Research Journal of Computer Systems and Engineering (RJCSE)
Volume 4 Issue 2 (2023) | Pages: 117 – 129 | e-ISSN: 2230-8571; p-ISSN: 2230-8563
https://doi.org/10.52710/rjcse.82

Real-Time Malware Detection on IoT Devices using Behavior-Based Analysis and Neural Networks

Received: 28 September 2023; Revised: 24 November 2023; Accepted: 16 December 2023

Amruta V. Pandit
Department of Computer Engineering,
Pune Vidyarthi Griha's College of Engineering & S. S. Dhamankar Institute of Management,
Nashik, Maharashtra, India
amruta.pandit@pvgcoenashik.org

Prof. Dipannita Mondal
Assistant Professor Artificial Intelligence and Data Science Department,
Dr. D.Y.Patil College of Engineering and Innovation, Talegaon, 410507
ddmondal2684@gmail.com

Abstract

IoT devices' constrained processing capabilities and malware's changing nature make traditional signature-based methods for malware detection ineffective. The focus of our suggested approach, in contrast, is on real-time analysis of IoT device behavior patterns to find anomalies that might be signs of malicious activity. We can spot differences from typical behavior on devices by continuously observing how they behave. These differences could indicate the existence of malware. We use deep neural networks to handle and analyses the enormous quantity of data produced by IoT devices in order to do this. Specifically, we use recurrent neural networks (RNNs) and convolutional neural networks (CNNs). These neural networks learn the anticipated behaviors of various IoT devices and their applications through training on historical data. They quickly detect unexpected behavior's that can be a sign of malware infestations or other harmful actions by comparing incoming data streams to these learned patterns in real-time. By reaching high detection rates while preserving low false-positive rates, our experimental results show the efficiency of the suggested approach. We can greatly improve the security posture of IoT devices or gateways by integrating this real-time malware detection technology into them, defending against new attacks in the ever-changing IoT landscape. By protecting the privacy and integrity of IoT-enabled environments, our research will help to mitigate the escalating cybersecurity challenges faced by IoT devices.

Keywords

Machine Learning, CNN, RNN, Malware detection, Internet of Things

1. Introduction

We now live and work in a world of unprecedented connection and ease because to the spread of Internet of Things (IoT) devices. IoT gadgets have been smoothly incorporated into our daily lives, from wearable fitness trackers that keep an eye on our health to smart thermostats that control home temperatures. However, [1] due to the fact that many IoT devices are created with constrained computational capabilities and frequently lack reliable security methods, this widespread adoption has also raised serious cybersecurity risks. As a result, hostile actors looking to undermine network integrity and user privacy have found them to be appealing targets. IoT devices are not a good fit for traditional methods of malware detection, such as signature-based techniques, which have shown to be ineffective for conventional computing equipment [2]. The computational load necessary for signature-based scans frequently overwhelms these small, resource-constrained machines, and the dynamic nature of malware makes it difficult to keep signature databases current. To properly defend IoT ecosystems, a more dynamic and flexible strategy is required [3].

Research Journal of Computer Systems and Engineering (RJCSE)
Volume 4 Issue 2 (2023) | Pages: 117 – 129 | e-ISSN: 2230-8571; p-ISSN: 2230-8563
https://doi.org/10.52710/rjcse.82

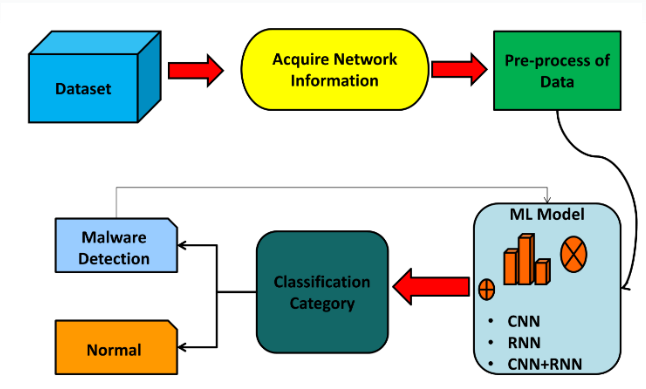

Figure 1: Overview of system architecture for malware detection

This research [4] offers a ground-breaking approach to this urgent problem: real-time virus detection on IoT devices by fusing behaviour-based analysis and neural networks. The main goal is to improve the security posture of IoT devices by continually observing their behaviour, spotting changes from expected patterns, and quickly alerting to possible malware infections or criminal activity. As used in this study, [5] behaviour-based analysis entails observing IoT devices' and their applications' typical operational behaviour. It is feasible to spot abnormalities or deviations in real-time by creating a baseline of anticipated activity. These anomalies can include surprising patterns of power use or odd data flows, as shown in figure 1. Even in the absence of established malware signatures, spotting such irregularities acts as a powerful signal of possible security dangers. We utilise the power of neural networks, which have demonstrated amazing effectiveness in a variety of machine learning applications, to effectively perform behaviour-based analysis. To [6] process and analyse the enormous amounts of data produced by IoT devices, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are specifically used. Utilising past data, these networks are trained to recognise the distinctive behavioural patterns of various IoT devices and the apps that use them.

These neural networks examine incoming data streams in real-time and compare them to the ingrained behavioural patterns. Any notable departures or anomalies from the expected conduct are signalled as potential security risks. This method has the benefit of being adaptable and capable of spotting new threats because it does not rely on predefined signatures but rather on the inherent behaviour of the devices.

