic latency and provision.

A research work [6] uses a testbed and uses a different kind of malicious traffic to compare the Snort and Suricata that is later collected by the Metasploit framework. All this experiment concludes to the fact that the Suricata require more resources like memory and central processing unit (CPU) to process the malicious traffic and detect the attack as compared to the Snort. Also, the testbed outcomes show that the Suricata can detect only four attacks as compared to the six attacks detection out of seven by the Snort. Further, the Suricata false positive rate is 74.3% false-positive rate (FPR) as compared to the 55.2% FPR in the case of Snort. Also, with the adoption of machine learning and fuzzy logic, this rate comes down to 16.9% FPR and 8.6% FPR respectively, in a hybrid approach. The experiment concludes that machine learning and signature-based approaches produce better results and are more effective for the detection of attacks. But implementing these systems in the resource constraint devices is still a challenge.

Based on blockchain technology, some research is developing IDPS solutions for IoT devices. As [14] shows, a collaborative IDS system based on a blockchain employs the essential notion of network node trust development. Another study [15] proposes an IDS system that detects ping of death assaults using an analytical approach based on packet length. However, energy consumption is one of the challenges with IoT devices. One study [16] supports the idea of nodes cooperating to identify an attack, thereby sharing the load and reducing energy consumption. There's a study [17] that employs evolutionary algorithms for low-cost attack detection in energy-constrained devices.

As this study [18] develops IDS, Passban, employing low-cost gateways, and using edge computing techniques for performance enhancement, anomaly detection is one of the important strategies for detecting assaults. For attack detection, however, this paper [19] use a specification heuristics technique. This method is effective at the device level. The agent-based intrusion detection system is proposed in one of the studies [20]. (AIDS). The system's fundamental idea is based on the chi-square statistical approach. This research reveals that DDoS attacks can be detected with a high degree of precision. The IDS system is implemented by AIDS using traditional technology. In our research, however, we're employing an SDN core network to enable real-time intrusion detection and mitigation.

3 Real Time SDN based IDPS

Fig. 1 depicts the system design for the proposed R-IDPS. The system is made up of seven different components: feature collector store (FCS), feature extractor agents (FEA), intrusion detection module (IDM), attack identification module (AIM), intrusion prevention module (IPM), analyzer module (AM), report and alert module (RAM). Agents submit network traffic statistics to the highlighted collector on a regular basis, and the IDM monitors the statistics for the entire network from there. The IDM keeps track of the typical MQTT traffic generated by IoT devices. The MQTT broker receives data from the IoT devices. The MQTT subscriber, on the other hand, receives data from the broker when needed. In the network, there is also a hacker testbed.

These bots try to flood the network with the packets using an ICMP flood attack to cause the DDoS attack in the network. As soon as the IDM detects these kinds of attacks, it marks such devices as suspicious devices and duplicates the traffic of these devices to the Analyzer module. This module further identifies the type of attack using the identification module. After successful identification the attack information is shared with the alarm and report module, this module raises the alarm in the system and logs it. At the same time, the attacker source and type are communicated to the mitigation module for prompt action to block the attacker from the network. For this purpose, the mitigation module updates the OpenFlow rules of the respective switches involved in the attack flows.

3.1 Feature Extractor Agent (FEA)

This module's rule is to keep track of real time network traffic. The agent is a program that extracts specific characteristics from stored network traffic. Tab. 1 contains a list of these features. The features are collected from the host on a regular basis by the module. The timer is set to 10 s by default. The time can be changed according to the system's requirements. The module delivers its output that is whole feature list to the feature processor which is called feature collector module after receiving the appropriate features data. The hypertext transfer protocol (HTTP) protocol is used to communicate between the agent and the collector.

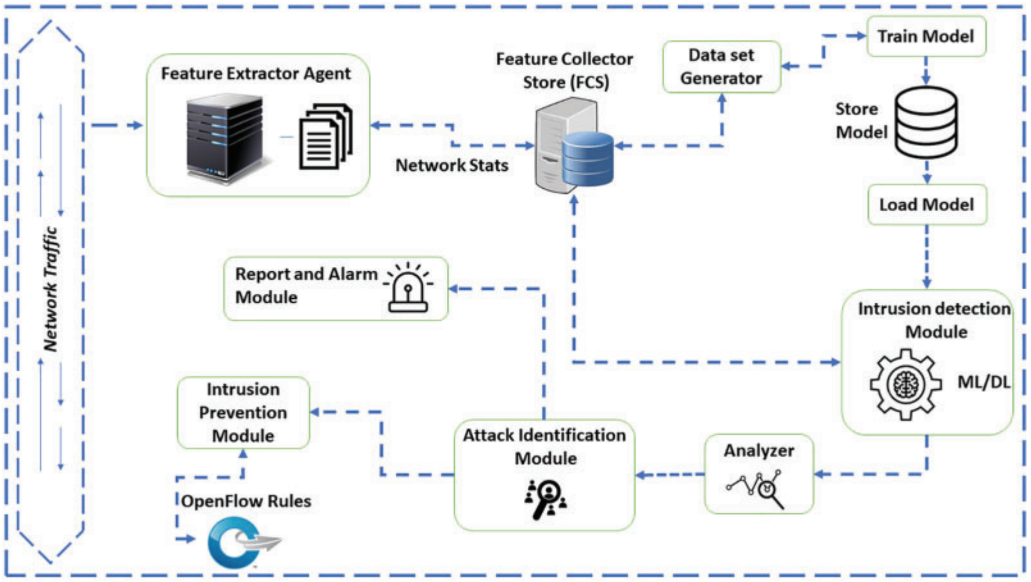


Figure 1: R-IDPS flow-based architecture

Table 1: Dataset features

Type	Feature ID	Description
Packet statistics	In echoes request	Total count of echo requests
	Out destination un-reach requests	Number requests unreachable to the destinations
	Input messages	Total number of data input messages
	Output messages	Total number of data output messages
	Out echo response	Total count of outgoing echo messages responses count of outgoing messages requests
	Outgoing echoes	Calculated mean of consecutive received packets
	Mean of packets	Delta change in the number of consecutive packets
	Change in packets	Total number of packets for a particular destination
	Destination address	Total number of TCP SYN request packets received
	Number SYN	Total count of echo requests
Session statistics	In timestamp	Input packet time stamp output packet timestamp
	Out timestamp	Total active connection at one time

(Continued)

Table 1: Continued

Type	Feature ID	Description
	Active session	Count of unsuccessful sessions at one time
	Invalid session	

3.2 Feature Collector Store (FCS)

This module's primary function is to operate as a repository for real time network traffic. This is a time series database that keeps track of all the records and their timestamps. This repository is updated on a regular basis by all network agents. The tables in this repository include the entire list of features that the agents retrieved and delivered. The agent identity data is also maintained in the store, in addition to this information.

