Generating IoT Specific Anomaly Datasets Using Cooja Simulator (Contiki-OS) and Performance Evaluation of Deep Learning Model Coupled with Aquila Optimizer

Vandana Choudhary, Sarvesh Tanwar and Tanupriya Choudhury

Department of Computer Science and Engineering, Amity Institute of Information Technology, Amity University Uttar Pradesh, India

Abstract: In recent times, the massive expansion of the Internet of Things (IoT) has transformed various facets of everyday life and industries. The compelling cause behind the widespread adoption of IoT is the increasing availability of affordable, compact, and energy-efficient computing devices. While these devices offer significant benefits, they also raise substantial security and privacy challenges. Consequently, safeguarding IoT networks and devices is imperative. To raise a robust security system for IoT networks, it is crucial to have an efficient anomaly-based intrusion detection system. In this study, we introduce a meticulous methodology to create IoT-specific datasets. Utilizing the Contiki-OS Cooja simulator, we generate datasets representative of real-world IoT security threats, including sinkholes, version numbers, and flooding attacks. We then evaluate the performance of a Convolutional Neural Network paired with an Aquila Optimizer (CNN-AO) using these self-generated datasets, by employing metrics such as accuracy, precision, recall, F1-score, sensitivity, specificity, and false alarm rate. Additionally, we compare the effectiveness of CNN and LSTM models in distinguishing between benign and malicious traffic. Our findings demonstrate that the CNN-AO model surpasses other models in accurately classifying normal and malicious traffic with an accuracy of 99.22, 99.77, and 99.55% for our self-generated malicious datasets based on sinkhole attack, version number attack, and flooding attack respectively. This novel approach not only establishes a solid foundation for future investigations in this domain but also provides valuable insights into enhancing IoT system security. In this study, we contribute to the field by introducing a robust methodology for IoT-specific dataset generation and evaluating a cutting-edge CNN-AO model for intrusion detection. Furthermore, it is crucial to note that this research was conducted with utmost ethical consideration. Ethical guidelines and data privacy concerns were meticulously addressed during the generation of IoT datasets and the simulation of real-world attack scenarios, ensuring the responsible conduct of our study.

Keywords: Internet of Things, Intrusion Detection System (IDS), Dataset Generation, Sinkhole Attack, Version Attack, Flooding Attack, Deep Learning

Introduction

IoT has seamlessly integrated technology into various aspects of our lives whether it be smart homes, the healthcare sector, or industrial automation to name a few presenting a dynamic landscape of advantages and challenges, especially in the realm of security and privacy (Al-Fuqaha et al., 2015; Balaji et al., 2019). The interconnectivity of IoT devices, characterized by resource constraints, low power capabilities, and diverse
communication protocols, inherently poses security risks and privacy breaches (Lin et al., 2017; Hassija et al., 2019). Attacks like sinkholes, version numbers, and flooding attacks are prominent threats that have the potential to compromise data confidentiality, integrity, and availability within IoT networks. Table 1 presents the objectives and impacts of sinkholes, version numbers, and flooding attacks. This study addresses the pressing need for effective IDS in the context of IoT security. Specifically, our work focuses on generating realistic and comprehensive datasets utilizing the Contiki Cooja simulator. We simulate attacks such as sinkholes, version numbers, and flooding attacks within the Routing Protocol for Low-power and Lossy Networks (RPL). A prevalent routing protocol tailored for resource-constrained devices in IoT deployments. The controlled simulation environment allows us to delve into the underlying characteristics of these attacks, providing valuable insights into their behavior and impact on IoT networks.

Unlike existing research, our methodology incorporates detailed simulations of specific attacks using the Contiki Cooja simulator, enabling us to create datasets that closely mirror real-world scenarios. These datasets serve as a foundation for evaluating the effectiveness of various deep-learning models. By comparing the effectiveness of these models in identifying attacks, our research aims to enhance the accuracy of attack detection mechanisms.

This novel approach, emphasizing simulation fidelity and methodological rigor, distinguishes our study from previous work in the field of IoT security, contributing significantly to the scientific discourse on intrusion detection in IoT networks.

### Problem Statement

The functionality of RPL-based IoT networks is compromised by attacks like sinkholes, version numbers, and flooding attacks to name a few. For preserving the confidentiality, integrity, and availability of data during transmission in a network, it is crucial to identify and correctly categorize these attacks. To tackle the challenge of lack of IoT-specific datasets and attack detection and classification in IoT-specific environments, there is a need to generate comprehensive malicious datasets and leverage the generated datasets to train various models to make them learn and identify complex patterns and features indicative of each attack type, enabling accurate classification of unforeseen attacks in RPL networks.

