systems

Article

Protection Schemes for DDoS, ARP Spoofing, and IP Fragmentation Attacks in Smart Factory

Tze Uei Chai [1,*], Hock Guan Goh [1], Soung-Yue Liew [1] and Vasaki Ponnusamy [2]

1 Faculty of Information and Communication Technology (FICT), Universiti Tunku Abdul Rahman, Kampar 31900, Malaysia
2 Higher Colleges of Technology, Fujairah P.O. Box 4114, United Arab Emirates
* Correspondence: tzeuei.c@1utar.my

Abstract: Industry Revolution 4.0 connects the Internet of Things (IoT) resource-constrained devices to Smart Factory solutions and delivers insights. As a result, a complex and dynamic network with a vulnerability inherited from the Internet becomes an attractive target for hackers to attack critical infrastructures. Therefore, this paper selects three potential attacks with the evaluation of the protections, namely (1) distributed denial of service (DDoS), (2) address resolution protocol (ARP) spoofing, and (3) Internet protocol (IP) fragmentation attacks. In the DDoS protection, the F1-score, accuracy, precision, and recall of the four-feature random forest with principal component analysis (RFPCA) model are 95.65%, 97%, 97.06%, and 94.29%, respectively. In the ARP spoofing, a batch processing method adopts the entropy calculated in the 20 s window with sensitivity to network abnormalities detection of various ARP spoofing scenarios involving victims' traffic. The detected attacker's MAC address is inserted in the block list to filter malicious traffic. The proposed protection in the IP fragmentation attack is implementing one-time code (OTC) and timestamp fields in the packet header. The simulation shows that the method detected 160 fake fragments from attackers among 2040 fragments.

Keywords: security threat; Internet of Things; smart factory; Industry 4.0; DDoS; ARP spoofing; IP fragmentation attack

check for updates

Citation: Chai, T.U.; Goh, H.G.; Liew, S.-Y.; Ponnusamy, V. Protection Schemes for DDoS, ARP Spoofing, and IP Fragmentation Attacks in Smart Factory. *Systems* **2023**, *11*, 211. https://doi.org/10.3390/systems11040211

Academic Editors: Tao Wang, Hanliang Fu and Zezhou Wu

Received: 15 February 2023
Revised: 14 April 2023
Accepted: 14 April 2023
Published: 20 April 2023

1. Introduction

Industry 4.0, called the Smart Factory, shown in Figure 1, is the phase of digitizing the manufacturing sector, which has been accelerated by exponential technologies and has vertical and horizontal integration across factory plants and supply chains [1]. Unlike the conventional sensor network, the IoT network is becoming more complex with heterogeneity that increases the connectivity and adopts standard communication protocols which link the physical components and receive commands through the Internet [2]. The revolution of Industry 4.0 has focused on flexibility and demand for customized products, and workshops are adaptive and scalable with a fast reconfiguration of machines. Cloud computing processes and shares data between cyber-physical systems (CPSs) [3]. A cyber-physical production system (CPPS) makes decentralized decisions in the production system in the virtual world [4]. In the Smart Factory, data are used in learning algorithms to support real-time intelligent and autonomously controlled systems [5,6].

The highly Interconnected and dynamically distributed environments also led to a high complexity that opened opportunities for attackers [7]. The security of the Smart Factory becomes even more important with the adoption of IoT in critical applications that can be diverted into a harmful situation to the entire system. In common practice, the security of the Cloud is handled only partially by service providers, mainly focusing on facilities and data centers [8]. At the same time, the customer needs to pay attention to interfaces and data that the customer might forget. A similar breach in one machine, network, or plant can persist in another as the concept of "digital production" shares the same Cloud settings [9]. There are unclear policies and best practices for a manufacturer to build IoT, such as protection with a strong password which can suffer from a brute-force attack [10]. The characteristics of sensor nodes that are low power, limited processing, and have memory constraints have prevented them from being embedded with secure and advanced protection algorithms. The IoT environment needs more security attention to avoid issues caused by misconfiguration [11]. For example, a UPnP (universal plug and play) service that automatically connects the local nodes may be misused by an attacker and should be restricted or disabled. Physical access to the network with a USB plug-in is simple and most straightforward for a hacker to penetrate the network system.

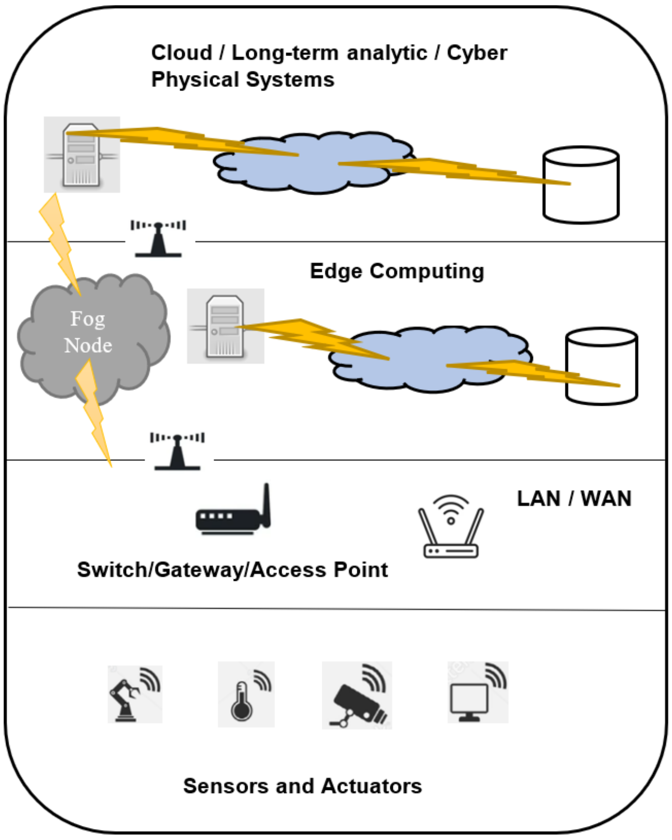
Cloud / Long-term analytic / Cyber Physical Systems
Edge Computing
Fog Node
LAN / WAN
Switch/Gateway/Access Point
Sensors and Actuators

Figure 1. A framework of a Smart Factory.

IoT runs software that requires security patches [12] and is sometimes late in reaching the devices through the Internet connection [13]. The attacker can exploit the zero-day attack on these devices by extracting relevant details from the Shodan search engine. Another security breach can occur when the updates come from an unauthorized or unauthenticated third-party application, even if the devices run in a closed-loop system. While in sensor networks, there are various supports for wireless communication such as radio frequency identification (RFID), Z-wave, and near-field communication (NFC), the mainstream connectivity options are WiFi and Bluetooth, which use the same frequency. These wireless signals are susceptible to interception, thus creating a vulnerability for the attacker to capture and exploit the traffic in the IoT network.

2. Related Works

The attacker launches the attack by stealing confidential data and bringing down critical infrastructure [13]. In the attack, the network can be redirected and mislead the internal nodes without knowing they are communicating with the intruders. By the time an active attack is launched, it is often too late and extensive resources are required to recover from the disastrous incident. The distributed denial of service (DDoS) attack aims to prevent the server from responding to a legitimate request using the easily exploited vulnerabilities of IoT devices [14]. The attack floods the target with excessive data leading to bandwidth depletion, resource exhaustion, and throughput downgrade [15]. The highly intensive attack requires many infected or compromised devices that turn into a botnet and are synchronized to launch the attack simultaneously, which is difficult to defend against [16].