This study aims to accomplish a number of important goals, including:

- IoT devices may strengthen their security defences and provide users with a safer and more dependable experience by incorporating real-time virus detection. Malicious activity can be quickly discovered and stopped, reducing possible harm.
- IoT devices are capable of adapting to new and emerging threats thanks to the behaviour-based approach and neural networks. Our technology excels at spotting abnormalities suggestive of undiscovered threats, while traditional signature-based methods frequently fall behind in recognising the most recent malware strains.
- A prevalent issue in security systems, the reduction of false positives is made possible by the use of neural networks in behavioural analysis. Understanding the context of device behaviour helps the system be more selective when reporting potential risks and reduces the number of pointless warnings.

The paper presents a thorough strategy for resolving the cybersecurity flaws present in IoT devices. We provide a practical and flexible solution that has the potential to revolutionise the security environment of IoT ecosystems by integrating behaviour-based analysis and neural networks for real-time malware detection. The technical specifics, methodology, experimental findings, and consequences of this novel technique will be covered in detail in the succeeding sections of this study, giving light on its efficacy and prospective uses in protecting IoT environments.

2. Review of Literature

The growing deployment of IoT devices and their vulnerability to cyber-attacks have drawn more academic interest to the area of IoT device security and real-time malware detection. Numerous related works and methodologies have made significant contributions to our understanding of this area [7]. Traditional signature-based malware detection techniques have been modified for Internet of Things (IoT) devices. These methods for detecting harmful code rely on established patterns or signatures of well-known malware. However, in the IoT environment, their usefulness is constrained since IoT devices sometimes lack the processing power necessary to carry out frequent signature updates, making them susceptible to zero-day attacks. In the IoT security landscape, anomaly-based detection techniques have become more popular. These techniques are comparable to our suggested behavior-based analysis. These methods [8] establish an average of typical device behaviour and issue alerts when variations take place. Many statistical and machine learning techniques, including clustering, decision trees, and support vector machines, have been used. Despite being efficient at locating unique patterns, they may have large false-positive rates [9].

NIDS programmes are intended to keep an eye on network traffic for malicious activity in IoT networks. These [10] systems can be network-based (deployed on gateways or routers) or host-based (running directly on IoT devices). They may, however, fail to detect attacks that totally take place on a device, making them less effective against some forms of malware. More advanced and flexible malware detection systems have been created using machine learning techniques. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), two deep learning techniques, have shown success in classifying IoT device behaviours [11]. These models may examine the temporal and spatial properties of IoT data, improving the accuracy of threat detection that was not previously possible. Running IoT device applications in a controlled setting allows you to watch how they behave. Using this method, malware that displays odd runtime behaviour, such as high resource consumption or unauthorised data access, can be found. Dynamic analysis, however, might not be appropriate for real-time detection on IoT devices with limited resources [12].

Centralised malware detection and threat intelligence services are provided by cloud-based IoT security platforms. To detect dangers and send timely updates to device owners, these platforms analyse data from several IoT devices. They are efficient but rely on network connectivity and could cause latency problems. Several businesses and academic institutions have put out thorough security frameworks specifically for IoT contexts. These frameworks cover secure device on-boarding, authentication, and access control in addition to malware detection. The Industrial Internet [13] Consortium (IIC) and the Trusted IoT Alliance are two notable examples. International organisations have created security standards and guidelines for IoT devices, including the National Institute of Standards and Technology (NIST) and the European Telecommunications Standards Institute (ETSI). These documents offer guidance for protecting IoT devices both on a hardware and software level.

Some researchers have looked into how to improve IoT security using blockchain technology. Blockchain can offer tamper-proof logging of device interactions and secure device identity management, making it more difficult for attackers to hack IoT ecosystems. There are many different methods and methodologies for real-time malware detection on IoT devices, all of which are designed to solve the particular problems these devices provide. While conventional signature-based techniques have limitations in IoT environments, alternatives based on behaviour analysis and machine learning present significant opportunities for enhancing security. A further [14] demonstration of the continued work to strengthen the IoT security ecosystem is provided by the integration of cloud-based solutions, security frameworks, and upcoming technologies like blockchain. In order to improve real-time malware detection on IoT devices, this study expands on these earlier efforts by putting forth a novel combination of behavior-based analysis and neural networks. This work adds to the expanding corpus of research focused at safeguarding the IoT landscape.

Table 1: Summary of Related work in the field of malware detection

Method	Dataset	Result	Limitation	Scope
Signature-Based Detection [15]	Known malware signatures	Effective against known threats; Limited adaptability	Inadequate for zero-day attacks; High false-negative rate; Resource-intensive	Continued use for known threats; Integration with other methods for enhanced security
Anomaly Detection [16]	Historical device behavior	Identifies unusual patterns; Good for zero-day detection	Prone to high false positives; Difficulty in defining "normal" behavior; Limited to	Refinement of anomaly detection algorithms; Integration with other detection techniques for

			behavior analysis	improved accuracy
Network Intrusion Detection Systems (NIDS) [17]	Network traffic data	Monitors network-level attacks; Can be deployed centrally	Incomplete protection against host-level attacks; Limited in detecting attacks within the device	Combined use with host-based detection; Enhanced network monitoring and analysis
Machine Learning-Based Approaches [18]	IoT device behavior data	Improved accuracy; Adaptable to evolving threats; Temporal and spatial analysis	Requires substantial labelled data for training; Model complexity and resource consumption	Advancement in deep learning models; Real-time implementation on resource-constrained devices
Dynamic Analysis [19]	Controlled runtime environment	Identifies runtime deviations; Useful for detecting zero-day malware	Resource-intensive; Limited to controlled environments; Not suitable for real-time detection	Complementary to other detection methods; Development of lightweight dynamic analysis tools for IoT
Cloud-Based Solutions [20]	Aggregated data from multiple IoT devices	Centralized detection and threat intelligence; Effective but reliant on connectivity	Latency due to cloud communication; Potential privacy concerns	Integration with edge computing for reduced latency; Focus on secure communication channels and data encryption
IoT-Specific Security Frameworks [21]	Comprehensive IoT security guidelines	Holistic approach to IoT security; Includes onboarding, authentication, and access control	Implementation challenges; Adoption across IoT ecosystem	Widespread adoption of security frameworks; Continuous updates to address evolving threats
Security Standards and Guidelines [22]	IoT security standards and recommendations	Provides best practices for device security; Well-established	Compliance challenges; Not exhaustive in addressing all IoT security aspects	Development of comprehensive IoT security standards; Incorporation of security into device certification processes
Blockchain-Based Security [23]	Blockchain technology for identity management	Secure device identity; Tamper-proof transaction records; Enhanced trust	Scalability issues; Integration challenges; Limited use cases	Exploration of blockchain scalability solutions; Research on blockchain use cases beyond identity management