3.3 Intrusion Detection Module (IDM)

Detecting the anomaly within the system is one of the most significant components of detection systems. There are a variety of approaches for identifying anomalies in network data, including statistics methods such as the chi-square method. However, these methods have significant drawbacks, such as sample size restrictions and difficulties in interpreting when the dependent or independent variables contain a comparatively greater number of categories (20 or more) [21].

Algorithm 1: Detect Anomaly

```
Input: Anomaly flag
Output: New OpenFlow rule
$database = Connect(DB\,IP, user\,ID, user\,pwd)$
$SVM\,model = loadModel$
    while $timer \leq 10$ do
        $Sleep$
        $query\,result = database.query\,()$
          if $query\,result != null$
          then
             for $query\,result\,[]$
             do $featureSet\,[] = record.getdData\,()$
             end for
             $IsAnomaly = svm\,inst.predict\,(featureSet\,[])$
             return $IsAnomaly$
             end if
```

Machine learning, on the other hand, is proven to be far more adaptable and futuristic. For this module, we utilize the support vector machine (SVM) machine learning approach to discover anomalies in the dataset. The FCS provides the data to IDM. It runs the training data, to train the SVM model. The model is stored after successful training. The IDM receives the current traffic data from the FCS in real time. It uses the loader module to load the model. After examining the data, the SVM predicts anomalies in the data. If an anomaly is discovered, the attack flows and source information are provided to the analyzer module. The system works normal in case of no anomaly.

3.4 Analyzer Module (AM)

The analyzer module is turned on after a successful detection. This module's aim is to notify the network of a potential assault. It flags the core network devices as suspicious as a result of this. The next process is to notify the attack identification module, this will help to determine the attack type.

3.5 Attack Identification Module (AIM)

An identifying module is used by the AIM to clearly identify the attack (IM). Each of the core switches has an IM associated to it. The IM's rules are preconfigured for potential attacks as well as attacks that are now undetectable. This module's primary function is to duplicate current network traffic and communicate it to an identification module.

Algorithm 2: Analyzing the suspicious devices

```
Input: Set of suspicious devices
Output: compromised device ID
$in - dev - set = Getthesus\,[DeviceIndex]$
  if $in - dev - set \neq null$
      then
            call $analyzer\,()$
            $dev\,status = "INSPECT"$
            $mirrortraffic\,()$
            $log\ the\ dev\ links$
            call $mitigation\ module$
                while $timer \leq 10$ do
                      Sleep
                devstatus = "EDGE"
```

Simultaneously, we must duplicate network traffic in order to provide network data to the IM. The notion of port mirroring is utilized to send current network traffic in parallel to the identification modules for this purpose. This will help to guarantee that routine network traffic runs smoothly. The AIM compares network flows to signatures that have been pre-configured. When a match is detected, the module sends out an alert. The RAM and the IPM are both notified of the alarm.

3.6 Intrusion Prevention Module (IPM)

Following the effective detection of the attack. The attack information is communicated by AIM to the mitigation module. The information includes information on the attacker and the attacker node. The intrusion prevention module locates all network traffic links between the attacker node and the rest of the network initially. The attacker node's core and edge switches are then determined by the module. The module provides new rules for the suspect node based on the type of attack after enlisting the attached devices. Either the action involves dropping packets or entirely blocking the node in this case. The revised rules are then applied to the appropriate switches.

Algorithm 3: Attack Mitigation

```
Input: List of compromised device
Output: New openflow rule
if $dev \neq null$
        then
            listofhosts = getallhosts [dev]
            path = networktopology [host]
            if path < path [host]
              then
                find nearest switch
                create new OpenFlow rules
Upload new rules to the openflow switches
                Action = "drop"
```

3.7 Report and Alarm Module (RAM)

The system should inform the system operators about the all the possible harmful activity that is occurring. This will help the system administrator to take further safeguards and amend required changes in the network's overall entrance and operational regulations. After collecting suspicious data, the module sends the entire attack report to the dashboard (software application, web-based). The data displays the type of assault, when it happened, where it came from, and what action was taken at the spot. This data is also saved in the system for future reference.

Algorithm 4: Update alarms

```
Input: Alarm packet
Output: Send alarm to ONOS
  RawAlarm = alarmpacket
   if RawAlarm ≠ null
            then
            NewAlarm = StructureAlarm (RawAlarm)
                Alarm List [] + = NewAlarm
                    Call ONOS Faultmanagment Servicel
                    Faultmanagment process = Alarm List []
                    Display Alarm
                    log Alarm
```

4 Experimental Testbed for Performance Assessment

A specialized testing environment (i.e., an SDN-based IoT testbed) is required to appropriately analyze the performance efficiency of the proposed IDS in terms of threat detection accuracy and computational power. Indeed, an accurate measurement campaign necessitates the availability of appropriate datasets including evidence of various sorts of hazards typically seen in real-world IoT deployments. We put up a real SDN-based IoT testbed to achieve this goal. This testbed is then deployed in two separate situations, with real network traffic being gathered in each scenario to provide several datasets that are subsequently used to thoroughly analyze the proposed IDS. In this section, we'll discuss the testbed setup as well as the two special testcases scenarios. Later we will describe the major aspects of the developed dataset, further the attacks details will be described that we carried out to create our test setup. Finally, we'll explain thoroughly the basic building blocks of the complete system software development.

4.1 Testbed Setup

WiFi Mininet has been used for network simulation running on an Ubuntu operating system to deploy a typical SDN-based testbed for IoT network scenarios. A tool like this produces IoT device nodes with a cloud backend. For our experimental setup, we regard simulated IoT devices running Ubuntu Core as an IoT operating system. The architecture of our testbed is depicted in Fig. 2. In a nutshell, the sensors used in python scripts communicate using WiFi technology. MQTT has been used for data transfer between IoT sensors and subscribers. The publisher nodes are the IoT nodes for this purpose while the subscriber nodes are the standard nodes registered with the MQTT broker. The network subscriber nodes, which are conventional hosts running the Ubuntu operating system that operate as subscribers attach to the broker's services via MQTT broker. The IoT device has the agent installed. These agents are installed on every other server or node that provides a service, to monitor its activity.