The merge of wireless sensor networks (WSNs) with IoT technology can cause a lack of defenses which previously existed in the unattended nature of the networks [17]. Malware such as Mirai [18] targets IoT networks. Then the infected devices delete the binary file leaving itself undetected [19]. In addition, it is relatively easy to infect vulnerable devices if the breach remains even after several reboots. The compromised devices may send a small amount of data, and when it accumulates at the gateway it can become enormous. The situation can be escalated if the infected gateway participates in the flooding process, which is an Internet gateway with a larger bandwidth than the sensor [20]. The research evaluates the effectiveness of detection through learning-based algorithms to eliminate damage [21]. However, those classifiers require the model's training from the large dataset prepared from the network traffic of the IoT environment. IoT networks are heterogeneous and support sensor nodes with different sending behavior [22]. The dataset must be free from error and misleading records that generate inaccurate predictions. Some common classifiers used to detect DDoS are support vector machine (SVM) [23], K-nearest neighbors (KNN) [24], and random forest (RF) [25]. Those classifiers depend on datasets to establish the boundary between classes and make prediction outcomes.

ARP stands for address resolution protocol, a standard and common protocol that operates at the data link layer invoked in a local node to prepare for communication. To begin communication, the node must look for the ARP table to retrieve MAC mapping to send a packet. However, several challenges in the protocol are faced due to a lack of a reliable procedure to transmit the ARP packet. The receiver accepts all the ARP packets with their address and processes them in the ARP cache. The attacker can apply various scenarios using ARP request, ARP reply, or both. It can hide the attack in a high ARP traffic volume, making it difficult to detect. As in typical ARP spoofing, this fit for the attacker starts with screening the local network to acquire the valid node's IP addresses and learn about the topology. Another problem is that the attack can be adopted and followed with a denial of service (DoS) or man-in-the-middle (MITM) attack. ARP spoofing, sometimes referred to as ARP poisoning, has a timeout period in the ARP entries that requires the attacker to continuously send the spoofed ARP packet to stay connected with the targets. Protection for ARP spoofing with static ARP entries may increase the administrative overhead and requires higher processing power. Packet filtering is another method adopted in ARP protection that analyzes each ARP packet and filters those that are malicious. The authors of [26] propose detection based on windows by packet count and show the effectiveness of handling ARP spoofing. Another method of applying detection is checking every ARP packet header [27].

IoT integrates data with other IT assets to deliver various services and applications, which can be challenging for maintaining up-to-date physical parts [28]. Some of these IoT solutions, which are more awkward, require a complex setup for firmware patches. These are cases where communication from the server involves extensive, transmitted data, such as firmware updates, upgrades, or patches. The IoT has to be reprogrammed over-the-air (OTA), and large packets must be fragmented to send to the IoT network. The packet header identifier is an essential detail in a fragment which can be guessed through some of the methods acquired by the attacker, even with the TCP connection [29]. Meanwhile, avoiding fragmentation vulnerabilities by simply preventing fragmentation can cost more transmission sessions [30]. A study shows that the TCP protocol does not prevent an attack from adversaries that send a spoofed ICMP, and the source can be tricked into fragmenting TCP segments [31]. With the exposition of packet header identifiers in a vast number of transmissions, the attacker can carefully craft a fragment to attack, such as causing misassociation and leading to unsuccessful firmware updates or the installation of malicious code.

3. Proposed Protection Mechanism

3.1. Random Forest

Random forest [32] is a machine learning algorithm that adopts the theory of many decision trees with the bagging technique. Each of the trees is called a forest and are not correlated with one another. Randomly selected attributes split the nodes in the decision tree without pruning. Additionally, the bootstrap method uses sampling with a replacement which can compensate for the situation of insufficient data to train the model. Figure 2 shows the prediction based on the average result of all the forests. Using the maximum voting technique, there is a high probability of getting the correct answers.

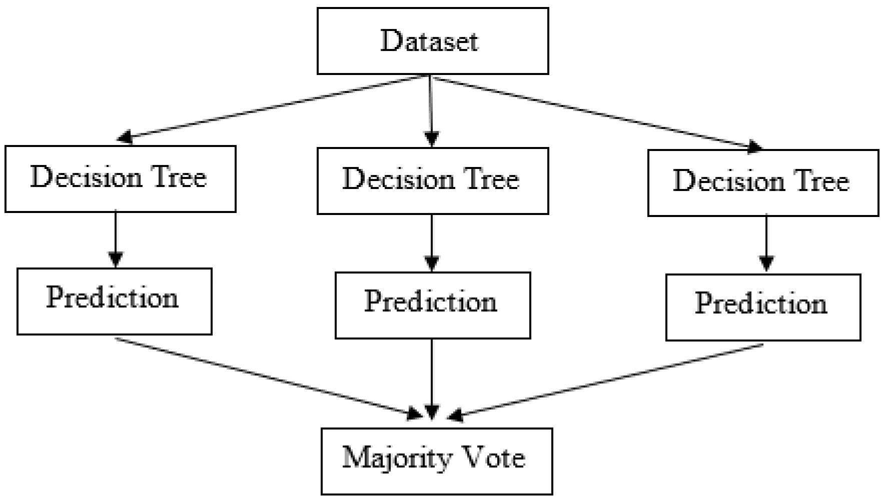

Figure 2. Random forest algorithm.

3.2. Principal Component Analysis

Principal component analysis (PCA) [33] is a technique for dimensionality reduction. It converts the original data into uncorrelated principal components (PCs) and arranges them from highest to lowest variance. The result for PCA is to transform the dataset into PCs with the highest possible variances. Thus, the first selected PC contributes the highest variance of the data.

The eigenvector U of matrix A and the λ which is the scalar eigenvalue for matrix A (square matrix) as in Equation (1):

$$AU = U\lambda \tag{1}$$

Equation (2) is the root of the characteristic equation, which is solved to obtain the value of λ:

$$det(\mathrm{A} - \lambda\mathrm{I}) = 0, \tag{2}$$

3.3. Entropy-Based Detection

Shannon's entropy [34,35] is an information theory that measures randomness and distribution. This method effectively detects an attack on networks, considering the massive traffic generated. When the attack occurs in the network, data collection and features can easily capture issues by deriving the abnormal behavior in the context. Some features used for entropy computation are source IP, destination IP, and protocol.

ARP Spoofing Analysis

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_b p(x_i), \tag{3}$$

Algorithm 1 uses entropy (H) shown in Equation (3) to detect abnormality in the local network. The entropy in the ARP spoofing mainly falls to 1.3 [34]. In ARP spoofing activities, the entropy can achieve a high value, but not too high, compared to the normal ARP process. The threshold set by entropy can detect the abnormality in ARP traffic, which triggers detection and protection.

Let S be a set of ARP packets collected throughout the simulation, grouped by each time window W. The x_i is the number of packets sent from a source. To calculate $p(x_i)$, x_i is divided by a denominator of the total number of ARP packets at the time window or timeslot i. The b represents base 2 of a log.

The ARP traffic is collected at each time window and compared with the predefined threshold. The feature that applies in the context is using the sender's MAC address. The entropy property shows the underlying probability distribution and describes the randomness of the flows in the observation point or location, in this case, the switch or access point (AP). The MAC address is selected to calculate the entropy of each time window. This feature shows the robustness of identifying the attacker's behaviors correctly. If the attacker begins to screen the local network for all active IP addresses, this increases the randomness of the destination IP address. In the case of this paper, the sender's MAC address can capture many ARP packets in the network traffic from the attacker.