This study's contributions include the following:

- Generating malicious datasets based on sinkhole attacks, version number attacks, and flooding attacks in the Contiki Cooja simulator
- Understanding the effects of a breached network while taking into account numerous factors like average power usage and average radio duty cycle
- Applying the IoT-specific datasets to train a few deep learning models namely, CNN-AO (Choudhary et al., 2023), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM), and comparing their performances

### Table 1: Description of sinkhole attack, version number attack, and flooding attack

<table>
<thead>
<tr>
<th>Attack</th>
<th>Objective</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sinkhole attack</td>
<td>The primary objective of this attack is to redirect legitimate network traffic to itself by manipulating routing metrics and falsely advertising an attractive routing path to the intended destination thereby captivating neighboring nodes</td>
<td>This attack leads to compromised data confidentiality, integrity, and availability in the network. Moreover, such an attack can also cause network congestion and degradation of the overall performance of a network, thereby disrupting the network's functionality.</td>
</tr>
<tr>
<td>Version number attack</td>
<td>The objective of this attack is to trick nodes into accepting false routing information by manipulating or fabricating version numbers. This could lead to the node updating its software with a the harmful version that the attacker can control</td>
<td>Version number attacks can lead to routing inconsistencies and can create routing loops within the destination-oriented directed cyclical graph. They can also cause unnecessary resource wastage within the network as nodes might initiate unnecessary route repairs or topology updates due to constant false advertisements of new versions.</td>
</tr>
<tr>
<td>Flooding attack</td>
<td>The objective of a flooding attack is to influx the system with a lot of malicious or undesired traffic</td>
<td>Here attacker overpowers the network with and too many packets, leading to delays in packet delivery, poor throughput, exhaustion of network resources and potential denial of service conditions. It makes the network inaccessible to legitimate nodes thereby disrupting communication.</td>
</tr>
</tbody>
</table>

366
Security and privacy concerns are inevitable owing to an increase in IoT device counts over the past few years. This situation necessitates the deployment of efficient IDSs to ensure the security of IoT networks. To build and utilize efficient IDS for safeguarding IoT environments, the availability of appropriate datasets apt for IoT contexts is crucial. Existing datasets like NSL-KDD (Tavallaee et al., 2009), ISCXIDS2012 (Shiravi et al., 2012), CICIDS2017, CICIDS2018 (Sharafaldin et al., 2018), etc., have been used extensively by the research community to develop and assess IDSs for IoT environments even to date.