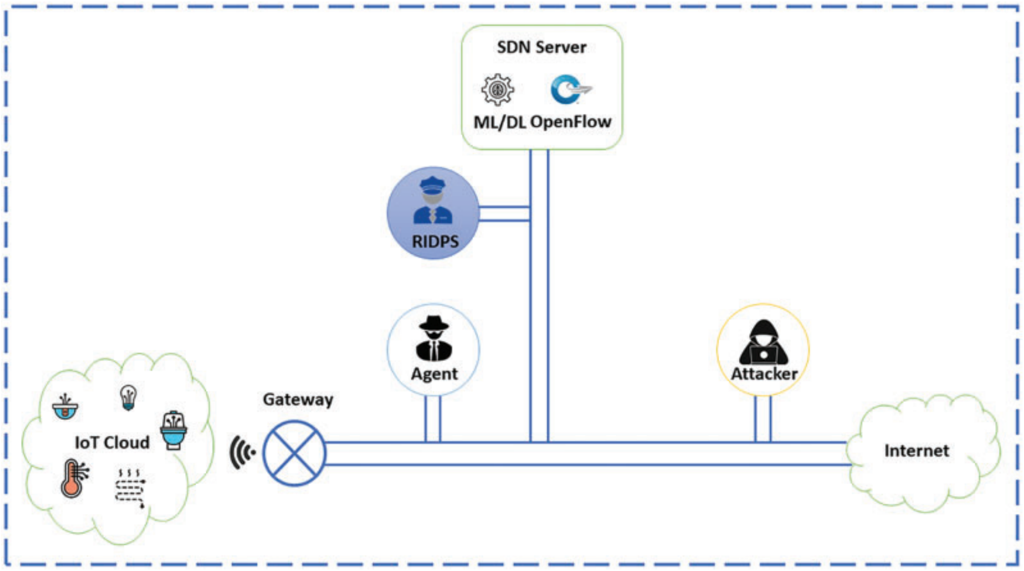


Figure 2: Testbed architecture

The network's core is made up of SDN-based servers and open flow switches. The R-IDPS is installed as an application module on the SDN server. The service initiated automatically on the SDN server startup. Furthermore, a database repository has been installed in the network. Any standard node can host the repository, in our case we install it on the independent host running a conventional ubuntu operating system. The repository contains a time series database, it stores the records using real time timestamps. This database serves as a repository for real-time network flows, that helps in mining the device features based on its network flows. The system's agents periodically send the network flows to this repository any deviation in the features data will help the system to detect the anomaly. Further, we use this repository to store the agent's information, IP address, time started or stopped and current status.

A simple Ubuntu host is used to establish the attack configuration. The hacking scenario was created using different hacking tools like hping3, Nmap, and other network monitoring tools especially Wireshark installed on the node. The main task for attacker is to disable network services, to accomplish this the attacker targeted the server and IoT device. DDoS assaults using the tools has been used by the attacker to launch the attacks. In this study, we look at two different types of DDoS attacks: ICMP flood attacks and TCP SYN attacks. For each attack, we devised two scenarios, as detailed below.

4.2 ICMP Flood Attack Scenario

We design a network configuration as illustrated in Fig. 3 to generate a scenario of ICMP flood attack. We established two IoT clouds with IoT sensors in this architecture. The server and other network nodes are connected to these IoT devices. Subscribers (ss∗), bots (bot∗), databases (db), and MQTT brokers (brk∗) are the nodes that make up the network. The MQTT protocol is used by the IoT device to update data using a publish-subscribe architecture. In this case, a MQTT broker server has been installed in the system. The IoT nodes attached to the sensors become publishers, so that they can send the sensor data to the server, while the nodes that need the sensor data for monitoring and data analysis become subscribers to the server.

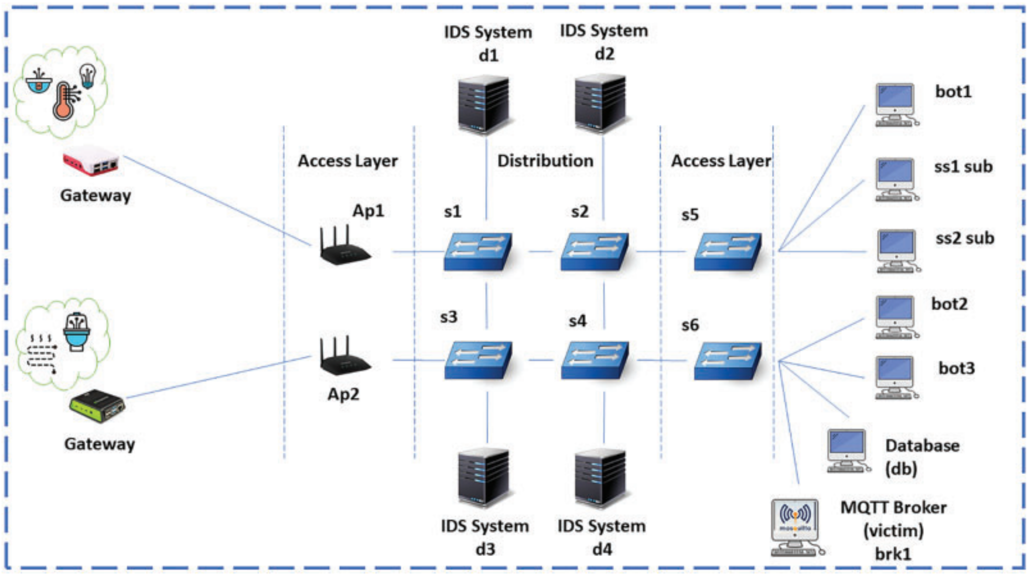


Figure 3: Mininet ICMP flood attack topology

The core of the network has been developed to mimic the normal ISP network setup. It has core distribution switches and edge switches. We use four distribution switches connected in a ring topology. Each distribution switch is attached to an attack identification module (AIM). AIM is hidden from the other network nodes to secure it from attackers. Further, the distribution switches are connected to four edge switches. All the host nodes are attached to the edge switches. Two of the edge switches contain built-in wireless routers to communicate with IoT networks, while the other two are normal OpenFlow switches. These edge switches are configured in a star topology with hosts. The host includes the IoT cloud, database repository, MQTT broker, subscribers, and bots.

The regular traffic python script is used to create the normal traffic scenario. This script feeds sensor data to the MQTT broker on a regular basis. The data is updated for MQTT subscribers after it is received. The main goal of the hacker is to bring down this service by attacking either the IoT device or the MQTT broker server. In this case, we've deployed our agent on the IoT device, and the bots are attempting an ICMP flood attack against it. To attack the bots, use the hping3 tool to carry out the appropriate attack, as shown in Fig. 4.