Algorithm 1 Proposed ARP Spoofing Detect and Protect Based on Entropy

```
log_monitor_ip_mac: dictionary to keep IP-MAC mapping
Input: Time Window slot (W1. . . Wn) = S, Attr1 as sender MAC
Output:
Set Entropy threshold $\leftarrow$ 1.35
for all Wi $\in$ S do
//get entropy of the ARP traffic in current window slot
curEntropyAttr1 $\leftarrow$ COMPUTE entropy of sender MAC
if (APi(MAC) in capture MAC address)
then
filterAttackerTraffic()
end if
//raise alarm
if (curEntropyAttr1 < Entropy threshold)
then
ARP List $\leftarrow$ UNIQUE IP, MAC from a high volume of ARP packet's in Wi
//get the Sender IP, MAC in ARP
MAC $\leftarrow$ MAC in ARP List
IP $\leftarrow$ IP in ARP List
length $\leftarrow$ COUNT ARP List
if (length > 1)//duplicate address found
then
detect spoofing $\leftarrow$ True
else
match $\leftarrow$ matchMacToIp (IP, MAC, log_monitor_ip_mac)
if(!match) then
detect spoofing $\leftarrow$ True
end if
end if
if (detect spoofing = True)
then
//keep the mac address in a list
```

Algorithm 1 *Count.*

```
UPDATE MAC in capture mac address
else
//keep the IP-MAC in dictionary for further validation
UPDATE IP, MAC in log_monitor_ip_mac
end if
end if
```

3.4. One-Time Code and Timestamp

The packet header identifier is vital on the Internet to a similar group of fragments and forms the original packets at the destination node.

The proposed mechanism is adding a timestamp and one-time code (OTC) in the fragment to protect against the fragmentation attack caused by a predictable identifier. The OTC and timestamp consume the option fields with a maximum of 40 bytes. The OTC is a number randomly generated parallel to the timestamp. With the additional fields added to the packet header, the attacker cannot force the reassembly of the fragments with only the source IP, destination IP, identification, and protocols. The following diagram shows the implementation of OTC and timestamp in Figure 3.

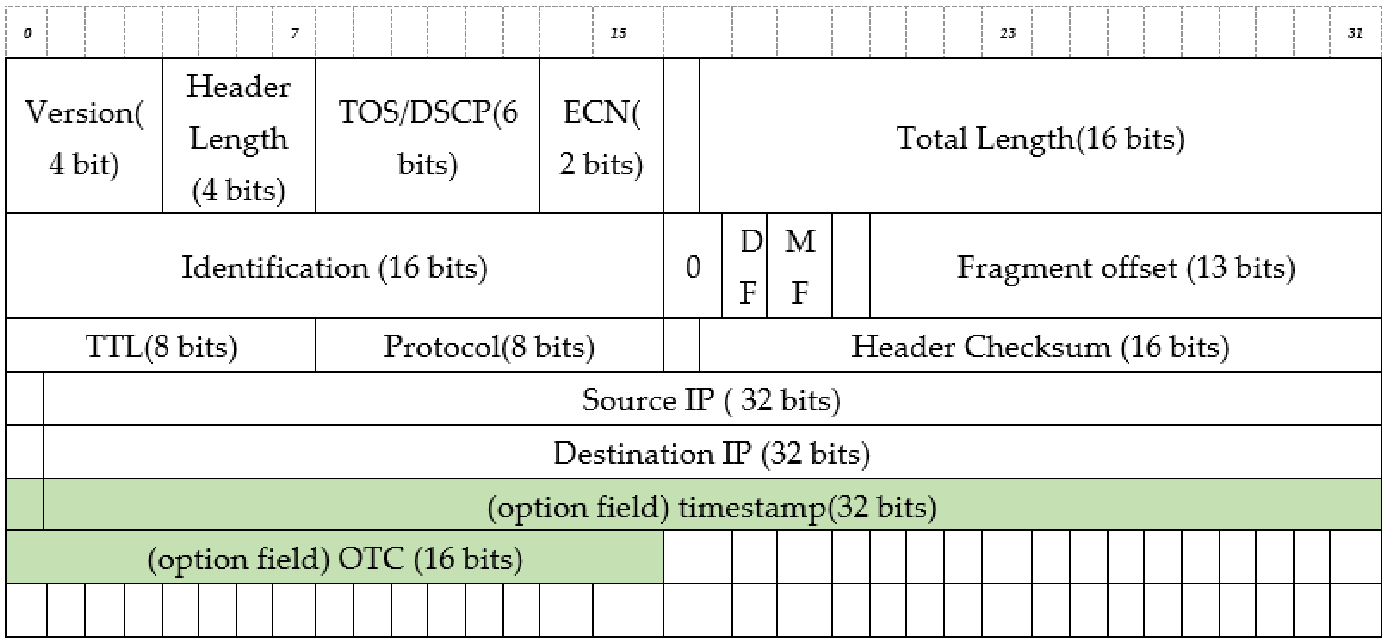

Figure 3. Proposed IPv4 header to include timestamp and OTC as an enhancement for fragmentation attack (misassociation) protection.

Type of service (TOS) or differentiated service code point (DSCP) specifies the different services like Voice over IP (VOIP). If the explicit congestion notification (ECN) is set, the system can send the notification of network congestion between endpoints, preventing packet drop. The Don't Fragment (DF) field allows for the discovery of the path of the maximum transmission unit (MTU) and prevents fragmentation between communication nodes. When fragmentation occurs, all of the fragments except the last one will have the More Fragment (MF) field set to true, indicating that more fragments are coming.

3.5. Dataset

The dataset is prepared by randomly generating data points with features stated in Table 1. This record simulates the DDoS scenarios. The classifiers use this dataset to train a model for predicting potential attackers.

Table 1. Configuration of DDoS simulation data for attack and benign.

Node	Source IP	Destination IP (Server)	Source Port	Destination Port	Bytes (Sent)
Legitimate (65 nodes)	Unique	1	Random	Highly random (port no. 80–11,000)	Varies
Attacker (35 nodes)	Unique	1	Random	Random (with port no 3000–11,000)	Similar

4. Simulations and Results

The hardware used to perform the analysis is a computer with Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz 2.40 GHz, 12 GB RAM with Windows 10 Home as the operating system. Simulation software is OMNeT++ [36] with an INET framework [37] as a model library. Execution of protection codes for DDoS and ARP spoofing was conducted in Python 3.8.8 with Spyder (Scientific Python Development Environment) IDE (integrated development environment). The customization in the INET framework for IP fragmentation protection code is needed to use the proposed packet format.

4.1. DDoS

DDoS simulates the infected devices, called bots, to synchronize the attack to target the server. The data are passed from the infected devices to a computer or gateway at each local network. Excessive data are flooding the server-side router, thus causing degradation of the services. The detection and protection mechanism are implemented in the simulation to recover the traffic. The models are evaluated based on predefined formulas described in the next section.

4.1.1. Simulation Analysis

The dataset comprises 65 percent benign traffic and 35 percent attack traffic generated from 100 nodes. The benign traffic consists of TCP and UDP data, and attack traffic is all from UDP data. Protocol, source port, destination port, and byte are features shown in Table 2. All of them are from numerical data types. The dataset is prepared based on the following mentioned criteria. First, attributes or features should not have missing or defective values. Second, attributes or features are verified with a correlation matrix. This method ensures that no highly correlated features (<0.8) exist among the features, which can prevent the curse of dimensionality in a model's training process. Based on Figure 4, the highest correlation among independent attributes is between the protocol and byte, with 0.326. Third, features of data in the dataset are standardized by performing feature scaling. This method avoids the features with a higher range of values dominating the classifiers' model.

Table 2. Dataset attributes.