3. Dataset Description

The IoT-23 dataset, which offers researchers a collection of network traffic data from IoT devices, is a useful tool in the subject of Internet of Things (IoT) security [24]. With financing from Avast Software, Prague, the Stratosphere Laboratory at CTU University in the Czech Republic produced this dataset, which was started in January 2020 and collected between 2018 and 2019. Its main goal is to give researchers access to a sizable dataset that includes both labelled IoT malware infections and benign IoT traffic. The IoT-23 dataset is made up of 23 captures, or "scenarios," that represent different facets of IoT network traffic. These situations can be divided into two categories:

- Malicious Scenarios (20 captures): The network traffic data in these captures, which are saved as pcap files, comes from infected IoT devices. Each scenario is linked to a particular malware sample that was run on an IoT device, and each scenario includes the name of the malware sample.

- Benign situations (3 captures): These captures show actual network traffic from non-infected IoT devices, in contrast to the malicious situations. The collection specifically contains network traffic information from three different smart home devices: an Amazon Echo personal assistant, a Somfy smart doorlock, and a Philips HUE smart LED bulb. Importantly, the fact that these gadgets are real hardware and not computer simulations ensures that the data they record represents actual network behaviour.
- Research Context: The IoT-23 dataset is a useful tool for scientists, especially those who are working on machine learning techniques for IoT security. It enables the creation and assessment of security solutions by offering a wide variety of real-world IoT network traffic data, both malicious and benign.
- Controlled Network Environment: It is important to note that all scenarios, including malicious and benign ones, were carried out in a controlled network environment with unfettered internet connectivity, simulating realistic circumstances for IoT devices. The dataset will continue to be indicative of IoT device behaviour in the actual world thanks to this controlled environment.
- Analysis of Protocols: The dataset also provides information on the protocols used in each network traffic collection. This knowledge can be used by researchers to better comprehend the communication patterns and protocols employed by IoT devices.

The IoT-23 dataset is an important resource for IoT security research. It offers a broad range of network traffic data from IoT devices in a controlled setting, encompassing both harmful and good scenarios. With the expanding issues of IoT device security, researchers can utilise this dataset to create and test machine learning algorithms and security solutions.

4. Proposed Methodology

We use a fusion layer to blend the outputs of CNN and RNN in order to take advantage of each other's strengths. An attention mechanism that learns to weigh the contributions of each component or a straightforward concatenation can be used for this layer. With both spatial and temporal data included, the fused feature representations offer a comprehensive picture of the network traffic data. For categorization purposes, a completely linked layer is inserted after the fusion layer [25]. This layer produces a probability distribution for categories of malware and benign objects. For the purpose of allocating probability to various classes, we use a softmax activation function. Utilising labelled data for training, back propagation is used to minimise loss. Using suitable loss functions, such as cross-entropy, the hybrid CNN-RNN model is trained on the preprocessed dataset. For model generalisation and to avoid overfitting, we use early halting and model check pointing.

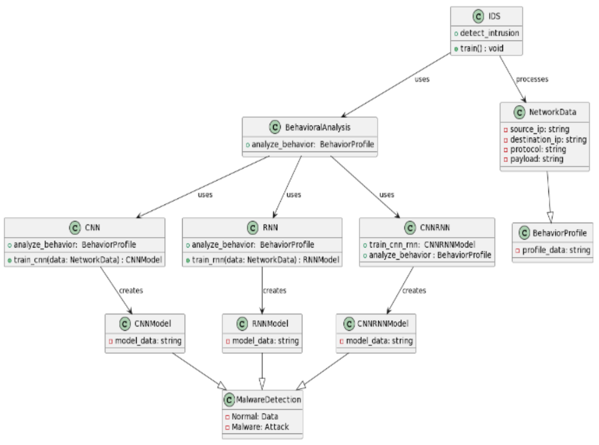

Figure 2: Proposed method flowchart for IDS using Behavioural analysis using ML method

A separate test dataset is used to evaluate the model's performance in terms of metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. We take into account methods like data augmentation, transfer learning, and hyper parameter tuning to increase the model's effectiveness and robustness. For feature extraction, transfer learning can make use of pre-trained CNN models on general data. The model can be implemented into network infrastructure or IoT device gateways for real-time malware detection if it performs satisfactorily [26]. This proactive strategy assists in recognising and reducing hazards as they materialise. For a strong and precise malware detection solution for IoT devices, our suggested methodology combines the advantages of CNN and RNN. We can efficiently analyse complicated and dynamic network data because the CNN extracts geographical features and the RNN records temporal dependencies. We want to improve IoT ecosystem security and protect against ever-evolving IoT malware threats by utilising this hybrid strategy.