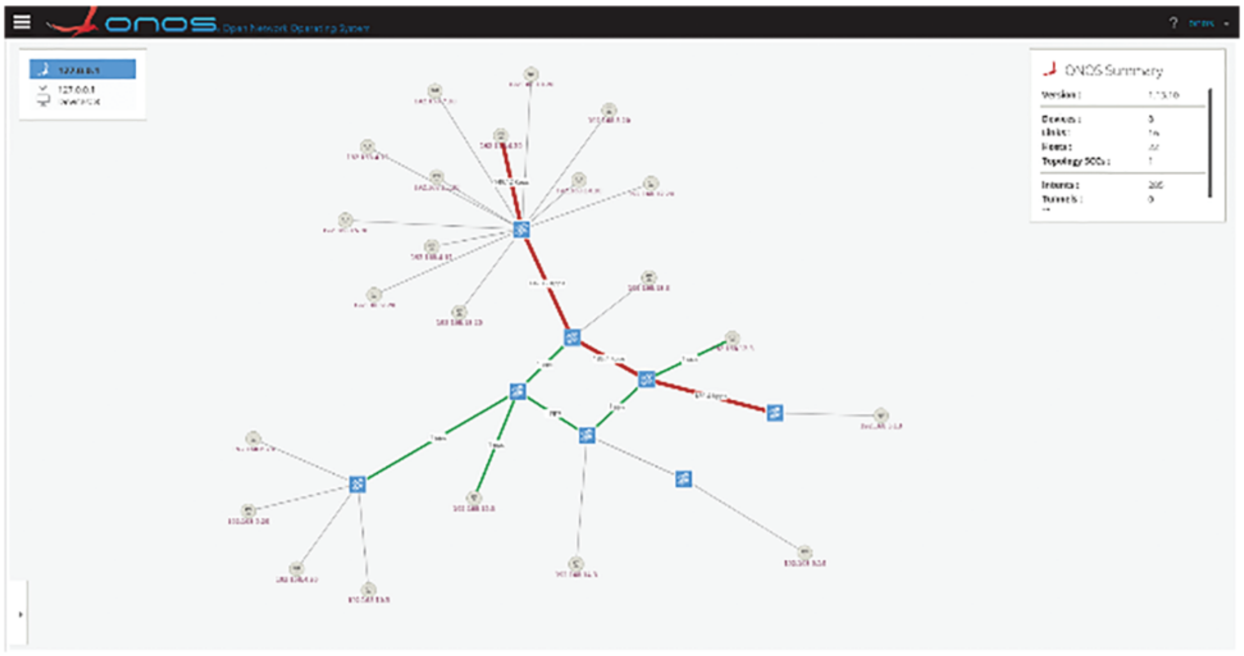


Figure 4: ONOS ICMP flood attack display

The network metrics are periodically sent to the database server by the agent deployed on the IoT device. The database server receives the updated network information as soon as the attack is begun. The machine learning algorithm used by the monitoring module installed on the ONOS server to detect the attack is support vector machine in our case. The network flows are sent to the attack identification module, using mirroring, after detection. This module determines the sort of attack and the origin of the attack. This information is forwarded to the mitigation module, which is a server-side application that processes the attack and modifies network rules. To prohibit the hacked host, these new OpenFlow rules are deployed at the corresponding OpenFlow switches.

4.3 TCP SYN Attack Scenario

For this type of attack, we use the same network topology and configuration as shown in Fig. 5. But we modify some nodes and communication patterns as the attack we are discussing uses a different protocol and that is TCP. An SYN flood is a type of denial-of-service attack, the attacker quickly starts TCP connection but does not finish a connection with a server. Waiting for half-opened connections consumes server resources, potentially making the system unusable to real traffic [22].

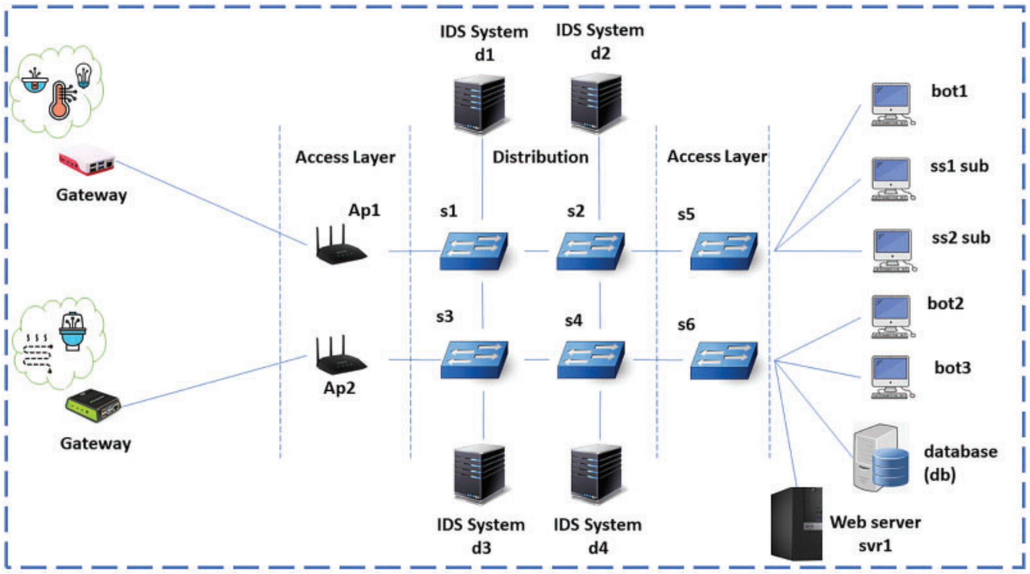
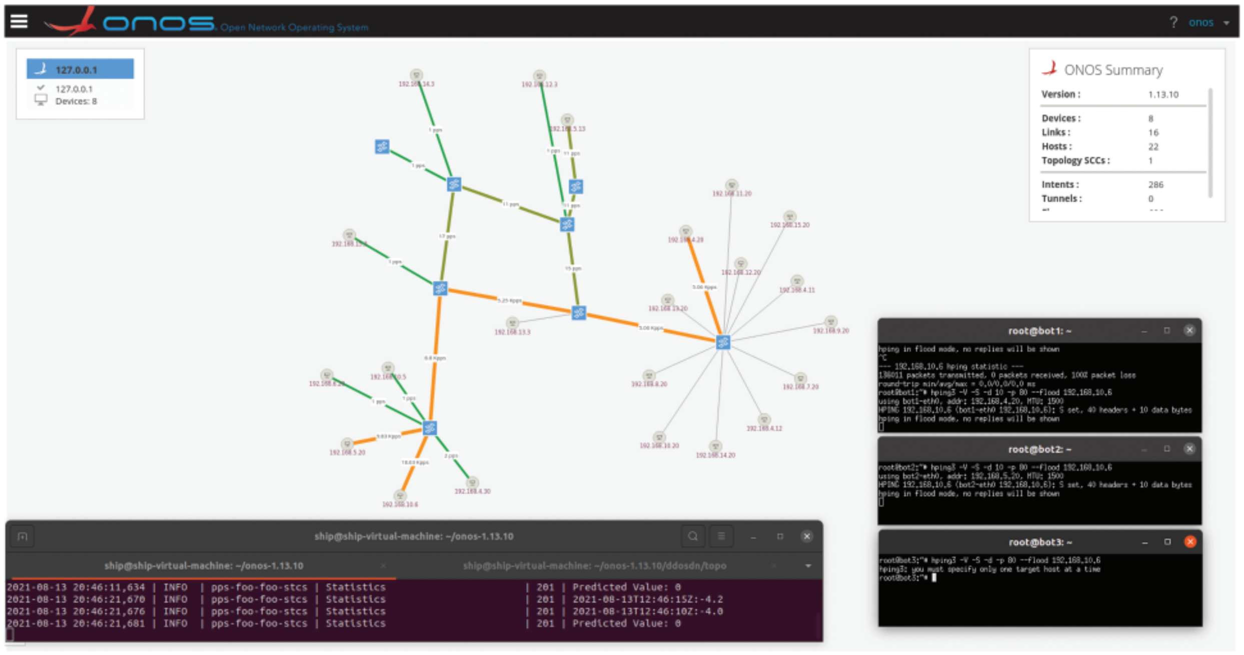