No.	Field Name	Field Name (Dataset)	Data Type	Description
1	Protocol	Protocol	Numerical	Protocol id represent TCP/UDP
2	Source port	Srcport	Numerical	Source port from sender (incoming)
3	Destination port	Destport	Numerical	Destination port (incoming)
4	Bytes	Bytes	Numerical	Data bytes (incoming)

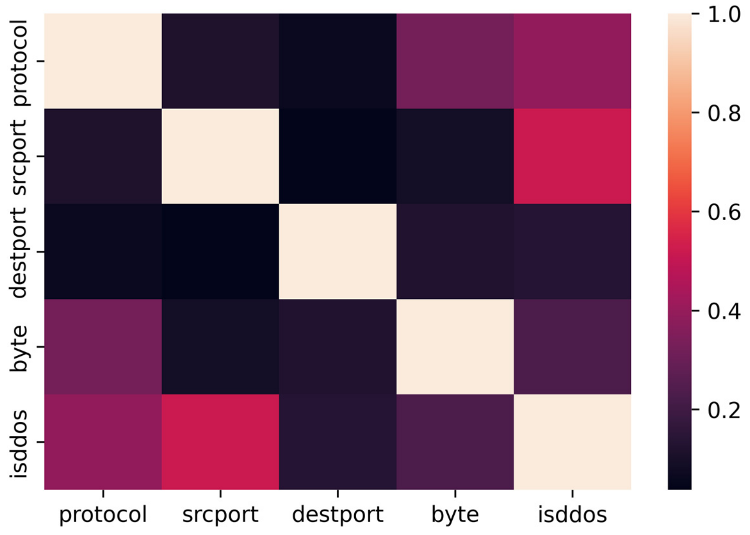

Figure 4. Correlation matrix of the dataset attributes.

Table 3 shows the classifiers in comparison. The simulation configuration is shown in Table 4.

Table 3. Papers referring to DDoS.

Method	Authors	Reference
SVM	K.M. Sudar, M. Beulah, P. Deepalakshmi, P. Nagaraj and P. Chinnasamy	[23]
KNN	S. Dong, and M. Sarem	[24]
RF	J. Pei, Y. Chen, and W. Ji	[25]

Table 4. DDoS simulation parameters.

Parameter	Value
No. of legitimate nodes (Sender/gateway)	65
No. of attacker (Gateway)	35
No. of router (Server-side)	1
No. of Server	1
Attacker target	Server
Simulation time	1 s
Attack duration	0.5–0.6 s
Sending interval (Attackers)	0.0001 s
Sending interval (Legitimate node)	UDP = 0.01–0.02 s; TCP = 0.3 s

Below are predefined definitions.

TP = true positive, TN = true negative, FP = false positive, FN = false negative

The formulas adopted in the evaluation of classifiers are shown below.

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}), \tag{4}$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}), \tag{5}$$

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}), \tag{6}$$

$$\text{F1-Score (a.k.a. F-Score)} = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall}), \tag{7}$$

4.1.2. Evaluation and Discussion

Precision and recall calculations are shown in Equations (4) and (5), respectively. The dataset was split into 75 percent for the training dataset and the remaining 25 percent for the test dataset. The classifiers' performances are related to the convergence model to make a DDoS classification. Based on Table 5, the added features can improve the prediction in terms of accuracy. Figure 5 shows the F1-scores of each classifier's predictions based on two, three, and four features. RF, KNN, and SVM classifiers have started with lower F1-scores in the two features model, which adopted protocol and source port to train a model. The scores increased in the three features model, which included the destination port. RFPCA prediction achieved a 95.65% F1-score and 97% accuracy using the dataset with four features to train the model.

Table 5. Comparison results of classifiers.

		RFPCA	**RF**	**KNN**	**SVM**
Two features	F1	91.43	83.33	69.7	74.67
	Accuracy	94	88	80	81
	Precision	91.43	81.08	74.2	70
	Recall	91.43	85.71	65.71	80
Three features	F1	91.18	89.55	81.01	79.02
	Accuracy	94	93	85	83
	Precision	93.94	93.75	72.72	69.57
	Recall	88.57	85.71	91.43	91.43
Four features	F1	95.65	91.18	82.67	80.52
	Accuracy	97	94	87	85
	Precision	97.06	93.94	77.5	73.81
	Recall	94.29	88.57	88.57	88.57

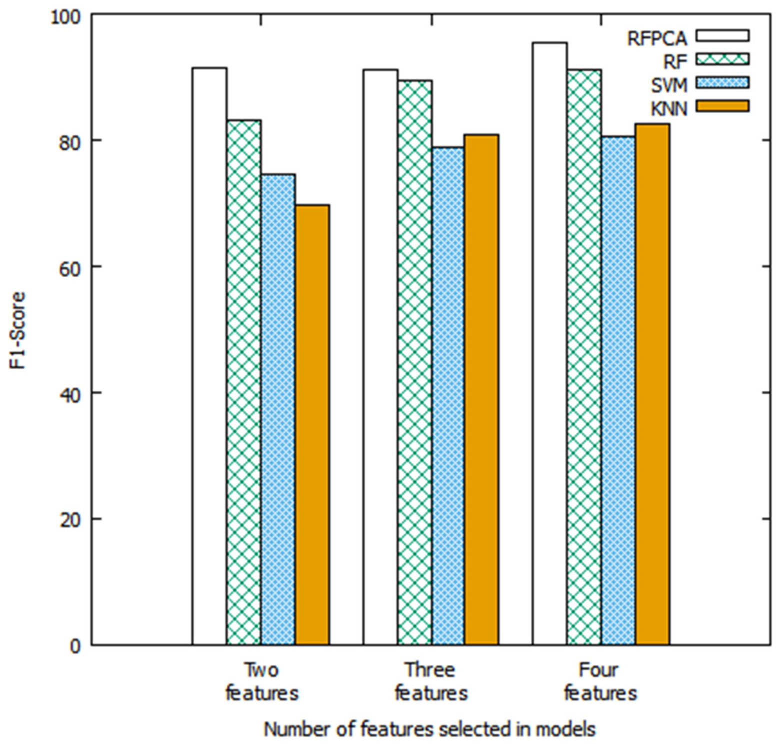

Figure 5. F1-score comparison of classifiers.

The same models in Figure 5 are applied in the DDoS classification problem presented in the simulation shown in Section 4.1.1. The entropy highlighted the different detection rates by the classifiers in the traffic analysis. Based on the model performance, training more features led to a more accurate model and may take more computation time. In Figures 6a, 7a and 8a, the entropy values were maintained between 5.43–5.57 in 0–0.5 s before the attack. The attack launched in 0.5–0.6 s, and entropy dropped to lower values showing that the attack was concentrated in the network and differentiating the detection rates between classifiers.