These datasets have contributed significantly to the field of research related to IDS for IoT. However, the rapidly changing dynamics of IoT networks due to advancements in technology and communication protocols entail the development of IoT-specific datasets. In this study, we review the application of datasets like NSL-KDD, CICIDS2017, and CICIDS2018, among others, in IDS research for IoT. Also, we understand the importance of developing IoT-specific datasets. Many intrusion/malware detection algorithms exist in the literature as described by Thakkar and Lohiya (2021); Asharf et al. (2020); Banaamah and Ahmad (2022); Alzubi et al. (2022; 2023). Table 2 summarizes a study of a few different studies conducted specifically on IDS for IoT so far.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Approach adopted</th>
<th>Detection model considered</th>
<th>Datasets considered</th>
<th>Performance evaluation metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fatani et al. (2021)</td>
<td>At first, the authors extract the associated features from the input datasets, using CNN. Then, used Aquila optimizer to select the most appropriate characteristics and to decrease the dimensionality of the data</td>
<td>CNN along with a binary version of Aquila optimizer</td>
<td>CIC2017, NSL-KDD, IoT-IoT and KDD99</td>
<td>Accuracy, precision sensitivity, F1-measure</td>
</tr>
<tr>
<td>Roopak et al. (2020)</td>
<td>The authors put forward an advanced IDS for DDoS attack detection in IoT networks using a multi-objective optimization (NSGA-II-JAG)</td>
<td>CNN with LSTM</td>
<td>CICIDS2017</td>
<td>Accuracy, F1-score</td>
</tr>
<tr>
<td>Nagisetty and Gupta (2019)</td>
<td>Using the Keras high-level deep learning library, a framework for the identification of harmful activity in IoT networks is described in this study</td>
<td>Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), Deep Neural Networks (DNN) and Autoencoder</td>
<td>UNSW-NB15 and NSL-KDD99</td>
<td>Accuracy, F1-score, RMSE</td>
</tr>
<tr>
<td>Hwang et al. (2019)</td>
<td>The authors offered a novel word embedding approach and used LSTM to determine the temporal links existing between the fields in the packet header to extract the semantics of packets</td>
<td>LSTM</td>
<td>ISCX2012, USTC-TFC2016, Mirai-RGU, Mirai-CCU</td>
<td>Accuracy, precision recall, F1-score, FPR</td>
</tr>
<tr>
<td>Almarshedi et al. (2023)</td>
<td>Synthetic Minority Over-sampling Technique (SMOTE) is implemented in this study to handle the imbalanced dataset to improve the classification. In addition, the authors assessed the performance of their proposed model in comparison with the CNN model using both balanced and imbalanced datasets. A comparison with other models and related works were also exhibited</td>
<td>CNN-LSTM</td>
<td>UNSW-NB15</td>
<td>ACU, AUC, precision, recall, F1-score</td>
</tr>
<tr>
<td>Sayed et al. (2022)</td>
<td>In this study, the authors tested various approaches for enhancing the accuracy of training DL models on unbalanced datasets employing resampling and cost-sensitive learning. This study seeks to investigate the performance of CNNs created for IoT devices during cyberattacks and determine whether they may be utilized as anomaly-based IDS</td>
<td>CNN</td>
<td>NF-UNSW-NB15-v2</td>
<td>Accuracy, precision recall, F1-score</td>
</tr>
<tr>
<td>Henry et al. (2023)</td>
<td>The authors of this research offer a method that combines both CNN and GRU with 3 convolutional layers and 2 GRU layers</td>
<td>CNN-GRU</td>
<td>CICIDS-2017</td>
<td>Accuracy, precision, recall false Positive Rate (FPR), true Positive rate (TPR) and other aligned metrics</td>
</tr>
<tr>
<td>Almunay and Albuyrak (2023)</td>
<td>The authors used CNN, LSTM and CNN-LSTM as intrusion detection algorithms to find the aberrant patterns in the datasets</td>
<td>CNN, LSTM, CNN-LSTM</td>
<td>UNSW-NB15 and X-IoTID</td>
<td>Accuracy, precision, recall, F1-score</td>
</tr>
<tr>
<td>Omarov et al. (2023)</td>
<td>This study uses BiLSTM, which is a upgraded version of LSTM over CNN. The authors have adopted batch normalization in this study</td>
<td>CNN-BiLSTM</td>
<td>UNSW-NB15</td>
<td>Accuracy, precision, recall, F-measure, execution time</td>
</tr>
<tr>
<td>Zhang et al. (2023)</td>
<td>A two-stage paradigm for IoT intrusion detection was put forth by the authors. In Stage 1, the authors looked at six different machine-learning methods-</td>
<td>Light GBM-CNN</td>
<td>CSE-CIC-IDS2018</td>
<td>ACC, DR, FAR precision, F1-score MCC, train time, test time</td>
</tr>
</tbody>
</table>
The research gaps addressed in our study are as follows:

- **Limited IoT-specific datasets**: The existing datasets, such as NSL-KDD, CIC2017, CICIDS2018 have been extensively utilized in IDS research for IoT. However, the study emphasizes the critical need for the development of IoT-specific datasets (Essop et al., 2021). This indicates a gap in the availability of comprehensive and tailored datasets specifically designed for IoT environments. To reduce this research gap, we aim to develop IoT-specific datasets in our study.

- **Inadequate evaluation in IoT context**: While the mentioned studies have employed various deep learning models, their evaluation primarily relies on existing datasets. This leads to a gap in evaluating intrusion detection models in the context of real IoT network scenarios, raising questions about the models’ applicability and effectiveness in practical IoT environments. To reduce this research gap, we aim to assess the performance of a few deep learning models using our self-generated IoT-specific datasets.

- **Limited attention to resource-constrained IoT devices**: The research studies mentioned do not explicitly address the details related to resource-constrained IoT devices. IoT devices often have limitations in processing power and memory. In our study, we analyzed the average power consumption and average duty cycle of resource-constrained IoT devices considered in the simulation to understand their impact on the overall network.

### Materials and Methods

The fundamental objective of this study is to generate datasets tailored for IoT applications and assess the performance of the CNN-AO model. A comparative analysis is then conducted with CNN and LSTM models to evaluate their respective performances.

We will be looking in particular into the effects of sinkholes, version numbers, and flooding attacks on the network. We have made our datasets accessible for public use. Table 3 shows the specifications of the hardware and software that were used to conduct the experiment.

For simulating various attack scenarios in an IoT environment, the Contiki Cooja simulator which provides a realistic emulation of IoT devices was used. We have considered sky motes and z1 mote with the simulation parameters as follows in Table 4 for carrying out attacks under different scenarios in IoT environments.

The data collected during the experiment include packet traces, network statistics, and any other relevant information that is used to assess the impact on the performance of a network.