Figure 5: Mininet TCP SYN attack scenario

An SYN flood attack is basically an incomplete connection, works by not giving the server the required acknowledgment (ACK) code. The malicious client node can force the server to send the SYN-ACK for a fake connection to a victim IP address, the victim will not send an ACK as it never delivered a SYN request. As the regular network congestion could be easily achieved using missing ACK, and the server keep waiting for the ACK. In an attack, the malicious client's creates thousands of half-open connections, this will bind server resources. As the number of such connection increases exponentially made the server resources wear out quickly. After that, the server is unable to connect to any clients, this results into a denial-of-service attack to even a genuine clients request. Denial of service is just one aspect of this attack, the server under such attack can even crash or malfunction in the network. This possesses much bigger threat to the overall network and service provision.

On the server side, we established a web server to create this type of assault. To place the server under TCP-based traffic strain, the IoT cloud employs a tool. It does it by requesting a JavaScript object notation (JSON) script-based resource from the server using a tool called siege. HTTP requests and responses are used in this communication, which necessitates the use of a TCP connection. The server is the target of the attacker. Especially the webserver that listens on port 80 for requests. The attacker takes advantage of the TCP protocol's three-handshake procedure. It uses the hping3 program to flood the server with SYN customized requests, the request can be modified using this toll like the size of the packet and the packet frequency etc., without sending the ACK back to the server, as shown in Fig. 6. The server is left with countless half opened connections that consumes almost all the server resources. Eventually, the server stops responding to even genuine service requests from IoT cloud nodes.



root@bot1: ~
```
hping in flood mode, no replies will be shown
^C
--- 192.168.10.6 hping statistic ---
138011 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@bot1:~# hping3 -V -S -d 10 -p 80 --flood 192.168.10.6
using bot1-eth0, addr: 192.168.4.20, MTU: 1500
HPING 192.168.10.6 (bot1-eth0 192.168.10.6): S set, 40 headers + 10 data bytes
hping in flood mode, no replies will be shown
```

root@bot2: ~
```
root@bot2:~# hping3 -V -S -d 10 -p 80 --flood 192.168.10.6
using bot2-eth0, addr: 192.168.5.20, MTU: 1500
HPING 192.168.10.6 (bot2-eth0 192.168.10.6): S set, 40 headers + 10 data bytes
hping in flood mode, no replies will be shown
```

root@bot3: ~
```
root@bot3:~# hping3 -V -S -d -p 80 --flood 192.168.10.6
hping3: you must specify only one target host at a time
root@bot3:~#
```

ship@ship-virtual-machine: ~/onos-1.13.10
```
2021-08-13 20:46:11,634 | INFO  | pps-foo-foo-stcs | Statistics                    | 201 | Predicted Value: 0
2021-08-13 20:46:21,670 | INFO  | pps-foo-foo-stcs | Statistics                    | 201 | 2021-08-13T12:46:15Z:-4.2
2021-08-13 20:46:21,676 | INFO  | pps-foo-foo-stcs | Statistics                    | 201 | 2021-08-13T12:46:10Z:-4.0
2021-08-13 20:46:21,681 | INFO  | pps-foo-foo-stcs | Statistics                    | 201 | Predicted Value: 0
```

Figure 6: ONOS TCP SYN attack display

5 Dataset Construction

There are publicly available IoT datasets in the literature, such as those utilized in [23–28]. These datasets, however, were unable to precisely fulfill our criteria. First and foremost, they are associated with extremely specialized attacks (e.g., only Botnet, only Telnet, and so on). On the other side, our goal is to develop an IDS that can detect a wider range of threats aimed at IoT devices. We also need a scalable dataset that will allow us to include fresh assaults in a range of settings. As a result, we chose to build our dataset from network traffic generated by our internal testbed. The latter can be set up to collect network traffic and then extract some widely accepted properties. Instead of focusing on packet flow, traffic flow timings, and data flow statistics throughout a certain period, as indicated in Tab. 1, we avoid examining aspects that are static to the environment (e.g., source or destination IP addresses and source/destination port numbers).

Figs. 3 and 5 show the distribution of attack flows, or hostile network flows, in the gathered traffic for this scenario. The number of attack flows against benign flows is detailed in Tab. 2. This dataset's benign traffic consists of 690 network flows that are used to simulate the system's normal operation. The regular and malicious network flows in ICMP flood attacks on IoT devices are 100 and 67, respectively. The datasets are built by mixing a variety of benign and malicious network traffic to adopt a practical approach. Finally, the SYN flood attack has 43 attack flows and 108 normal network flows.

Table 2: Traffic flows for the dataset construction

Traffic type	Number of attack flows	Number of normal flows
Regular network traffic	0	690
ICMP	67	100
TCP SYN	43	108

6 Results and Discussion

This section describes and analyze R-IDPS findings using machine learning-related metrics (see Section 4.2) and hardware resource use in second scenarios (see Section 4.3).

6.1 Performance Evaluation and Analysis Based on Machine Learning

We start with the standard terms of a confusion matrix, namely true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), remembering that a TP happens when R-IDPS properly detects an attack in our tests (i.e., a hit). When R-IDPS successfully detects a normal flow (i.e., a correct rejection), it generates an FP when it detects an attack by mistake (i.e., a false alarm), and a FN when it fails to detect an assault (i.e., a miss). The number of flows used for testing is listed in Tab. 3, with 450 benign flows and 96 attack flows for scenario 1. While in scenario 2 the normal flows and attack flows are 696 and 194 respectively.

Table 3: Accuracy measurement

Attack	Technique	Normal	Attack	FP	TP	FN	TN	Precision	Recall	F1
TCP SYN flood	SVM	696	194	0	190	4	696	100	0.97	0.98
ICMP flood	SVM	450	96	0	95	1	450	100	0.98	0.99

It's also worth noting that R-IDPS is a system for detecting anomalies. An anomaly is defined as a rare occurrence when compared to usual occurrences, regardless of domain. In all these circumstances, the terms of the confusion matrix are often combined to compute Precision (P) Eq. (1) and Recall (R) Eq. (2), and they are harmonic mean, i.e., the F-measure, also known as F1-score Eq. (3). Finally, keep in mind that FP is still an expressive performance indicator while evaluating an IDS. As a result, reducing FP is still one of R-IDPS' core objectives. This is equivalent of saying that a high accuracy anomaly detection system should have greater P values (P value is a statistic that tells how many anomalies were accurately discovered) in terms of Tab. 3 aggregated metrics. The results data shows high precision values of R-IDPS against flood attacks in the context of TCP and ICMP attacks.

$$precision = \frac{TP}{TP + FP} \times 100 \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{2}$$

$$F1 = 2 \times \left[\frac{Precision \times Recall}{Precision + Recall}\right] \tag{3}$$

The performance indicators for scenario 1 and scenario 2 are shown in Tab. 3. organized by kind of attack. The best values for each group and each aggregated statistic are bolded there to improve readability.