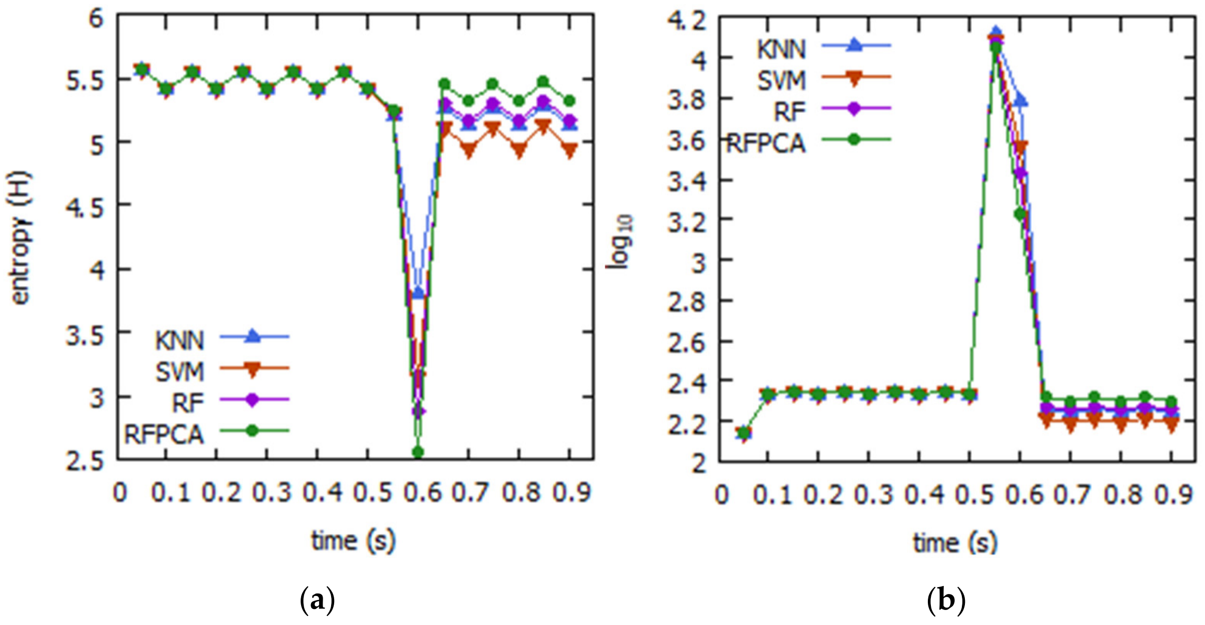

Figure 6. Comparison of detecting attacks and protecting networks of different models trained on two features. (**a**) Entropy-based analysis. (**b**) Log_{10} results with different models' implementation.

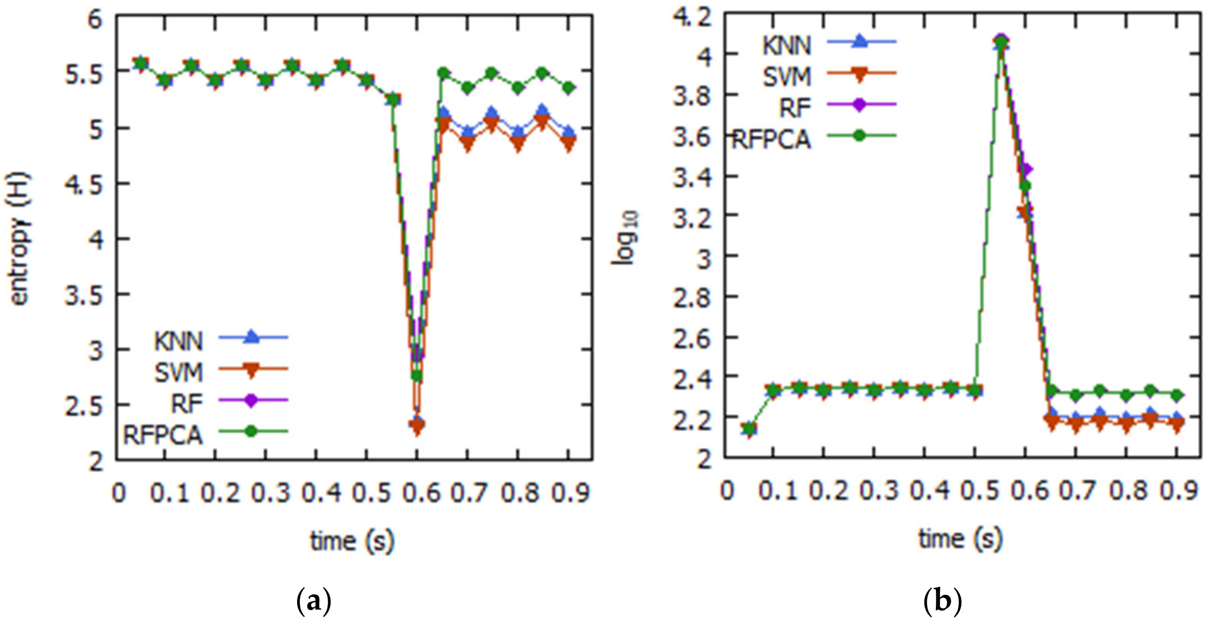

Figure 7. Comparison of detecting attacks and protecting networks of different models trained on three features. (**a**) Entropy-based analysis. (**b**) Log_{10} results with different models' implementation.

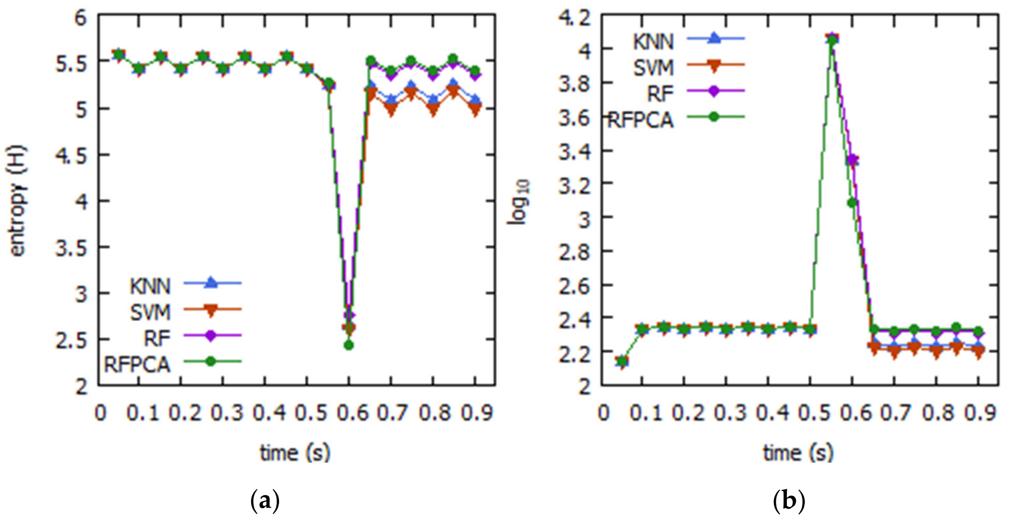

Figure 8. Comparison of detecting attacks and protecting networks of different models trained on four features. (**a**) Entropy-based analysis. (**b**) Log_{10} results with different models' implementation.

According to Figures 6b, 7b and 8b, the high detection rates prevented the attack traffic from overwhelming the network. Before the attack, traffic was maintained between 2.15–2.35. In Figure 8b, RFPCA shows a quick recovery of the network back to the normal stage, which dropped to 3.08 at 0.6 s and was maintained between 2.32–2.34 after 0.6 s.

Based on the overall results, the recall rate contributed to classifying the attack traffic. High detection rate is essential to capturing an attack, while good classification models can protect and reduce maintenance costs. The high F1-score shows an overall performance ability to correctly classify the DDoS in an imbalanced dataset, which seeks to balance precision and recall. Both RFPCA and RF models trained using four features have a good classification model. RFPCA, which performed the PCA before the training, can maintain high variances of the sample features and lead to the overall prediction performance.

4.2. ARP Spoofing

ARP spoofing simulates the attack in the local network, which can be either wired or wireless communication. Various attack scenarios are adopted to compare the effectiveness of the detection and protection mechanisms.

4.2.1. Simulation Analysis

The simulation has three attack modes, as shown in Table 6, implemented with the configuration in Table 7. Table 8 shows the different methods of protection. The proposed method sets the entropy threshold to $H < 1.35$. In a typical ARP spoofing, the attacker screens the local network to identify the active IP in the local network and the attack follows based on the selected attack mode. The system identifies the ARP spoofing that has occurred, triggers the protection algorithm, and inserts rules to filter the ARP packet based on the attacker's MAC.

Table 6. ARP spoofing attack mode.