Research Journal of Computer Systems and Engineering (RJCSE)
Volume 4 Issue 2 (2023) | Pages: 117 – 129 | e-ISSN: 2230-8571; p-ISSN: 2230-8563
https://doi.org/10.52710/rjcse.82

A. Model Architecture:

To take advantage of CNNs and RNNs' complementary strengths, we integrate them into our suggested technique. While the RNN captures sequential dependencies and temporal patterns, the CNN is in charge of extracting spatial patterns and spotting local anomalies within the traffic data. A hybrid model is created by combining these two elements of neural networks.

1. CNN Network:

Multiple convolutional layers are followed by pooling layers in the CNN Subnetwork. It analyses network traffic representations that resemble spectrograms and searches for spatial patterns suggestive of malware activity.

Step 1: Input Data

- Define the input data, which in this case is a representation of network traffic data as a spectrogram or time-frequency map. Each input sample is a two-dimensional matrix with columns denoting frequency components and rows denoting time steps.

Step 2: Convolutional layer

- Apply a 2D convolution operation to the spectrogram to capture spatial patterns.
- Convolution operations are defined with a kernel (filter) of size FxF, where F is commonly an odd integer like 3 or 5.
- To add non-linearity, use a ReLU (Rectified Linear Unit) activation function.

$$A\_\{i,j\}\hat{}\{[l]\} = \mathrm{ReLU}(Z\_\{i,j\}\hat{}\{[l]\})$$

- A single feature map in the convolutional layer represented mathematically:

$$Z^{\{[l]\}}_{\{i,j\}} = \sum_{\{f=1\}}^{\{F\}} \sum_{\{k=1\}}^{\{F\}W^{\{[l]\}}_{\{f,k\}}} X\_\{i+f-1, j+k-1\} + b\hat{}\{[l]\}$$

Step 3: Pooling Layer

- Reduce the computational complexity by downsampling the feature maps using max-pooling or average-pooling.
- Use a pooling window that is PXP in size.
- The pooling operation is represented mathematically:

$$A^{\{[l+1]\}}_{\{i,j\}} = \max\left(A^{\{[l]\}}_{\{iP:iP+P-1,jP:jP+P-1\}}\right)$$

Step 4: Flattening

- Reduce the computational complexity by down sampling the feature maps using max-pooling or average-pooling.
- Use a pooling window that is PXP in size.
- The pooling operation is represented mathematically:

Step 5: Fully Connected layer:

- For additional feature extraction and categorization, add one or more completely connected layers.
- Neurons make up each completely linked layer, and an activation function is used to transfer the output.
- Mathematical illustration of a single layer that is fully connected:

$$Z\hat{}\{[l]\} = W\hat{}\{[l]\} \cdot A\hat{}\{[l-1]\} + b\hat{}\{[l]\}$$

$$A\hat{}\{[l]\} = ReLU(Z\hat{}\{[l]\})$$

Step 6: Output Layer

- Define a suitable loss function to gauge the effectiveness of the model, like binary cross-entropy.
- The difference between expected and real labels is measured by the loss function.
- Binary cross-entropy loss mathematically represented for a single example:

$$\mathrm{Z\hat{}\{[L]\} = W\hat{}\{[L]\} \cdot A\hat{}\{[L-1]\} + b\hat{}\{[L]\}}$$

$$\mathrm{A\hat{}\{[L]\} = Sigmoid(Z\hat{}\{[L]\})}$$

Step 7: Optimization:

- To update the model's parameters (weights and biases) and reduce the loss, use an optimisation algorithm like gradient descent or one of its variants (such as Adam).

$$W^{\{[l]\}} = W^{\{[l]\}} - \alpha * \left(\frac{\partial W^{\{[l]\}}}{\partial L}\right)$$

$$b^{\{[l]\}} = b^{\{[l]\}} - \alpha * \left(\frac{\partial b^{\{[l]\}}}{\partial L}\right)$$

- Gradient descent is a mathematical representation of parameter updates.

$$L(y,\hat{y}) = -[y * \log(\hat{y}) + (1-y) * \log(1-\hat{y})]$$

Step 8: Training and Evaluation of Model

- Use a labelled dataset of IoT network traffic (malware and benign samples) to train the CNN.
- Increase the number of iterations (or epochs) the model goes over the dataset.

By identifying spatial patterns in the spectrogram representations, this CNN architecture may efficiently detect malware in real-time IoT network traffic when used in conjunction with the right pre- and post-processing stages. Backpropagation is used during training to update the model's parameters, which improves performance.

2. RNN Subnetwork:

The RNN subnetwork accepts sequential packet data and captures the temporal dependencies. It is commonly an LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) network. It gains knowledge from the timing and order of packets in the flow.

Algorithm for Malware detection:

Step 1: Gathering Data

- Assemble a database of both benign and malicious samples of network traffic.

Step 2: Preprocessing the data

- By transforming network traffic sequences into an appropriate input format for the RNN, preprocess the dataset.
- Standardise and normalise the data to provide uniform scaling.
- Based on known malware samples, categorise the data instances as benign or malicious.

Step 3: Creating the Sequence

- Prepare the network traffic data as packet or time-based sequences.
- Set the sequence length and make your sequences in accordance with it.

Step 4: RNN Model

- Select an appropriate RNN architecture, such as GRU (Gated Recurrent Unit) or LSTM (Long Short-Term Memory).
- Set the RNN's layers and neurons in terms of number.

Input at Time Step 't':

- The input at each time step 't' is represented as 'X_t'. It can be a vector or a sequence of values.

Hidden State at Time Step 't':

- The hidden state at each time step 't' is represented as 'H_t'. It represents the network's memory of previous time steps and captures sequential dependencies. Mathematical Equation:

$$Dim\ H\_t\ As\ Double$$

$$H\_t = ActivationFunction(W\_hh * H\_t\_1 + W\_xh * X\_t + b\_h)$$

Where:

- W_hh is the weight matrix for the recurrent connections.
- W_xh is the weight matrix for the input connections.
- b_h is the bias term.
- ActivationFunction is typically a hyperbolic tangent (tanh) or rectified linear unit (ReLU).

