An AI-Based Network Forensic Readiness Framework for Resource-Constrained Environments

Syed Rizvi[1][0000−0003−1938−8036], Mark Scanlon[2][0000−0002−6581−7164], Jimmy McGibney[1][0000−0003−4541−1420], and John Sheppard[1][0000−0003−4212−0329]

[1] South East Technological University, Waterford, Ireland
Syed.Rizvi@postgrad.wit.ie, {Jimmy.McGibney, John.Sheppard}@setu.ie
[2] School of Computer Science, University College Dublin, D04 V1W8, Ireland
mark.scanlon@ucd.ie

Abstract. In recent years, the adoption of Internet of Things (IoT) devices has transformed industries and daily life. However, the integration of real-time services and internet connectivity increases the risk of attackers exploiting network vulnerabilities. Investigating such vulnerabilities in Resource-Constrained Environments (RCEs) poses challenges due to limited computational capacity, power constraints, and the heterogeneity of IoT-generated data and traffic. To address these issues, this study proposes a framework integrating optimised artificial intelligence models trained on the CICIoT2023 and CSE-CIC-IDS2018 datasets. A Docker-based simulation replicates constrained environments and captures network traffic in real time. The framework continuously monitors resources and dynamically selects the most suitable AI model for attack detection. Once an attack is detected, the system captures and preserves digitally signed critical forensic artefacts, categorised into system metadata, event/resource logs, network data, and processes. The AI-based framework aligns with ISO/IEC 27043:2015 Digital Forensic Readiness principles, automating many manual procedures and reducing both time and human effort. The quantitative evaluation demonstrates the effectiveness of the proposed network forensic readiness framework to address the specific challenges of RCEs.

Keywords: Network Forensic Readiness · IoT Forensics · Artificial Intelligence · Resource-constrained Environments

1 Introduction

The ubiquitous nature of Internet of Things (IoT) devices makes them vulnerable to attack. Investigating these attacks can be difficult because IoT devices have limited processing capabilities, power constraints, and storage, and different types of devices pose significant challenges for forensic investigation [2, 18].

IoT devices can be the source of potentially pertinent digital forensic evidence and are a common focus of digital forensic research [3]. However, IoT activities

are often short-lived and their associated data can be volatile. This presents challenges to trace security breaches, as the collection of evidence from IoT devices that are no longer connected to the network is complicated [7]. Data Integrity and Authenticity (DIA) are important in IoT forensics due to tampering risks and the transient nature of IoT data. Traditional digital forensic models developed for post-incident investigations struggle to adapt to the heterogeneous and complex nature of IoT devices. This limitation has complicated forensic investigations after security incidents, highlighting the need for a proactive approach.

Network Forensic Readiness (NFR) provides a proactive method to prepare for security incidents. NFR focusses on the identification, collection, and preservation of network evidence to support compliance and investigations [5]. The complexity of IoT networks necessitates a robust framework NFR approach to minimise human involvement in data collection, processing, and preservation while reducing investigative costs and latency. The framework proposed as part of this paper is aligned with the ISO/IEC 27043:2015 standard [23] to ensure that appropriate and effective Digital Forensic Readiness (DFR) methods are in place before an incident occurs.

The integration of Artificial Intelligence (AI) into network forensics has transformed attack detection and forensic readiness. The deployment of AI models in Resource-Constrained Environments (RCEs) poses significant challenges due to limited computational capabilities, energy constraints, and limited network bandwidth [21, 16]. A Network Intrusion Detection System (NIDS) serves as the first line of defence. An AI-based NIDS has significant demands on device resources and network bandwidth. It requires mechanisms to dynamically prioritise or delay detection based on resource availability. The transient nature of events in RCE, where evidence can vanish after network access, highlights the need for efficient and scalable frameworks.