Attack Mode	Description
Mode-1	The attacker sends spoofed ARP requests to the router and Node03.
Mode-2	The attacker sends spoofed ARP replies, mimicking a gratuitous ARP variant to the router and Node03.
Mode-3	The attacker replies to a legitimate ARP request with spoofed ARP messages to Node03 and then sends spoofed ARP requests to the router.

Note: The attacker sends the spoofed ARP packets continuously during the attack session to maintain the connection between target nodes.

Table 7. ARP spoofing simulation parameter.

Parameter	Value
No. of attackers	1
No. nodes in local network (with router)	10
No. of servers	1
Attacker target	2 nodes (Router and Node03)
Simulation time	100 s
Attack session	20–70 s
ARP cache timeout	30 s

Table 8. Comparison of detection and protection.

Method	Description
Proposed method	ARP packets are grouped by time window. The protection algorithm is triggered to check for any IP-MAC mapping mismatch or duplicate IP showing the attacker's address.
Method-1 [26]	Each window consists of a maximum of 260 packets. In the separate window, if less than 1/3 of the ARP requests belong to the ARP replies triggered in bulk, then the source address in the ARP reply is the attacker's address.
Method-2 [27]	Spoofed ARP, a variant of gratuitous ARP, is detected if the destination address specifies the destination node.

4.2.2. Evaluation and Discussion

In Figures 9a, 10a and 11a, entropy values dropped below the threshold in 0–80 s indicating that entropy can distinguish attack traffic and regular traffic well. Local nodes initiated the ARP processes were in a cache timeout in the 40–60 s range. Figures 9a and 11a show that the victims were involved in the ARP process by responding to ARP requests, and the entropy values achieved 1.22 in Figure 9a and 1.27 in Figure 11a. In the following figures, the detection time of each method is compared with the total of ARP packets.

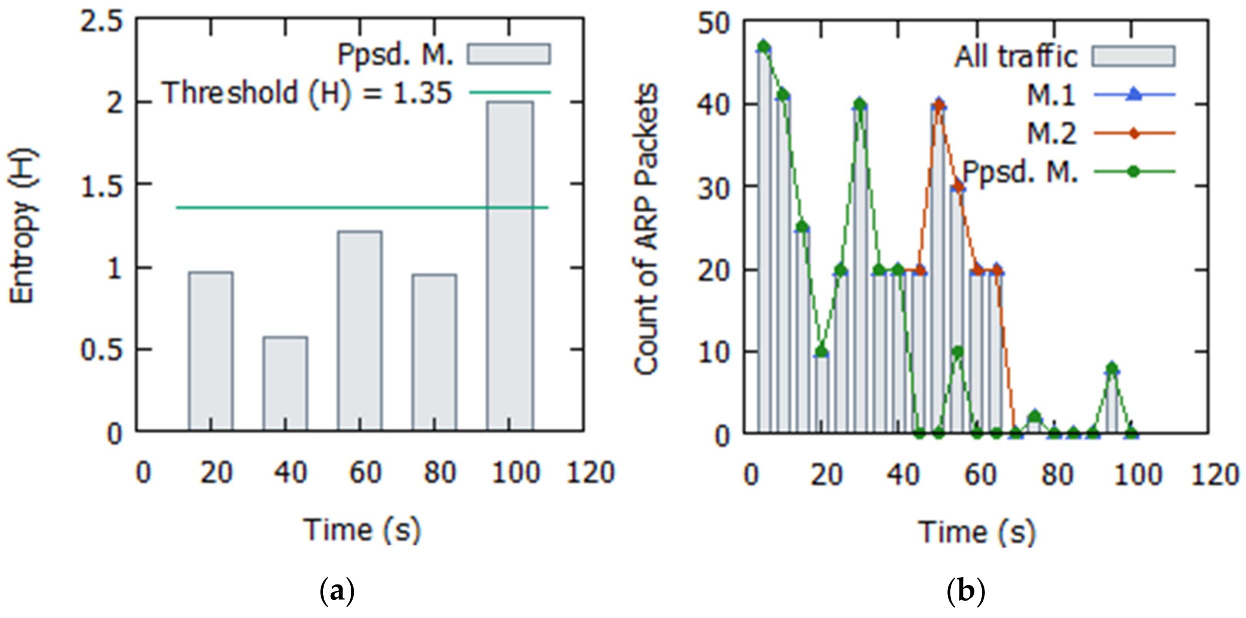

Figure 9. Mode-1 scenario in ARP spoofing. (**a**) Entropy of proposed method (**b**) The comparison results of different types of protection in a total of 363 packets.

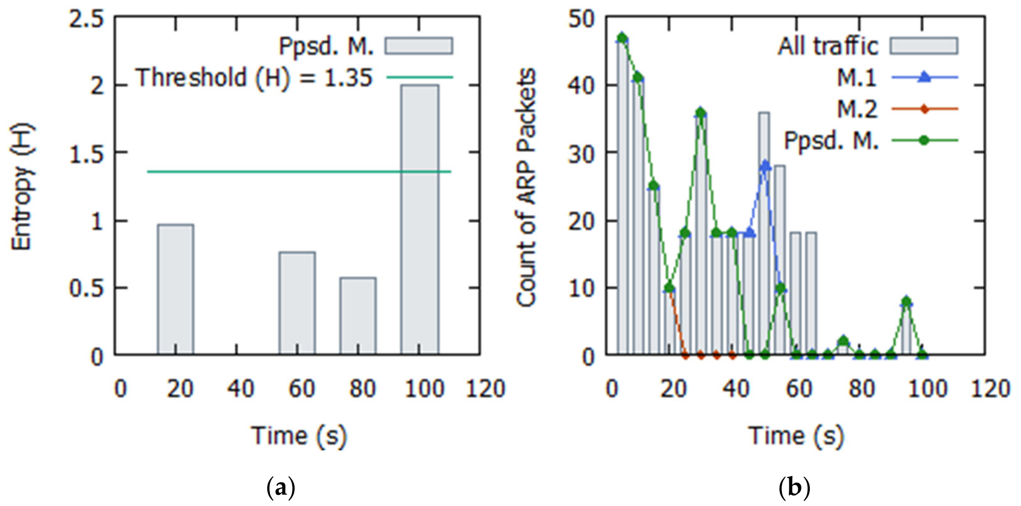

Figure 10. Mode-2 scenario in ARP spoofing. (**a**) Entropy of proposed method (**b**) The comparison results of different types of protection in a total of 341 packets.

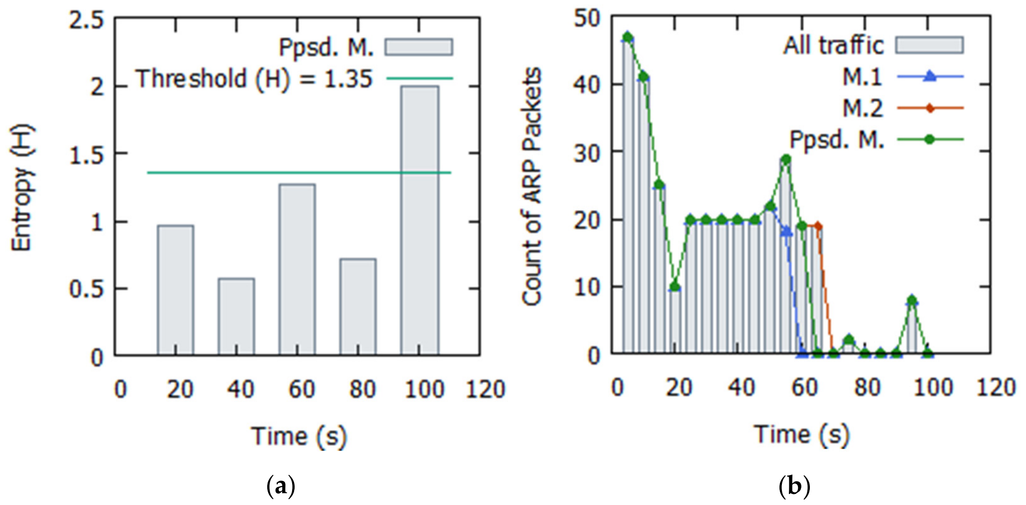

Figure 11. Mode-3 scenario in ARP spoofing. (**a**) Entropy of proposed method (**b**) The comparison results of different types of protection in a total of 322 packets.