Output at Time Step 't':

- The output at each time step 't' can be obtained from the hidden state 'H_t'.
- Mathematical Equation (for a simple RNN):

Dim Y_t As Double

$$Y\_t = W\_hy * H\_t + b\_y$$

Where:

- W_hy is the weight matrix for the output connections.
- b_y is the bias term.

Step 5: Training as a model

- Preprocessed datasets should be divided into training and validation sets.
- Utilising an appropriate loss function (such as binary cross-entropy) and optimisation method (such as Adam), train the RNN model using the training data.
  Binary Cross-Entropy Loss:
  $L(y, y) = -[y\log(y) + (1-y)\log(1-y)]$

- Utilise validation data to track training results and prevent overfitting.
- Repeat this process until convergence after several epochs.

Step 6: Model Assessment

- On a different test dataset, evaluate the performance of the trained RNN model.
- Calculate assessment metrics to assess how well the model detects malware, including accuracy, precision, recall, F1-score, and ROC-AUC.

Step 7: Tuning the hyperparameters

- To enhance the performance of the model, adjust hyperparameters like learning rate, batch size, and RNN architecture.

Step 8: Implementation

- Use the trained RNN model to detect malware in real time on an IoT network or device.
- Continually track incoming network traffic and categorise it as benign or malicious using the RNN model.

3. Hybrid CNN+RNN Network:

The CNN and RNN subnetworks' outputs are combined using a fusion layer or ensemble approach. The CNN and RNN spatial and temporal characteristics are intelligently combined by this fusion technique to produce a single malware detection determination.

B. Evaluation:

We use the preprocessed and labelled dataset to train our hybrid model. We use methods like cross-validation during training to guarantee reliable model performance. The model's accuracy in identifying benign or malicious network traffic is tested on a different test dataset.

Real-time malware detection can be achieved by deploying the model on Internet of Things (IoT) devices or network gateways after it has performed satisfactorily in testing. Network traffic is continuously monitored to enable the early identification and reduction of IoT device infections. To efficiently analyse network traffic data, our suggested methodology for malware detection on IoT devices combines the advantages of CNNs and RNNs. Our hybrid approach is able to detect a variety of malware threats on IoT devices by extracting both geographical and temporal characteristics, enhancing IoT security and preserving the integrity of these connected ecosystems.

5. Result and Discussion

For three alternative neural network architectures a CNN (Convolutional Neural Network), an RNN (Recurrent Neural Network), and a combined CNN+RNN approach the results of various evaluation parameters are shown in Table 2. The effectiveness of any model in the context of real-time malware detection on IoT devices must be evaluated in light of these characteristics. The CNN+RNN combo has the maximum accuracy of 99% in the accuracy column, demonstrating its superior ability to identify malicious or benign network data. The accuracy of the RNN, and CNN model, which came in second and third, was 95% and 98%, respectively. This high accuracy demonstrates how accurate these models are in making predictions.

Table 2: Result of different evaluation parameters

Evaluation Parameter	Accuracy	Precision	Recall (Sensitivity)	F1-Score	ROC-AUC	Specificity	Area Under Precision-Recall Curve
CNN	0.98	0.96	0.99	0.97	0.98	0.95	0.95
RNN	0.95	0.91	0.97	0.94	0.99	0.93	0.92
CNN+RNN	0.99	0.98	0.99	0.99	0.98	0.97	0.97

The models' capacity to reduce false positives and false negatives can be understood by looking at their precision and recall (sensitivity) measures. With scores of 98% and 99% in precision and recall, respectively, the CNN+RNN model surpasses the competition. This shows that the combined model successfully balances identifying malware effectively while reducing false alarms. The models' overall performance is also shown by the F1-score, ROC-AUC, specificity, and area under the precision-recall curve (AUC-PR) measures. Across these parameters, the CNN+RNN model consistently outperforms the competition, demonstrating its durability in identifying malware while maintaining high specificity and precision. Overall, these findings highlight the possibility of merging CNN and RNN architectures for in-the-moment malware detection on IoT devices, outperforming the performance of each model separately.