In first Scenario, SVM detects all the attacks with acceptable accuracy. More specifically, as illustrated in Fig. 7, SVM operates best when the number of attack flows is minimal. The detection rate decreases as the number of attacks increases. However, the performance ratings for both precision and recall are always at their highest levels, with F1 at 99% and recall at 98% for ICMP flood. However, as the quantity of flows increased, the system became overworked, and we began to see more FN. We put up a webserver in second scenario to create a SYN flood attack. In this test setup the bots launch attack on the server, and we use the siege tool to produce a lot of server traffic, putting the network under strain. In this case, machine learning model plays an important role in detecting the attacking flows like SYN attack. In this case, the results are the same, when we increase the number of bots automatically the attack flows increased as a result, this imparts over the performance of the IDS and IDS performance decreased. In the standard attack situation, though, the recall is 0.97 and F1 is 0.98, which is shows the high effectiveness of the system.

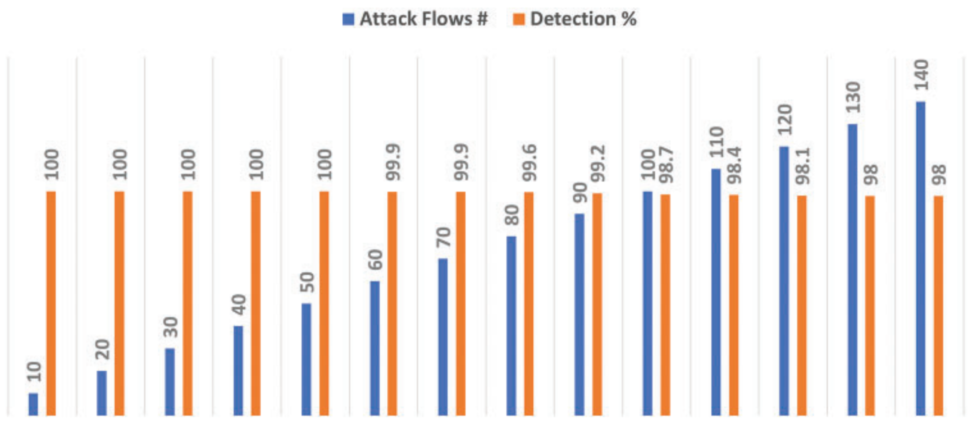


Figure 7: Accuracy detection percentage

The location of R-IDPS is very important, if we place the R-IDPS at the switch the overall performance of the system improves comprehensively especially in the scenario of network traffic load stress during the flood attack. This deployment scenarios are our future research goal, further we would like to see the comparison of edge vs centralized R-IDPS deployment in terms of performance and accuracy improvement and their impact on the results.

6.2 Resource Utilization Analysis

Machine learning-related metrics are only one side of the coin. We're also looking into the possibility of deploying R-IDPS on the edge, as previously stated. As a result, in this section, we report our findings in terms of the hardware resources. The resources required by an IoT device in order to operate R-IDPS needs to be analyzed. Therefore, we'll focus on Scenario one, in this we will use agent at the IoT devices on the edge and figure out the resource consumption by measuring the CPU load, memory utilization. Further we also analyze the network throughput during normal traffic and at the time of attack. Figs. 8 and 9 show the memory utilization, CPU load and network throughput of the system. We took the values in both scenarios first with the agent installed and without agent.

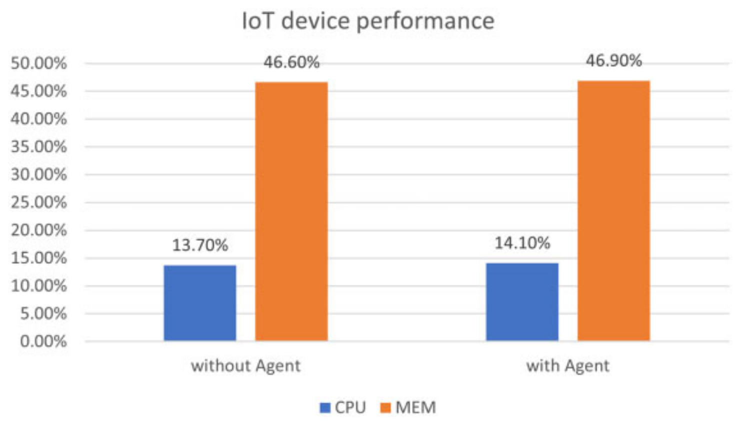


Figure 8: IoT device resource utilization

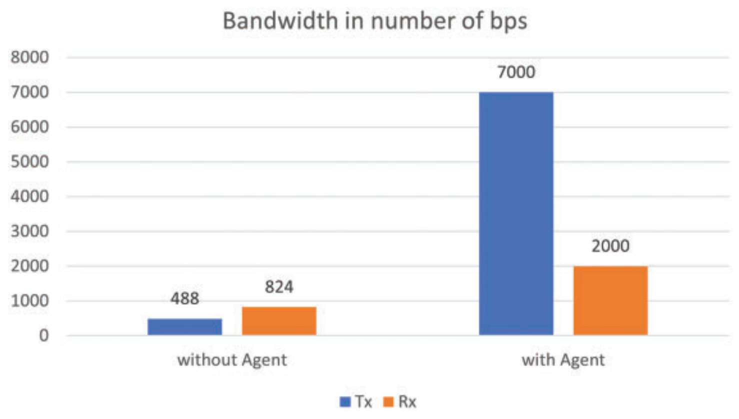


Figure 9: IoT device bandwidth utilization

The R-IDPS agent's resource requirements are limited, according to our experiment. In comparison to other commercially accessible options. The memory consumption of an IoT device is normally 46.60%, but when we activate the agent, it rises to 46.90%. The agent's actual memory consumption is merely 0.30%. Similarly, if we look at the CPU load, we can see that the typical load is 13.70%, but when we run the agent, it jumps to 14.10%. It's only a 0.40% gain. However, bandwidth is a bigger problem for us because it might impair the overall network performance. In a typical situation, the IoT gadget consumes approximately 0.8 kbps of network bandwidth. Only 6.2 kbps are contributed by the activated agent, and the total network utilization becomes 7 kbps.

6.3 Comparing R-IDPS and Snort under Load Test

The R-IDPS is put under test. To validate the results, the framework is compared to the more conventional SNORT-based framework. In the first stage, the number of attacks increased stepwise. In the first step, we use 30 bots and tested R-IDPS the results are shown in Fig. 11. Detection is 100%. Then we increase the number of bot attacks to 60, this time the detection rate is 98%. Finally, we tested the framework with 90 bots and the detection rate is 94%. With the stats, it is hard to conclude the performance of the proposed framework. So, we created the same environment and put SNORT in this environment without the R-IDPS support. Now the same attacks were performed, and the results are shown in Fig. 10. For 30 bots it's 100%, for 60 bots its 100% but surprisingly for 90 bots, it falls to 90%. As in Fig. 12, we compare the results and found that R-IDPS start to perform well under load and high volume of attack scenario almost 4% better to SNORT only solution.