1.1 Key Contributions

In this paper, an AI-based optimised NFR framework is proposed to improve forensic readiness in RCE. The framework facilitates real-time network traffic capture and processing at the source. The new insights from the work highlight the trade-off between resource utilisation, throughput, and detection accuracy in RCE. The framework introduces dynamic model selection in real time based on available computational resources to ensure optimal performance. It supports comprehensive evidence acquisition to help forensic investigation by collecting system metadata, logs, and memory snapshots at the time of the incident. The effectiveness of the framework is validated through extensive real-time attack simulations involving seven different types of attack across five distinct devices, providing a quantitative evaluation of its performance. The framework performs multiclass attack classification and detects attacks in just 0.3 ms per packet, making it suitable for real-time use. The suggested framework achieved the highest 99.58% and 99.91% accuracy in the CICIoT2023 and CSE-CIC-IDS2018 datasets, respectively.

2 Related Work

Several studies have explored various aspects to improve attack detection, evidence acquisition, preservation, and analysis while addressing the limited computational nature of RCE.

Sadineni et al. [19] presented Ready-IoT, a forensic-readiness model that gathers data at both the network and application layers. The workflow spans scenario definition, device setup, event detection, evidence acquisition, preservation and readiness configuration. Experiments in Contiki-NG's *Cooja* emulator assessed the model against jamming and synchronisation attacks in the Time-Slotted Channel Hopping (TSCH) layer. Evidence is retained in a secure database, maintaining the chain of custody.

Recently, Waguespack et al. [25] introduced the Memory Anomaly Recognition System (MARS), a host IDS that monitors device memory within a Trusted Execution Environment (TEE). A Convolutional Neural Network (CNN) deployed on a remote server analyses n-gram sequences from memory dumps to detect deviations from the memory baseline. These sequences are also converted into audio spectra to extract features such as MFCCs, Mel spectrograms, and chroma variants for intrusion detection. MARS uses a watchdog timer to enforce periodic memory capture, triggering a device reset upon anomaly detection. This mechanism mitigates basic malware, while a secure boot process protects firmware integrity against advanced threats. Evaluations of the STM32 controller demonstrated scalability, trustworthiness, accuracy, and robustness.

Kebande et al. [8] introduced a comprehensive forensic model for IoT devices based on the ISO/IEC 27043 standard. The model spans the entire forensic lifecycle through three phases: forensic readiness (proactive), forensic initialisation (incident), and forensic investigation (reactive). The proposed model eliminates the need for ad hoc approaches, supports diverse IoT applications, and provides a customisable and configurable framework. Rizal et al. [15] also introduced a framework by integrating the Gated Recurrent Unit (GRU) model to improve forensic readiness in an IoT environment based on the ISO/IEC 27043 standard. The framework includes a smart repository and a sophisticated storage system designed to organise evidence using metadata and contextual information. The repository automates tasks, recognises patterns, and provides intelligent insights.

Many researchers have explored and highlighted the capabilities of AI for digital forensics, network forensics, and NFR. However, RCEs face persistent challenges such as data diversity, proliferation, integrity, authorisation issues, resource constraints, and the volatile nature of data[17]. The black-box nature and high computational demands of AI models complicate their application for NFR within RCE. This study aims to address these challenges through the design and deployment of an optimised AI-based framework for NFR within RCE, aligned with the ISO/IEC 27043:2015 standard.

3 Methodology

This section provides a detailed overview of the design and architecture of the proposed framework. This study follows the Design Science Research (DSR) methodology [11]. The DSR focusses on the development of a framework that addresses real-world problems and contributes to scientific knowledge.

3.1 Framework Design

The proposed approach extends beyond algorithmic or conceptual models through a framework design and implementation for NFR, aligned with the ISO/IEC 27043:2015 standard. The architecture of the proposed AI-based optimised NFR framework is shown in Figure 1.