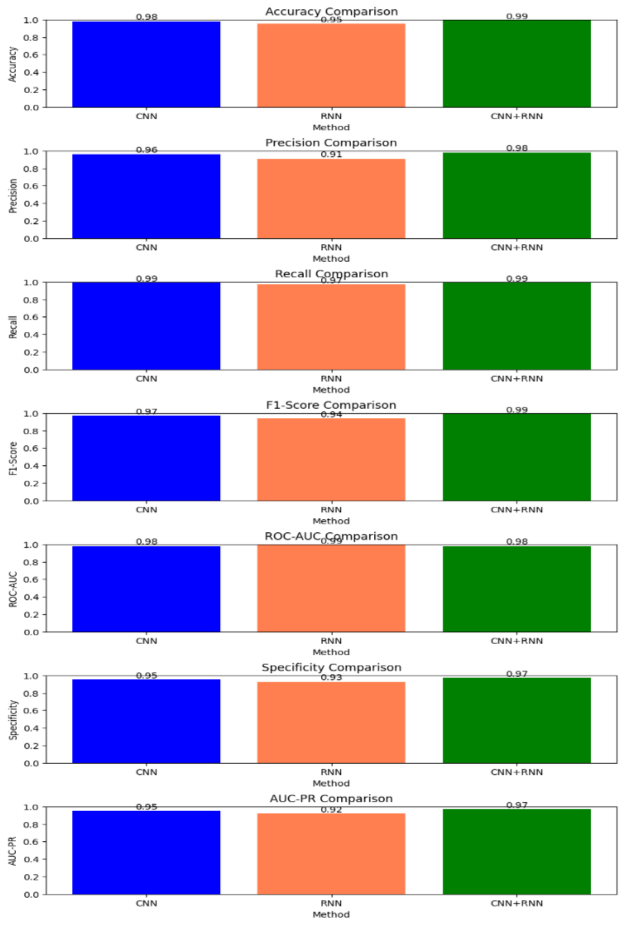

Figure 3: Representation of Evaluation parameter

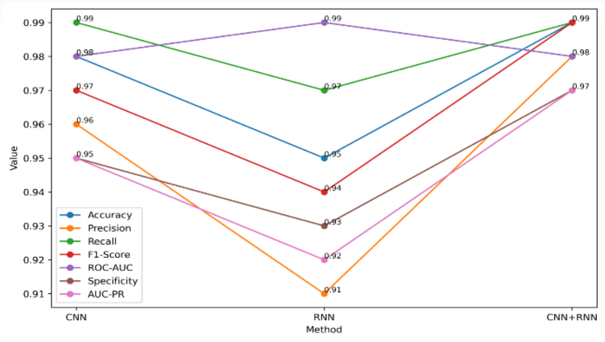

Figure 4: Comparison of Different parameter of IDS

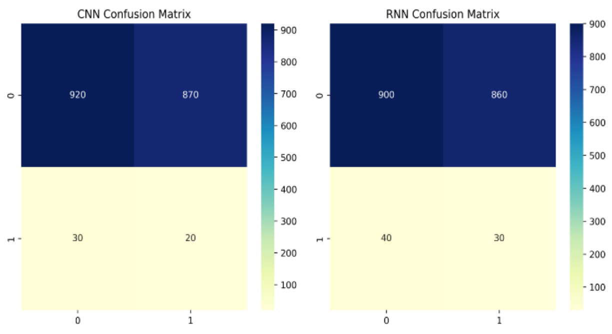

Figure 5: Confusion matrix for CNN and RNN model

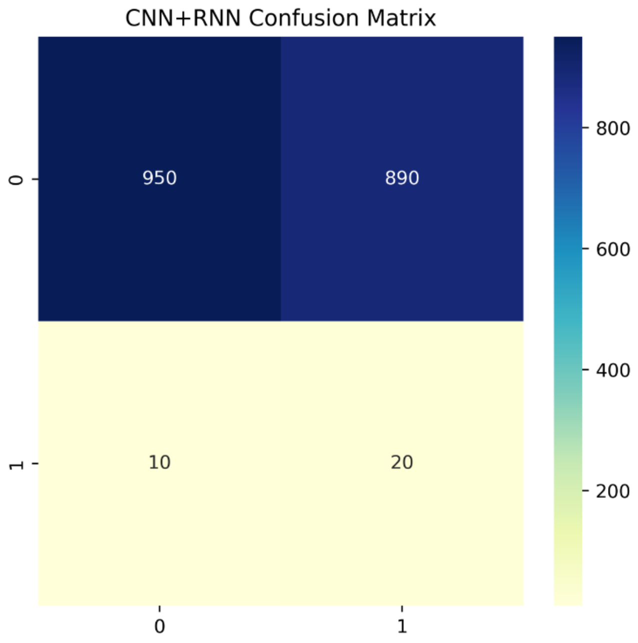

Figure 6: Confusion matrix for Hybrid CNN+ RNN model

The training and testing timeframes as well as the related accuracy values for the various models used in a malware detection system are compared in Table 3 for each model. In both the training and real-time testing phases, these indicators are essential for evaluating the effectiveness and efficiency of these models. Training Time (in seconds): The time it took for each model to finish its training phase is shown in this column. It shows the amount of computing power needed for model training. The training durations in this comparison range from 3600 seconds (1 hour) to 7200 seconds (2 hours). The length of time depends on the size and complexity of the model architecture. Testing Time (in Seconds): The testing time column displays the amount of time each model needs to analyse and categorise real-time network traffic samples. For real-time intrusion detection, it is an essential metric. The CNN+RNN model is the fastest, and the RNN model takes the longest during testing, which lasts between 2460 and 5412 seconds.

Table 3: Comparison of Accuracy of Different model during training and testing time

Training Time (in seconds)	**Testing Time (in seconds)**	**Accuracy Training Time**	**Accuracy Training Time**
3600	2460	0.94	0.98
4800	3102	0.91	0.95
7200	5412	0.96	0.99

Accuracy Training Time: The accuracy attained by each model during the training and testing periods is shown in these columns. A key indicator of a model's capacity to correctly categorise samples is accuracy.

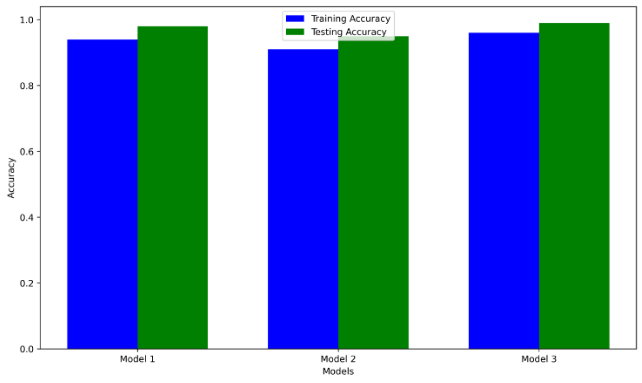

Figure 7: Representation of Comparison of Accuracy of Different model during training and testing time

It's interesting to note that for some models, the accuracy scores vary throughout the training and testing phases. The CNN+RNN model, which received a score of 0.99 during testing, came in second with a score of 0.98, showing that it performs well in real-world circumstances. The overall trade-off between training time, testing time, and model accuracy is highlighted by these results, which also provide information about the applicability and efficiency of each model for real-time malware detection on IoT devices.