Next, we discuss a proposed methodology for generating malicious datasets using the Contiki Cooja simulator and assessing the performance of various models using self-generated datasets. The self-generated datasets will then be used for training and testing purposes. The methodology adopted to conduct the current study is outlined in Fig. 1.
Table 3: Specifications of the hardware and software used

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Microsoft Windows 10, Contiki-OS 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool used</td>
<td>Contiki Cooja simulator, Jupyter notebook</td>
</tr>
<tr>
<td>Processor</td>
<td>AMD Ryzen 5, 4500U</td>
</tr>
<tr>
<td>Memory</td>
<td>256 GB SSD</td>
</tr>
<tr>
<td>RAM</td>
<td>8 GB of RAM</td>
</tr>
<tr>
<td>GPU</td>
<td>AMD Radeon graphics operating at 2.38 GHz</td>
</tr>
</tbody>
</table>

Table 4: Simulation parameters

<table>
<thead>
<tr>
<th>Simulator parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root node</td>
<td>1</td>
</tr>
<tr>
<td>Sender nodes</td>
<td>2,3,4,5,6,7,8,9,10,11,12 (attack node)</td>
</tr>
<tr>
<td>Positioning</td>
<td>Random positioning</td>
</tr>
<tr>
<td>Radio medium</td>
<td>Unit Disk Graph Medium (UDGM): Distance loss</td>
</tr>
<tr>
<td>Interface range</td>
<td>100 m</td>
</tr>
<tr>
<td>Transmission range</td>
<td>50 m</td>
</tr>
<tr>
<td>Mote startup delay (ms)</td>
<td>1000 ns</td>
</tr>
<tr>
<td>Random seed</td>
<td>123,456</td>
</tr>
<tr>
<td>Objective function</td>
<td>Minimum rank with hysteresis objective function</td>
</tr>
</tbody>
</table>

Fig. 1: Methodology adopted to generate malicious datasets and to evaluate their performance

Fig. 2: Network topology considered for launching sinkhole attack

Fig. 3: Network topology considered for launching version number attack

Fig. 4: Network topology considered for launching flooding attack

Following is a detailed explanation of each step.

Simulation Phase

At this step, we simulate three attack scenarios in the Cooja Simulator. The network topology considered for launching three attacks namely: Sinkhole attack, version number attack, and flooding attack in each scenario is illustrated in Figs. 2-4 respectively. For each attack scenario, we have considered a total of 12 motes. Out of 12 motes, green-colored are UDP-server motes, yellow-colored motes are UDP-client motes, and purple-colored motes are the malicious motes.

To simulate a sinkhole attack, sky mote was used. A sinkhole attack is a variant of an isolation attack and is a combination of a rank decrease attack and a black hole attack. A sinkhole attacker captivates its neighbors to select it as their preferred parent and becomes its child using a rank decrease attack. Subsequently, it discards all packets sent by its children, effectively isolating them from both the root node and the network through a black hole attack.
To simulate a version number attack, sky mote was used. RPL uses Destination Oriented Direct Acyclic Graph (DODAG) version number and rank mechanism to identify and maintain a network topology but no security mechanism could prevent this parameter from illicit changes. An attacking node can easily alter the version number without authorization and trigger a needless rebuild of the entire DODAG graph, thus declining network resources.

To simulate a flooding attack, sky mote, and z1 mote were used. Of the 12 motes in this scenario, mote 1 is a UDP-server sky mote, mote 2-11 are UDP-client sky motes and mote 12 is the malicious z1 mote (flooding attacker mote). A flooding attacker floods the network with a significant volume of traffic, rendering nodes unavailable. This disruptive activity severely impacts the network’s performance and availability.

**Data Generation and Data Preparation Phase**

The data generated during the simulation of various attack scenarios are captured in .pcap files using the radio messages tool. The .pcap files corresponding to the sinkhole attack, version number attack, and flooding attack are stored as radiolog-1689230527100.pcap, radiolog-1689260705200.pcap, and radiolog-1689398858604.pcap respectively. The data is then analyzed and extracted from these .pcap files and stored as .csv files using another tool, called Wireshark. Now that the self-generated data is stored in .csv files we can preprocess and prepare the data to assure its quality and compatibility with the deep-learning algorithms. The preprocessing typically includes data cleaning, handling missing values, data encoding, feature selection, and data normalization. Figure 5 illustrates a snippet of the .csv file extracted from the .pcap file, showcasing data related to the flooding attack scenario. All the generated datasets are represented by features such as packet number, timestamp, source and destination IPv6 address, communication protocol, packet size, information related to ICMPv6, and corresponding information of the simulated network.