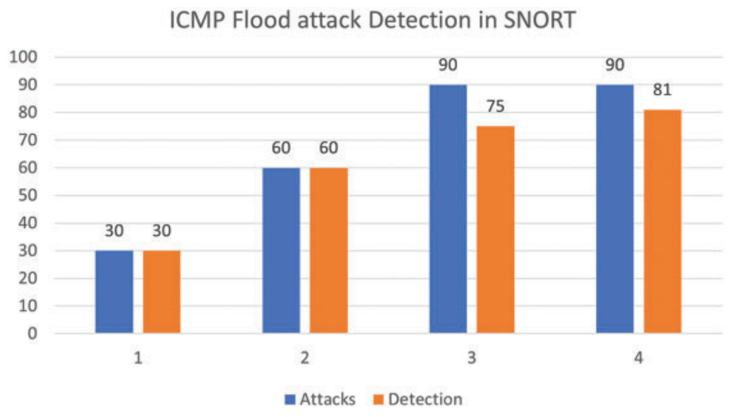


Figure 10: Attack detection SNORT

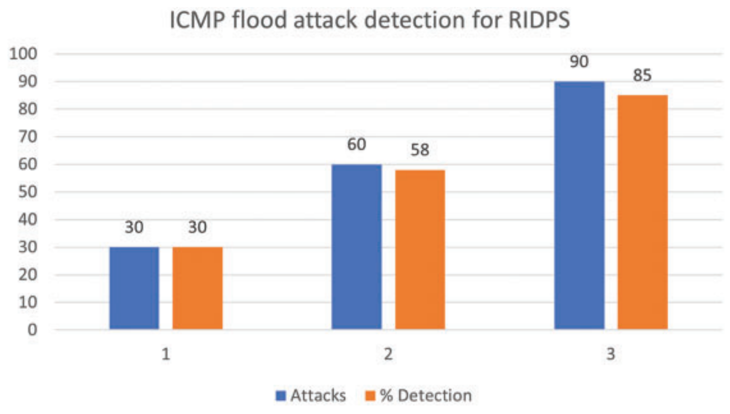


Figure 11: Attack detection R-IDPS

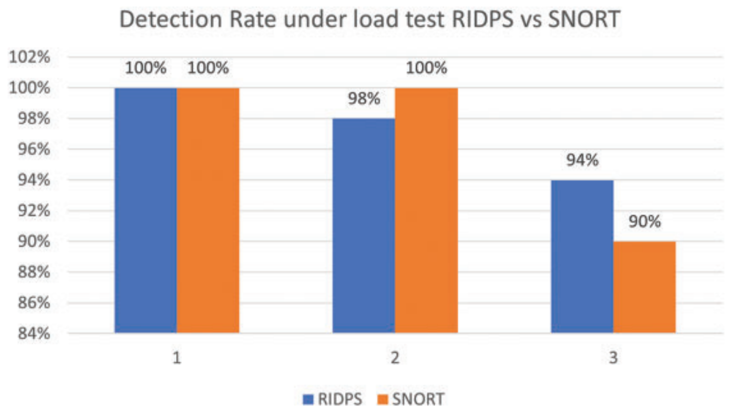


Figure 12: Detection rate under load test for R-IDPS and SNORT

6.4 Network Traffic Latency During the Attack

One of the important aspects is to disruption of services in the IoT network due to volumetric attacks, especially at the network layer. The DDoS attacks can easily exhaust the network resources even before detected and blocked from the network. The reason for this situation is the placement of the IDPS node within the network. If the detection system is placed not in line with the external traffic, then the attack flows will easily pass through the core network devices before they reach the detection systems. This will create resource exhaustion at the core switches resulting in total service disruption and sometimes service failure. In this context, the IDPS system is placed in line with the external network traffic. This topology creates other issues like the detection systems are not aware of internal attacks and malicious nodes.

Keeping this in context we test the R-IDPS, the proposed framework works at the device level and blocks the malicious traffic at the edge switches. This helps the network retain its services as normal even in the scenario of severe DDoS attacks. As we see in Fig. 13 first snort is used and tested under DDoS attack. The normal traffic affected badly normal latency is kept to two seconds but during an attack it too 54 s of delay and sometimes the sessions completely disconnected before the attack is identified. However, when the same scenario is tested under R-IDPS as shown in Fig. 14, the normal traffic took 2 s and during the DDoS attacks, it is delayed to 8 s without breaking any session of service. These results display the efficiency and robustness of R-IDPS as compared to the conventional frameworks.

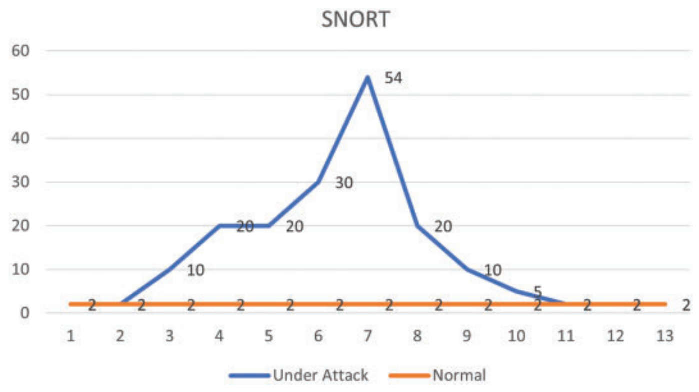


Figure 13: Network traffic latency SNORT

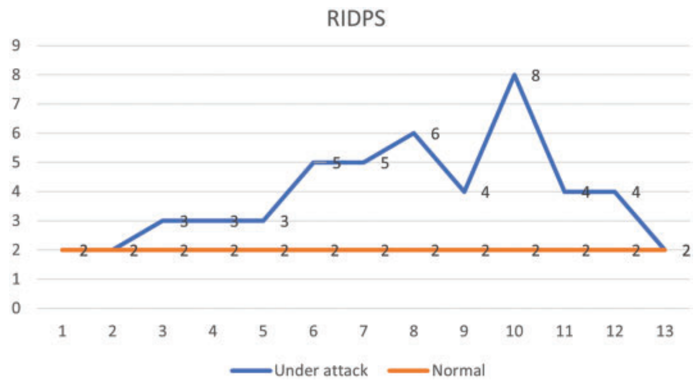


Figure 14: Network traffic latency R-IDPS

7 Conclusion

In this paper, the R-IDPS uses the power of SDN to make IoT intrusion detection and prevention completely autonomous. The interesting part of the research is performance evaluation of the centralized IDPS system vs distributed IDPS systems in the context of IoT networks. The integration of agents, SDN and machine learning techniques was a challenge, but the results show the power of distributed design in terms of effectiveness and performance. The study finds that the SVM uses fewer resources and is quite suitable for low-power devices especially like IoT devices. According to the findings, using an agent-based system improves overall system performance while maximizing the host's limited resources. Furthermore, the system's internal dataset and online model training allow it to scale up swiftly. Further the proposed system design protects IoT devices against impending threats and network services, without being down for long periods of time. The detection performance of the system against the DDoS attacks shows excellent precision and low false-positive and false-negative rates. The proposed design is customizable and can be used in a variety of situations. Machine learning will be employed in future studies not just for attack detection but also for attack identification scenarios at the network edges, along with the performance analysis of different machine learning models in the context of IoT devices.