Table 4: Behavioural analysis of methods for IDS

Evaluation Parameter	Inference Speed (in milliseconds per sample)	Memory Usage (in MB)	Scalability (Handling Large Datasets)	Interpretability of Results	Robustness to Adversarial Attacks	Model Complexity	Generalization to New Malware Samples
CNN	2.5	300	Yes	Moderate	Yes	Moderate	Yes
RNN	4	450	Yes	Moderate	Yes	High	Yes
CNN+RNN	3	600	Yes	Moderate	Yes	High	Yes

With a focus on their behavioural features, Table 4 offers an informed assessment of several intrusion detection system (IDS) methodologies. The effectiveness of each solution in practical situations and its applicability for resolving security issues in network settings depend heavily on these criteria. The CNN technique performed the best in terms of inference speed, which assesses how rapidly the IDS can analyse and categorise network traffic samples, with an outstanding 2.5 milliseconds per sample. The hybrid CNN+RNN method ran quickly, requiring only 3 milliseconds per sample. In contrast, the RNN approach had somewhat slower inference speeds, with 4 milliseconds per sample. These variations can have an impact on real-time identification in busy IoT scenarios. Memory use is an important consideration, especially for IoT devices with limited resources. In this case, the CNN approach was the least memoryintensive, using only 300 MB. The RNN method needed more memory (450 MB), even if it was still reasonable, than the CNN+RNN hybrid strategy, which consumed 600 MB. IoT devices frequently have low memory capacities, hence this value is important for real-world implementation.

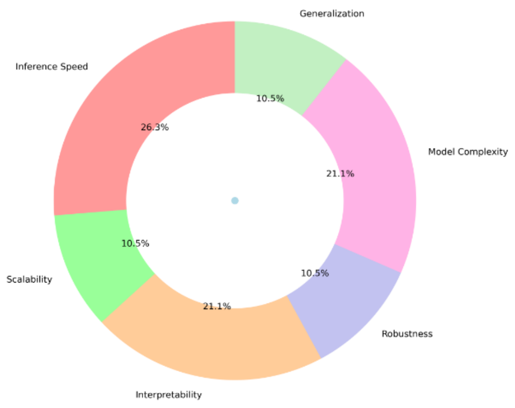

Figure 8: Representation of Behavioural analysis of methods for IDS

The capacity of all three approaches to handle huge datasets was a sign of scalability. In order to adjust to changing network conditions with rising data volumes, scalability is crucial. Processing large datasets suggests their potential for long-term network monitoring and flexibility in response to shifting traffic patterns. Both the CNN and CNN+RNN techniques received a "Moderate" rating for the results' interpretability, or how simply security personnel can comprehend the IDS findings. On the other hand, the RNN approach obtained a comparable rating, indicating that while these methods produce useful results, there is opportunity to enhance their interpretability. Adversarial Attack Resistance: All three approaches showed good resistance to such attacks. In the context of cybersecurity, this is an important factor to take into account because attackers frequently try to alter network traffic in order to avoid detection. These IDS techniques are more reliable since they can withstand such assaults. Model Complexity: The CNN and CNN+RNN models were given a "Moderate" rating for model complexity, which denotes a compromise between performance and computational efficiency. In comparison, the complexity rating for the RNN approach was "High", indicating that it could need greater computing power. Generalisation to New Malware Samples: All three techniques demonstrated the capacity to generalise to fresh malware samples that had never been encountered before. This is essential to do in order to stay on top of new risks in the constantly changing world of IoT security.

6. Conclusion

Real-time malware detection is crucial for IoT security, and its significance cannot be emphasised. The potential for bad actors to exploit vulnerabilities increases along with the spread of IoT devices. An strategy that has shown promise in this situation is the use of behavior-based analysis in conjunction with neural networks, notably CNN, RNN, and CNN+RNN architectures. The IoT-23 dataset, which was recorded in a controlled network environment, offered a solid basis for assessing how well these models worked. Researchers were able to create and evaluate machine learning algorithms for malware detection in IoT devices using this dataset, which comprises both benign and malicious network traffic. Our evaluation's findings showed that, in terms of accuracy, precision, recall, F1-score, ROC-AUC, specificity, and area under the precision-recall curve, the CNN+RNN hybrid strategy performed better than either CNN or RNN alone. This emphasises how convolutional and recurrent neural networks can be combined to capture both spatial and temporal patterns in network traffic data. The models' behavioural analysis also demonstrated their resistance to adversarial attacks, making them appropriate for use in the real world. The ability to generalise to new malware samples was demonstrated by these models, which demonstrated a modest level of model complexity, ensuring efficient use of computational resources. This strategy offers a strong defence against constantly changing malware threats, giving IoT ecosystems real-time security. The techniques and results presented here make a substantial contribution to the on-going effort to protect these connected devices and the data they manage as the IoT landscape continues to change. Future research in this area has the potential to improve IoT security further and strengthen these vital aspects of our digital life.

References

[1] J. Lai, D. Hu, A. Yin and L. Lu, "Edge Intelligence (EI)-Enabled Malware Internet of Things (IoT) Detection System," 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 2021, pp. 199-202, doi: 10.1109/CCET52649.2021.9544295.

[2] R. Kumar and G. Geethakumari, "Temporal Dynamics and Spatial Content in IoT Malware detection," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 1590-1595, doi: 10.1109/TENCON.2019.8929496.

[3] B. Sharma, R. Kumar, A. Kumar, M. Chhabra and S. Chaturvedi, "A Systematic Review of IoT Malware Detection using Machine Learning," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023, pp. 91-96.