In Figure 9b, the proposed method detected ARP spoofing and blocked the malicious traffic in 40 s. There was no action from Method-1 and Method-2, and the total ARP traffic remained at 305 attacks out of 363 packets.

In Figure 10b, Method-2 detected the attack in 20 s. This ARP packet adopted a gratuitous ARP variant in this attack to forge the target's IP-MAC mapping. The proposed method detected ARP spoofing in 40 s. Method-1 blocked the attacker at 46 s. There was a total of 305 attacks out of 341 packets.

In Figure 11b, the attacker MAC could be identified after 45 s when the Node03 initiates an ARP request after the ARP cache timeout. Method-1 blocked the attacker at 51 s. The proposed method blocked the attacker in 60 s. There was no action reflected in Method-2 in this case, and the attacks remained at 270 out of 322 packets.

According to the results obtained, the proposed method can detect different spoofed ARP packets and automatically capture the suspicious address mapping matched with the attacker.

4.3. IP Fragmentation Attack (Mis-Association)

The server routes the firmware packets through the gateway to the IoT. Data transmission is subjected to limited computation and memory. The packet header identifier is predictable because it sends large packets to the target. The attackers launched the attack by sending fragments with the same identity. The attack attempts to merge an attack fragment with fragments from the server, leading to incorrect packet reassembling and the dropping of the packet.

4.3.1. Simulation Analysis

The configuration in Table 9 shows a simulation that has implemented the OTC and timestamp to confront and mitigate the fragmentation attack. All fragments from the same packet should have a similar identifier, source IP, and destination IP with valid OTC and timestamp. The validity of fragments is verified at the destination node after the reassembly process.

Table 9. Fragmentation attack parameters of the simulation.

Parameter	Value
No. target IoT	1
No. of attackers	4
Total cycle of simulation/length	40 cycle/400 s
Data/cycle (Firmware size)	4 kB
Protection	One-time code (OTC) and timestamp

4.3.2. Evaluation and Discussion

Based on Figure 12, the target IoT has received all the fragments sent from the gateway and attackers. Figure 13 shows that the destination IoT (target) detected all attackers' fragments. The identifier plays a vital role in Internet-based communication, and the IoT can connect to a vulnerable WiFi network for communication. Attackers employed a method to randomly spot the identifier values by sending many fragments into the network to increase the chances of a successful attack on the IoT but have failed in all the attempts based on the simulation. The protection of OTC and timestamp has created difficulty for attackers to launch a successful IP fragmentation. Without any success of the fragments used by attackers, the destination node has received all the data correctly.

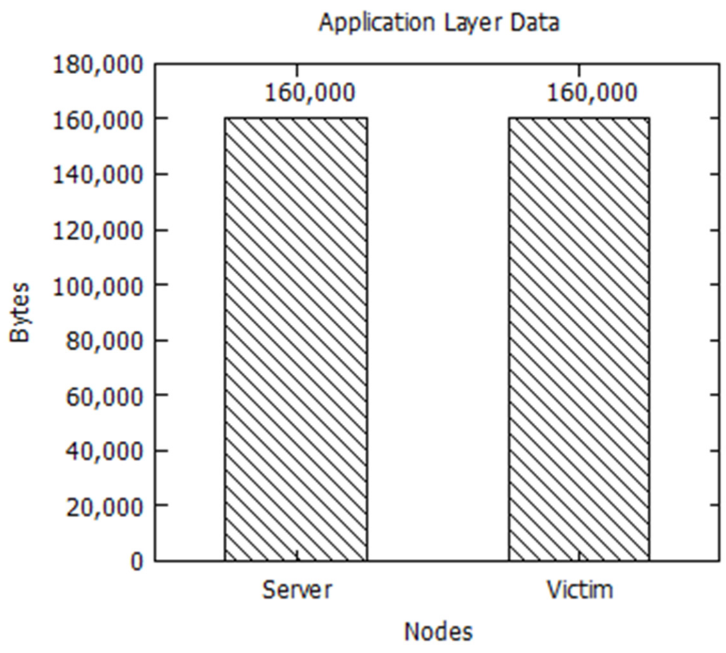

Figure 12. Total bytes from sender to receiver (end-to-end).

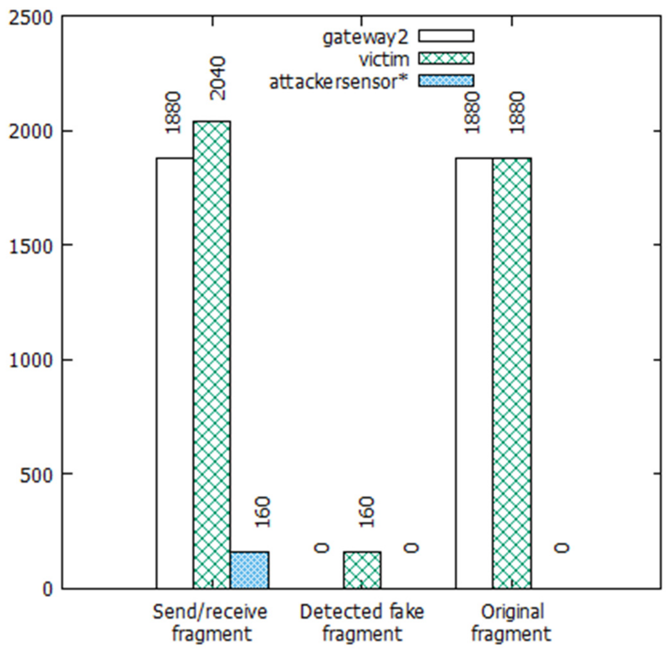

Figure 13. Fragment count and statuses.

In typical transmission, the data travel from different networks with different MTU sizes and cause fragmentation, especially when a large amount of data are involved. Only the first fragment carried the high-layer information essential for routing and network functionalities, which may lead to the retransmission of the entire packet, if lost. A general practice to avoid fragmentation in the first place requires additional transmission, and a TCP connection can cause an attacker to adopt alternative methods to attack. If the system can identify the attack immediately, the error may be corrected with a different transmission. In the worst cases, in which the attacker has launched maliciously, it can open a security breach in the network. Thus, protection for the packet header identifier contributes to the secure transmission of fragments with the identifier, which the operating system implemented straightforwardly, and the secure transmission is now protected.

5. Conclusions

In this paper, the proposed protection for DDoS, ARP spoofing, and IP fragmentation attacks are presented and discussed. Cybersecurity threats come from internal and external sources, and the attacks can be stealthy. These attacks are a common yet severe problem, and once they occur, they can cause massive damage to the infrastructure and loss to the organization. Several solutions have been proposed to solve the problems, each with advantages and disadvantages, and the detection rates and accuracy are varied. The proposed RF classifier with PCA can train a model that copes with dynamic environments and delivers good protection. The protection in ARP spoofing captures suspicious address mapping and is adaptable to different spoofed ARP packets. The protection of the packet header identifier solves the security breach in the IP-based communication, and the destination node can detect the fake fragments without impacting the reassembly process. The system should have sufficient protection capable of detecting the attack at the earliest chance, if not immediately, without impacting the organization's productivity.