**Model Training Phase**

At this step, CNN-AO, CNN, and LSTM models were trained using the self-generated datasets obtained from the previous step. These deep learning models were chosen as they have the ability to learn patterns and representations directly from the data provided during the training process. They automatically learn features, reducing the need for manual feature engineering. Deep learning models use activation functions, introducing non-linearity into the network. This non-linearity allows them to learn complex relationships in the data. They use backpropagation algorithms to minimize the difference between predicted outputs and actual targets. This iterative process adjusts the model's parameters to improve its predictions. Deep learning models optimize their weights using gradient descent-based optimization algorithms. These algorithms find the optimal set of parameters that minimize the loss function. Deep learning models often benefit from specialized hardware accelerators like GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), which significantly speed up the training process. These attributes render them the ideal selection for experimental endeavors prior to implementing practical applications in the real world.

The CNN-AO model is a sequential CNN configuration featuring two convolution layers, a max pooling layer, flattening, and a dense layer with sigmoid activation. It employs a method where the Aquila optimization algorithm determines the values for the number of units and kernel size in the second convolutional layer, as well as pool size and strides in the max pooling layer. Abualigah et al. (2021) stated that the Aquila optimizer is an innovative population-based optimization algorithm that draws inspiration from the hunting behaviors of the...
Aquila in nature. Renowned for its exceptional visual acuity and hunting prowess, the Aquila bird efficiently captures prey. The Algorithm Operates (AO) in two main phases (Abualigah et al., 2020; 2021; Salcedo-Sanz, 2016):

- Diversification (exploration): In this stage, the algorithm creates random operators to investigate diverse regions within the search space
- Intensification (exploitation): In the subsequent phase, the algorithm concentrates on finding the optimal solution within the search space. An optimization procedure is applied to determine the most suitable values for different system parameters, enabling the system's design to be executed at minimal cost. Once the Aquila optimization algorithm identifies the best fit and solution for the parameters computed in the preceding step, a novel model is constructed accordingly

CNN employs convolutional layers to scan input data, capturing local features and spatial relationships. On the other hand, LSTMs are specialized for handling sequential data. LSTMs possess a unique memory cell that enables them to capture long-term dependencies and patterns in sequences. This makes them invaluable for tasks requiring an understanding of context and temporal relationships.

**Evaluation Phase**

After the models are trained, an evaluation of the model’s performance is done using the testing dataset for each of the attack scenarios. The effectiveness of various models namely, CNN-AO, CNN, and LSTM is evaluated using diverse metrics to gauge their ability to perform well on new, unseen data and generalize effectively. The performance of these models was evaluated on evaluation metrics namely: Accuracy, precision, recall, F1-score, sensitivity, specificity, and false alarm rate:

- **Accuracy**: This metric represents the ratio of correctly predicted instances to the total number of instances in a dataset. It serves as a measure of the overall correctness of the model's predictions
- **Precision**: It is the ratio of correctly predicted positive observations to the total predicted positives
- **Recall or sensitivity**: It measures the ability of a classification model to identify all the relevant instances (true positives) in a dataset. It calculates the ratio of correctly predicted positive observations to all the actual positives
- **F1-score**: It provides a balance between precision and recall when there is an uneven class distribution. It is a harmonic mean of precision and recall
- **Specificity**: It calculates the ratio of correctly predicted negative observations to all the actual negatives
- **False alarm rate**: It calculates the ratio of false positive predictions to all the actual negative instances. A lower false alarm rate indicates a better performance of the model is not misclassifying negative instances as positive

**Results and Discussion**

This section presents the CNN-AO, CNN, and LSTM model's performance. These models were implemented in Python using Jupyter Notebook with libraries such as pandas, numpy, Sk learn, Keras, and Tensorflow to name a few. Training included up to 5 epochs with early stopping.

Figures 6-8 depict the classification report for a sinkhole, version number, and flooding attack scenario respectively using the CNN-AO model.

The performance results of CNN-AO, CNN, and LSTM models for self-generated malicious datasets and others are summarized in Table 5.

From Table 5, we can conclude that the CNN-AO model performs better in terms of accuracy and false alarm rate as compared to CNN and LSTM models using the same datasets. Additionally, the standard deviation of all performance metrics within each self-generated malicious scenario for every model has been calculated and is summarized in Fig. 9.

The collective standard deviation of performance metrics across various scenarios for each model offers an assessment of the overall variability in the metrics’ performance.