Funding Statement: The authors would like to express their gratitude to the sponsor, as this research was funded by the University of Malaya in Malaysia (GrantNo. GPF017D-2019).

Conflicts of Interest: The authors declare that they have no conflicts of interest to disclose regarding the current research.

References

[1] CISCO, "Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022," *Update*, vol. 2017, (accessed 2022), pp. 5–33, 2019.

[2] Spamhaus, "Spamhaus botnet threat update: Q2-2020," vol. 2020, pp. 15–40, 2020. [Online]. Available: https://www.spamhaus.org/news/article/800/(accessed2022).

[3] W. Haider, G. Creech, Y. Xie and J. Hu, "Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, pp. 29, 2016.

[4] I. Greenberg, "Cyber attack trends analysis report, volume1, Norton, MA, USA: CPS technologies, 2019. [Online]. Available: www.checkpointdirect.co.uk/media/downloads/check-point-2019-security-report-volume-1.pdf,(accessed2022).

[5] T. Zitta, M. Neruda and L. Vojtech, "The security of RFID readers with IDS/IPS solution using Raspberry Pi," in *18th Int. Carpathian Control Conf. (ICCC), 2017*, Sinaia, Romani, IEEE, pp. 316–320, 2017.

[6] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to Snort system," *Future Generation Computer Systems*, vol. 80, no. 3, pp. 157–170, 2018.

[7] A. Mansour, M. Azab, M. R. M. Rizk and M. Abdelazim, "Biologically-inspired SDN-based intrusion detection and prevention mechanism for heterogeneous IoT networks," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conf. (IEMCON)*, University of British Columbia, Vancouver, Canada, pp. 1120–1125, 2018.

[8] O. Flauzac, C. González, A. Hachani and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *IEEE 29th Int. Conf. on Advanced Information Networking and Applications Workshops*, Gwangju, Korea, IEEE, pp. 688–693, 2015.

[9] N. Mazhar, R. Salleh, M. Zeeshan, M. M. Hameed and N. Khan, "R-IDPS: Real time SDN based IDPS system for IoT security," in *IEEE 18th Int. Conf. on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, Karachi, Pakistan, IEEE, pp. 71–76, 2021.

[10] B. Caswell, J. Beale and A. Baker, "*Snort intrusion detection and prevention toolkit*," 1st ed., vol. 1, Syngress: Elsevier Inc, USA, pp. 31–67, 2007.

[11] OISF, "Suricata user guide," *Release 5.0.3*, Boston, MA 02115, USA: Open Information Security Foundation, pp. 3–120, 2019.

[12] S. S. Tirumala, H. Sathu and A. Sarrafzadeh, "Free and open source intrusion detection systems: A study," *Int. Conf. on Machine Learning and Cybernetics (ICMLC)*, IEEE, vol. 1, pp. 205–210, 2015.

[13] A. Sforzin, F. G. Mármol, M. Conti and J. M. Bohli, "RPiDS: Raspberry Pi IDS—A fruitful intrusion detection system for IoT," in *2016 Int. IEEE Conf. on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Toulouse, France, IEEE, pp. 440–448, 2016.

[14] G. D. Putra, V. Dedeoglu, S. S. Kanhere and R. Jurdak, "Towards scalable and trustworthy decentralized collaborative intrusion detection system for IoT," *arXiv*, vol. 2020, pp. 256–257, 2020.

[15] A. Abdollahi and M. Fathi, "An intrusion detection system on ping of death attacks in IoT networks," *Wireless Personal Communications*, vol. 112, no. 4, pp. 1–14, 2020.

[16] J. Arshad, M. A. Azad, M. M. Abdeltaif and K. Salah, "An intrusion detection framework for energy constrained IoT devices," *Mechanical Systems and Signal Processing*, vol. 136, no. 3, pp. 106436, 2020.

[17] C. Wu, Y. A. Liu, F. Wu, F. Liu, H. Lu *et al.,* "A hybrid intrusion detection system for IoT applications with constrained resources," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 12, no. 1, pp. 109–130, 2020.

[18] M. Eskandari, Z. H. Janjua, M. Vecchio and F. Antonelli, "Passban IDS: An intelligent anomaly based intrusion detection system for IoT edge devices," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020.

[19] M. J. Babu and A. R. Reddy, "SH-IDS: Specification heuristics based intrusion detection system for IoT networks," *Wireless Personal Communications*, vol. 112, no. 3, pp. 2023–2045, 2020.

[20] F. Y. Leu and I. L. Lin, "A DoS/DDoS attack detection system using chi-square statistic approach," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 2, pp. 41–51, 2010.

[21] R. Bohannon, "Chi-square limitations and alternatives," *Physical Therapy*, vol. 66, no. 6, pp. 1002, 1986.

[22] C. W. Pages, "*CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks*," 1st ed., USA: Carnegie Mellon University, pp. 119–127, 1996.

[23] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai *et al.,* "N-baiot—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[24] V. H. Bezerra, V. G. T. da Costa, S. B. Junior, R. S. Miani and B. B. Zarpelao, "One-class classification to detect botnets in IoT devices," in *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, SBC*, Natal, Brazil, pp. 43–56, 2018.

[25] Y. M. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama *et al.,* "IoTPOT: A novel honeypot for revealing current IoT threats," *Journal of Information Processing*, vol. 24, no. 3, pp. 522–533, 2016.

[26] R. Gopi, V. Sathiyamoorthi, S. Selvakumar, R. Manikandan, P. Chatterjee *et al.,* "Enhanced method of ANN based model for detection of DDoS attacks on multimedia internet of things," *Multimedia Tools and Applications*, vol. 31, no. 5, pp. 1–19, 2021.

[27] R. P. Nayak, S. Sethi, S. K. Bhoi, K. S. Sahoo, N. Jhanjhi *et al.,* "TBDDosa-MD: Trust-based DDoS misbehave detection approach in software-defined vehicular network (SDVN)," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3513–3529, 2021.

[28] J. Kaur, S. Ahmed, Y. Kumar, A. Alaboudi, N. Jhanjhi *et al.,* "Packet optimization of software defined network using lion optimization," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2617–2633, 2021.