Author Contributions: Conceptualization, T.U.C., H.G.G. and V.P.; methodology, T.U.C.; software, T.U.C.; validation, T.U.C.; formal analysis, T.U.C.; investigation, T.U.C., S.-Y.L. and H.G.G.; writing—original draft preparation, T.U.C.; writing—T.U.C. and H.G.G.; visualization, T.U.C. and H.G.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the respected editor and reviewer for their support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Russmann, M.; Lorenz, M.; Gerbert, P.; Waldner, M.; Justus, J.; Engel, P.; Harnisch, M. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consult. Group* **2015**, *9*, 54–89.
2. Chen, B.; Wan, J.; Shu, L.; Li, P.; Mukherjee, M.; Yin, B. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access* **2017**, *6*, 6505–6519. [CrossRef]
3. Yao, X.; Zhou, J.; Lin, Y.; Li, Y.; Yu, H.; Liu, Y. Smart manufacturing based on cyber-physical systems and beyond. *J. Intell. Manuf.* **2019**, *30*, 2805–2817. [CrossRef]
4. Nagy, J.; Olah, J.; Erdei, E.; Mate, D.; Popp, J. The role and impact of industry 4.0 and the internet of things on the business strategy of the value chain-the case of hungary. *Sustainability* **2018**, *10*, 3491. [CrossRef]
5. Liu, Q.; Li, P.; Zhao, W.; Cai, W.; Yu, S.; Leung, V.C.M. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE Access* **2018**, *6*, 12103–12117. [CrossRef]
6. Syafrudin, M.; Alfian, G.; Fitriyani, N.L.; Rhee, J. Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors* **2018**, *18*, 2946. [CrossRef] [PubMed]
7. Sengupta, J.; Ruj, S.; das Bit, S. A secure fog-based architecture for industrial internet of things and industry 4.0. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2316–2324. [CrossRef]
8. Qi, Q.; Tao, F. A smart manufacturing service system based on edge computing, fog computing, and cloud computing. *IEEE Access* **2019**, *7*, 86769–88677. [CrossRef]
9. Vaidya, S.; Ambad, P.; Bhosle, S. Industry 4.0—A glimpse. *Procedia Manuf.* **2018**, *20*, 233–238. [CrossRef]
10. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT privacy and security: Challenges and solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
11. Tuptuk, N.; Hailes, S. Security of smart manufacturing systems. *J. Manuf. Syst.* **2018**, *47*, 93–106. [CrossRef]
12. Yu, M.; Zhuge, J.; Cao, M.; Shi, Z.; Jiang, L. A survey of security vulnerability analysis, discovery, detection, and mitigation on IoT devices. *Future Internet* **2020**, *12*, 27. [CrossRef]
13. Staddon, E.; Loscri, V.; Mitton, N. Attack categorisation for IoT applications in critical infrastructures, a survey. *Appl. Sci.* **2021**, *11*, 7228. [CrossRef]
14. Lee, Y.; Chae, H.; Lee, K. Countermeasures against large-scale reflection DDoS attacks using exploit IoT devices. *Automatika* **2021**, *62*, 127–136. [CrossRef]
15. Li, D.; Yu, C.; Zhou, Q.; Yu, J. Using SVM to Detect DDoS Attack in SDN Network. *J. Phys. Conf. Ser.* **2019**, *1237*, 1–5. [CrossRef]
16. Lohachab, A.; Karambir, B. Critical analysis of DDoS—An emerging security threat over IoT networks. *J. Commun. Inf. Netw.* **2018**, *3*, 57–78. [CrossRef]
17. Butun, I.; Osterberg, P.; Song, H. Security of the internet of things: Vulnerabilities, attacks and countermeasures. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 616–644. [CrossRef]
18. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the mirai botnet. In Proceedings of the 26th USENIX Security Symposium, Vancouver, BC, Canada, 16–18 August 2017; pp. 1093–1110.
19. Sinanovic, H.; Mrdovic, S. Analysis of Mirai malicious software. In Proceedings of the 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 21–23 September 2017.
20. *Router and IoT Vulnerabilities: Insecure by Design*; IOT Security Foundation: West Lothian, UK, 2021.
21. Aytac, T.; Aydın, M.A.; Zaim, A.H. Detection DDoS attacks using machine learning methods. *Electrica* **2020**, *20*, 159–167. [CrossRef]
22. Wang, S.; Gomez, K.; Sithamparanathan, K.; Asghar, M.R.; Russello, G.; Zanna, P. Mitigating DDoS attacks in SDN-based IoT networks leveraging secure control and data plane algorithm. *Appl. Sci.* **2021**, *11*, 929. [CrossRef]
23. Sudar, K.M.; Beulah, M.; Deepalakshmi, P.; Nagaraj, P.; Chinnasamy, P. Detection of distributed denial of service attacks in SDN using machine learning techniques. In Proceedings of the 2021 International Conference on Computer Communication and Informatics, ICCCI 2021, Coimbatore, India, 27–29 January 2021; pp. 25–29.
24. Dong, S.; Sarem, M. DDoS attack detection method based on Improved KNN with the degree of DDoS attack in software-defined networks. *IEEE Access* **2020**, *8*, 5039–5048. [CrossRef]
25. Pei, J.; Chen, Y.; Ji, W. A DDoS attack detection method based on machine learning. *J. Phys. Conf. Ser.* **2019**, *1237*, 032040. [CrossRef]

26. Gao, W.; Sun, Y.; Fu, Q.; Wu, Z.; Ma, X.; Zheng, K.; Huang, X. ARP poisoning prevention in internet of things. In Proceedings of the 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, China, 19–21 October 2018; pp. 733–736.
27. Abid, M.; Singh, A. ARP spoofing detection via wireshark and veracode. *Int. J. New Technol. Res.* **2018**, *4*, 263063.
28. Aly, M.; Khomh, F.; Gueheneuc, Y.G.; Washizaki, H.; Yacout, S. Is fragmentation a threat to the success of the internet of things. *IEEE Internet Things J.* **2019**, *6*, 472–487. [CrossRef]
29. Feng, X.; Li, Q.; Sun, K.; Xu, K.; Liu, B.; Zheng, X.; Yang, Q.; Duan, H.; Qian, Z. PMTUD is not panacea: Revisiting IP fragmentation attacks against TCP. In Proceedings of the Network & Distributed System Security Symposium (NDSS), San Diego, CA, USA, 24–28 April 2022.
30. Suciu, I.; Vilajosana, X.; Adelantado, F. An analysis of packet fragmentation impact in LPWAN. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018.
31. Dai, T.; Shulman, H.; Waidner, M. DNS-over-TCP considered vulnerable. In Proceedings of the Applied Networking Research Workshop (ANRW), New York, NY, USA, 24–30 July 2021; pp. 76–81.
32. Mohandoss, P.; Shi, Y.; Suo, K. Outlier prediction using random forest classifier. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021.
33. Hoang, D.H.; Nguyen, H.D. A PCA-based method for IoT network traffic anomaly detection. In Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, Republic of Korea, 11–14 February 2018; pp. 381–386.
34. Alghawli, S. Complex methods detect anomalies in real time based on time series analysis. *Alex. Eng. J.* **2022**, *61*, 549–561. [CrossRef]
35. Komazec, T.; Gajin, S. Analysis of flow-based anomaly detection using shannon's entropy. In Proceedings of the 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 26–27 November 2019; pp. 1–4.
36. OMNeT++. Omnet++ Discrete Event Simulator. Available online: omnetpp.org (accessed on 10 January 2023).
37. INET. Inet Framework. Available online: https://inet.omnetpp.org/Introduction.htm (accessed on 10 January 2023).