Several conclusions can be drawn from Fig. 9. The flooding attack scenario shows relatively low variability (standard deviation) across most metrics, indicating consistent performance across different evaluation criteria. The version number attack scenario exhibits moderate variability in performance metrics, suggesting some fluctuations in precision and F1-score. The sinkhole attack scenario displays substantial variability across the evaluated metrics. This high standard deviation implies inconsistency in performance metrics, indicating challenges in accurately detecting and classifying sinkhole attacks based on the chosen criteria.

In this study, three custom malicious datasets were created utilizing the Contiki Cooja simulator. These datasets were generated through the simulation of three distinct attacks: Sinkhole, version number, and flooding attack. This section presents the findings related to the sensor map, the average power consumption, and the average radio duty cycle of the network obtained during the simulation of malicious network scenarios.

Figure 10 vividly demonstrates the impact of the sinkhole attack in the Cooja network simulation. In this attack, the sinkhole node pretends to be a legitimate destination and deceives other nodes into routing their traffic through it. Consequently, the victim nodes who select it as their parent are prevented from becoming part of the network. Nodes 2-5 and 10 are victim nodes in this case.
Table 5: Performance results of CNN-AO, CNN, and LSTM models for self-generated malicious datasets and others

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1-score (%)</th>
<th>Sensitivity (%)</th>
<th>Specificity (%)</th>
<th>FAR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-generated malicious</td>
<td>CNN</td>
<td>92.76</td>
<td>38.39</td>
<td>100.00</td>
<td>55.48</td>
<td>100.00</td>
<td>92.42</td>
<td>07.570</td>
</tr>
<tr>
<td>Scenario 1</td>
<td>LSTM</td>
<td>98.91</td>
<td>80.57</td>
<td>100.00</td>
<td>89.24</td>
<td>100.00</td>
<td>98.86</td>
<td>01.130</td>
</tr>
<tr>
<td>(Sinkhole attack)</td>
<td>CNN-AO</td>
<td>99.22</td>
<td>85.39</td>
<td>100.00</td>
<td>92.12</td>
<td>100.00</td>
<td>99.19</td>
<td>0.807</td>
</tr>
<tr>
<td>Self-generated malicious</td>
<td>CNN</td>
<td>93.41</td>
<td>47.83</td>
<td>99.26</td>
<td>64.56</td>
<td>99.26</td>
<td>93.03</td>
<td>06.960</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>LSTM</td>
<td>98.19</td>
<td>77.02</td>
<td>100.00</td>
<td>87.02</td>
<td>100.00</td>
<td>98.08</td>
<td>01.910</td>
</tr>
<tr>
<td>(Version number attack)</td>
<td>CNN-AO</td>
<td>99.77</td>
<td>96.36</td>
<td>100.00</td>
<td>98.14</td>
<td>100.00</td>
<td>99.75</td>
<td>02.242</td>
</tr>
<tr>
<td>Self-generated malicious</td>
<td>CNN</td>
<td>98.89</td>
<td>99.01</td>
<td>99.42</td>
<td>99.22</td>
<td>99.42</td>
<td>97.61</td>
<td>02.380</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>LSTM</td>
<td>99.47</td>
<td>99.83</td>
<td>99.41</td>
<td>99.41</td>
<td>99.41</td>
<td>99.61</td>
<td>0.380</td>
</tr>
<tr>
<td>(Flooding attack)</td>
<td>CNN-AO</td>
<td>99.55</td>
<td>99.96</td>
<td>99.40</td>
<td>99.68</td>
<td>99.40</td>
<td>99.91</td>
<td>0.084</td>
</tr>
<tr>
<td>Normal-hello flooding,</td>
<td>CNN</td>
<td>96.87</td>
<td>94.85</td>
<td>99.65</td>
<td>97.19</td>
<td>99.65</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamel and Elhamayed (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal selective forward,</td>
<td>CNN</td>
<td>96.02</td>
<td>99.61</td>
<td>84.59</td>
<td>91.49</td>
<td>84.59</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamel and Elhamayed (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal-sinkhole,</td>
<td>CNN</td>
<td>98.57</td>
<td>99.84</td>
<td>96.84</td>
<td>98.32</td>
<td>96.84</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamel and Elhamayed (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal-wormhole,</td>
<td>CNN</td>
<td>98.09</td>
<td>99.5</td>
<td>94.4</td>
<td>97.15</td>
<td>94.40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamel and Elhamayed (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal-version</td>
<td>CNN</td>
<td>90.40</td>
<td>95.05</td>
<td>80.08</td>
<td>88.56</td>
<td>80.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamel and Elhamayed (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6: Classification report for a sinkhole attack scenario using CNN-AO model

Fig. 7: Classification report for a version number attack scenario using the CNN-AO model

Fig. 8: Classification report for a flooding attack scenario using the CNN-AO model

Fig. 9: Standard deviation of performance metrics across different attack scenarios
Observations

Figure 11 illustrates a scenario involving a version number attack. Nodes 5-10 have selected node 12 as their parent to establish a connection with the sink. In this malicious setup, node 12 deceives neighboring nodes by propagating false routing information by manipulating or fabricating version numbers. This deceptive behavior leads to unnecessary resource wastage, as affected nodes may initiate unnecessary route repairs or topology updates in response to constant false advertisements of new versions.