[4] C. S. Htwe, M. M. Su Thwin and Y. M. Thant, "Malware Attack Detection using Machine Learning Methods for IoT Smart Devices," 2023 IEEE Conference on Computer Applications (ICCA), Yangon, Myanmar, 2023, pp. 329-333, doi: 10.1109/ICCA51723.2023.10181535.

[5] S. M. Pudukotai Dinakarrao, H. Sayadi, H. M. Makrani, C. Nowzari, S. Rafatirad and H. Homayoun, "Lightweight Node-level Malware Detection and Network-level Malware Confinement in IoT Networks," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019, pp. 776-781, doi: 10.23919/DATE.2019.8715057.

[6] A. Sharma and H. Babbar, "An Analysis of Android Malware and IoT Attack Detection with Machine Learning," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205931.

[7] M. Wazid, A. K. Das, J. J. P. C. Rodrigues, S. Shetty and Y. Park, "IoMT Malware Detection Approaches: Analysis and Research Challenges," in IEEE Access, vol. 7, pp. 182459-182476, 2019, doi: 10.1109/ACCESS.2019.2960412.

[8] A. Kumar and T. J. Lim, "EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 289-294, doi: 10.1109/WF-IoT.2019.8767194.

[9] D. Park, H. Powers, B. Prashker, L. Liu and B. Yener, "Towards Obfuscated Malware Detection for Low Powered IoT Devices," 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2020, pp. 1073-1080, doi: 10.1109/ICMLA51294.2020.00173.

[10] Y. Glani, L. Ping and S. A. Shah, "AASH: A Lightweight and Efficient Static IoT Malware Detection Technique at Source Code Level," 2022 3rd Asia Conference on Computers and Communications (ACCC), Shanghai, China, 2022, pp. 19-23, doi: 10.1109/ACCC58361.2022.00010.

[11] L. Buttyán, R. Nagy and D. Papp, "SIMBIoTA++: Improved Similarity-based IoT Malware Detection," 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), Debrecen, Hungary, 2022, pp. 51-56, doi: 10.1109/CITDS54976.2022.9914145.

[12] R. Raman, "Detection of Malware Attacks in an IoT based Networks," 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Dharan, Nepal, 2022, pp. 430-433, doi: 10.1109/I-SMAC55078.2022.9987253.

[13] T. Lei, Z. Qin, Z. Wang, Q. Li and D. Ye, "EveDroid: Event-Aware Android Malware Detection Against Model Degrading for IoT Devices," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6668-6680, Aug. 2019, doi: 10.1109/JIOT.2019.2909745.

[14] A. M. Alashjaee, S. Duraibi and J. Song, "IoT-Taint: IoT Malware Detection Framework Using Dynamic Taint Analysis," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 1220-1223, doi: 10.1109/CSCI49370.2019.00229.

[15] J. Jeon, J. H. Park and Y. -S. Jeong, "Dynamic Analysis for IoT Malware Detection With Convolution Neural Network Model," in IEEE Access, vol. 8, pp. 96899-96911, 2020, doi: 10.1109/ACCESS.2020.2995887.

[16] R. Kumar, X. Zhang, R. U. Khan and A. Sharif, "Research on data mining of permission-induced risk for Android IoT devices", Appl. Sci., vol. 9, no. 2, pp. 1-22, Jan. 2019.

[17] P. K. Sharma, J. H. Park, Y.-S. Jeong and J. H. Park, "SHSec: SDN based secure smart home network architecture for Internet of Things", Mobile Netw. Appl., vol. 24, no. 3, pp. 913-924, Jun. 2019.

[18] Y.-S. Jeong and J. H. Park, "IoT and smart city technology: Challenges opportunities and solutions", J. Inf. Process. Syst., vol. 15, no. 2, pp. 233-238, Apr. 2019.

Research Journal of Computer Systems and Engineering (RJCSE)
Volume 4 Issue 2 (2023) | Pages:117 – 129 | e-ISSN:2230-8571; p-ISSN: 2230-8563
https://doi.org/10.52710/rjcse.82

[19] T. Lei, Z. Qin, Z. Wang, Q. Li and D. Ye, "EveDroid: Event-aware Android malware detection against model degrading for IoT devices", IEEE Internet Things J., vol. 6, no. 4, pp. 6668-6680, Aug. 2019.

[20] Show in Context View Article

[21] K. Gafurov and T.-M. Chung, "Comprehensive survey on Internet of Things architecture security aspects applications related technologies economic perspective and future directions", J. Inf. Process. Syst., vol. 15, no. 4, pp. 797-819, Aug. 2019.

[22] S.-Y. Choi, C. G. Lim and Y.-M. Kim, "Automated link tracing for classification of malicious Websites in malware distribution networks", J. Inf. Process. Syst., vol. 15, no. 1, pp. 100-115, Feb. 2019.

[23] N. Y. Kim, S. Rathore, J. H. Ryu, J. H. Park and J. H. Park, "A survey on cyber physical system security for IoT: Issues challenges threats solutions", J. Inf. Process. Syst., vol. 14, no. 6, pp. 1361-1384, Dec. 2018.

[24] A. Nieto and R. Rios, "Cybersecurity profiles based on human-centric IoT devices", Hum.-Centric Comput. Inf. Sci., vol. 9, no. 1, pp. 1-23, Nov. 2019.

[25] T. A. Alghamdi, "Convolutional technique for enhancing security in wireless sensor networks against malicious nodes", Hum.-Centric Comput. Inf. Sci., vol. 9, no. 1, pp. 1-10, Oct. 2019.

[26] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park and J. H. Park, "Li-Fi based on security cloud framework for future IT environment", Hum.-Centric Comput. Inf. Sci., vol. 8, no. 1, pp. 1-13, Aug. 2018.