Figure 12 depicts a flooding attack scenario where the malicious node 12 inundates the network with an overwhelming amount of malicious or unwanted traffic.

Average Power Consumption

It represents the average of the overall power consumption of each network node. Each node's total power consumption comprises the power used in Low Power Mode (LPM), power consumption during CPU operation (CPU), power consumption during transmission (radio transmission), and power consumption during listening (radio listening). The power consumption of each node within each malicious scenario is illustrated in Figs. 13-15 respectively.

Figure 13, the sinkhole attacking node prevents nodes 2-5 and 10 from joining the network. Consequently, these nodes were not taken into account when calculating the average power consumption.

Figure 14, depicts the version number attack, nodes 5-10 exhibit the highest power consumption compared to other nodes. This increased power usage is a result of these nodes selecting the malicious node 12 as their connection to the sink. Instead of facilitating the connection to the sink, the malicious node misleads them, causing these nodes to engage in unnecessary route repairs and topology updates due to continuous false information about new versions. This deceptive behavior leads to significant resource wastage within the network.
Illustrated in Fig. 15, the radio listening activity of nodes 5, 7, 8, and 9 stands out as the highest among the parameters utilized to calculate nodes' power consumption. This heightened radioactivity is due to the proximity of these nodes to the malicious node, which persistently sends data to these victim nodes.

**Average Radio Duty Cycle**

The term "average radio duty cycle" describes the average proportion of a node’s total active time over a certain period when its radio is active. It is calculated by dividing the total amount of time a node’s radio is on by the network’s total amount of simulation time. The average radio duty cycle of each node within each malicious scenario is presented in Figs. 16-18 respectively.

Figure 16, the radio listening values for nodes 6, 7, 8, 9, and 11 surpass the radio transmission values.

Figure 17, the radio transmit value is higher than the radio listen for all nodes.

Figure 18, the radio listening of nodes 5, 7, 8, and 9 is more as compared to radio transmission as they are the victim of malicious flooding attack node 12.

Further results obtained during the simulation of malicious network scenarios are summarized in Table 6. From Table 6, several conclusions can be drawn regarding the different malicious network scenarios:

- **Power consumption**: Malicious scenario 3 (flooding attack) exhibits significantly higher power consumption compared to the other scenarios. This indicates that flooding attacks demand more energy resources from the network nodes due to the continuous influx of data.

- **Duty cycles**: The duty cycles, especially the listen duty cycle, are considerably higher in malicious scenario 3 (flooding attack). This high listen duty cycle suggests that nodes are actively engaged in receiving data for a significant portion of the time, contributing to the increased power consumption in this scenario.

- **Packet discrepancies**: Malicious scenario 2 (version number attack) generates a higher number of attack packets compared to the other scenarios, indicating a more aggressive nature of the attack. This higher number of attack packets can potentially overwhelm the network, leading to disruptions in communication.

- **Normal packets**: Malicious scenario 1 (sinkhole attack) has a notably higher number of normal packets, suggesting that despite the attack, a substantial amount of legitimate communication is still occurring. This could imply that the sinkhole attack is more subtle in nature, selectively diverting specific traffic without completely disrupting the network.

- **Total number of packets**: Malicious scenario 3 (flooding attack) results in the highest total number of packets, indicating an extensive network activity due to the constant influx of both normal and attack packets. This overwhelming traffic can lead to network congestion and degradation of performance.
Table 6: Simulation results obtained during the simulation of various malicious network scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>CPU power</th>
<th>LPM power</th>
<th>Listen power</th>
<th>Transmit power</th>
<th>Average power consumption</th>
<th>Listen duty cycle</th>
<th>Transmit duty cycle</th>
<th>Number of normal packets</th>
<th>Number of attack packets</th>
<th>Total number of packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious scenario 1 (Sinkhole attack)</td>
<td>0.341</td>
<td>0.153</td>
<td>0.420</td>
<td>0.033</td>
<td>0.947</td>
<td>0.700</td>
<td>0.061</td>
<td>165152</td>
<td>7825</td>
<td>173004</td>
</tr>
<tr>
<td>Malicious scenario 2 (Version number attack)</td>
<td>0.812</td>
<td>0.139</td>
<td>2.045</td>
<td>2.591</td>
<td>5.587</td>
<td>3.408</td>
<td>4.880</td>
<td>177486</td>
<td>11103</td>
<td>188589</td>
</tr>
<tr>
<td>Malicious scenario 3 (Flooding attack)</td>
<td>1.102</td>
<td>0.130</td>
<td>7.275</td>
<td>2.331</td>
<td>10.837</td>
<td>12.124</td>
<td>4.390</td>
<td>67479</td>
<td>161593</td>
<td>229072</td>
</tr>
</tbody>
</table>

Fig. 17: Average radio duty cycle in case of version number attack

Fig. 18: Average radio duty cycle in case of a flooding attack

Conclusion

In this study, we addressed the crucial challenge of enhancing the security of IoT networks by focusing on the evaluation of IDS for IoT. The rapid expansion of IoT devices, while providing numerous advantages, has also introduced significant security and privacy concerns. To combat these challenges, we devised a meticulous methodology for generating IoT-specific datasets utilizing the Contiki Cooja simulator. Through controlled simulations, we replicated real-world IoT security threats, specifically sinkhole attacks, version number attacks, and flooding attacks, within the context of RPL networks.

Our contributions to this research encompassed several key aspects. First and foremost, we meticulously crafted malicious datasets, capturing the intricacies of each attack type, in order to create a foundation for evaluating the performance of intrusion detection models. Leveraging these self-generated datasets, we trained advanced deep-learning models including CNN-AO, CNN, and LSTM. Comparative analyses were conducted, highlighting the superior performance of the CNN-AO model in accurately classifying normal and malicious network traffic. This study incorporates seven evaluation measures accuracy, precision, recall, F1-score, sensitivity, specificity, and false alarm rate, and found that CNN-AO has an accuracy of 99.22, 99.77, and 99.55% for our self-generated malicious datasets: Sinkhole attack, version number attack and flooding attack respectively.

Our study not only advanced the field by introducing a robust methodology for IoT-specific dataset generation but also shed light on the critical importance of simulation fidelity in creating datasets that mirror real-world scenarios. This unique approach differentiated our research from existing studies in IoT security. By providing a comprehensive analysis of our experiments and their outcomes, we pave the way for future research endeavors aimed at strengthening IoT security. Furthermore, our study advocates for the continued
exploration and development of IoT-specific datasets to meet the evolving challenges posed by emerging technologies and communication protocols.

Acknowledgment

I would like to express my sincere gratitude to my guide, Dr. Sarvesh Tanwar, and co-guide Prof. (Dr.) Tanupriya Choudhury, for their invaluable guidance, unwavering support, and expert mentorship throughout the entire research process.

Funding Information

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author’s Contributions

Vandana Choudhary: Conducted background study and experiments, analyzed data, and wrote the manuscript.

Sarvesh Tanwar: Helped in drafting the manuscript critical insights, and provided valuable guidance and mentorship throughout the research process ensuring the paper's quality and rigor.

Tanupriya Choudhury: Helped in drafting the manuscript, critical insights and provided valuable guidance and mentorship throughout the research process ensuring the paper's quality and rigor.

Ethics

The corresponding author declared that this study has not been submitted elsewhere.

Future Work

This study marks a considerable advancement in the generation and classification of attacks in RPL networks, but there is still a need to perform further investigation by expanding the datasets to incorporate more attacks, different network topologies, and increased simulation time. The performance and reliability of the attack classification of the models can also potentially be increased by examining the features to a greater extent that were collected from the generated dataset and employing other deep-learning models. Additionally, assessing the trained model's performance in actual IoT deployments and confirming its efficacy against real-world threats will offer insightful information and further confirm the model's applicability and utility. We may contribute to the establishment of more reliable and accurate IDS for IoT by exploring these potential future research directions.

Data Availability

All data generated or analyzed during this study are included in this published article and are available at https://amityedu96491-my.sharepoint.com/:f:/g/personal/vandana_choudhary_s_amity_edu/Ep6C6iT-tFlNLoO46OHjqi10B4WOcbqlik7cwczy607Xluuw?e=LrJ0Hb.

Conflict of Interest

The authors declare that they do not have any conflicts of interest that influence the work reported in this study.

References

https://doi.org/10.3390/app10113827

https://doi.org/10.1016/j.cie.2021.107250

https://doi.org/10.1109/COMST.2015.2444095

https://doi.org/10.32604/iasc.2023.026799

https://doi.org/10.1016/j.jestch.2022.101322

https://doi.org/10.1007/s10586-021-03459-1

https://doi.org/10.1007/s11036-023-02105-x