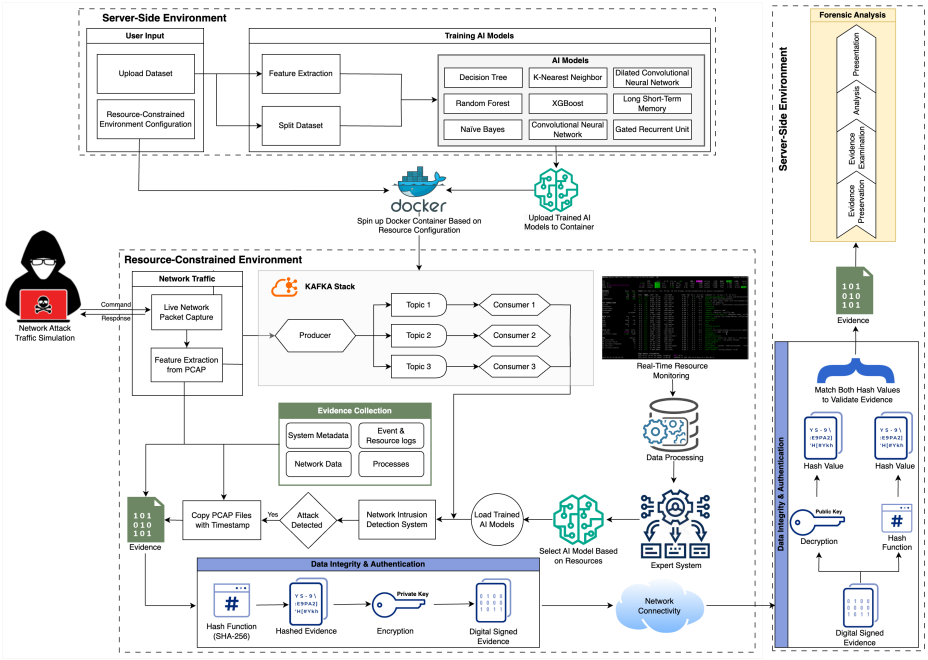

Fig. 1. The Proposed AI-based Optimised Network Forensic Readiness Framework Architecture for Resource-Constrained Environments

The NIDS component is the core feature of the proposed framework. This component utilises optimised AI models to enhance attack detection capabilities in RCE. AI models that incorporate both ML and DL approaches were trained using the CICIoT2023 [13] and CSE-CIC-IDS2018 [4] datasets. Both datasets were pre-processed using NumPy, Matplotlib, and Pandas by removing duplicates, handling missing/inconsistent values, and removing irrelevant features, which reduced the number of instances per attack class.

The use of dynamically interchangeable AI models instead of a single conventional model stems from the recognition that each model has its unique characteristics and corresponding performance-to-resource ratio under RCEs. These trained models are deployed within the RCE, where the framework continuously monitors resources such as power, memory, and CPU usage. An integrated expert system dynamically selects the most appropriate AI model based on available resources to ensure optimal trade-offs between real-time attack detection and resource utilisation. Upon detection of the attack, the framework activates evidence acquisition components to collect relevant data for forensic analysis. The DIA component secures the acquired evidence, which is then transmitted to a server-side environment for validation and secure preservation. Each component is elaborated in detail in the following subsections.

3.2 Network Intrusion Detection System

The framework's NIDS component used five ML and four DL algorithms to train models on the CICIoT2023 and CSE-CIC-IDS2018 datasets. Both ML and DL approaches were considered after reviewing relevant research that demonstrated their effectiveness for NIDS within RCE. The ML algorithms used for NIDS include Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Extreme Gradient Boosting (XGBoost), and K-Nearest Neighbours (KNN). For DL, the models included CNN and Recurrent Neural Networks (RNNs). To optimise these models for RCEs, extensive experiments were conducted by systematically tuning key hyperparameters such as tree depth, number of estimators, learning rate, maximum features, number of layers, filter sizes, kernel sizes, activation functions, dropout rates, and dilation rates to achieve the best performance.

Convolutional Neural Networks The CNN models, 1D Convolutional Neural Network (1D-CNN) and 1D-Dilated Convolutional Neural Network (1D-DCNN), were used to develop DL-based NIDS. These models account for the sequential nature of network traffic and extract spatial features from hierarchical structures such as TCP connections, flows, sessions, services, and hosts. 1D-DCNN captures a broader context that helps to reduce processing time without significantly increasing computational costs. This characteristic of 1D-DCNN makes it suitable for real-time NIDS in RCE.

The 1D-CNN architecture begins with convolutional layers using 32 filters to capture localised patterns that result in a tensor shape (None, 10, 32). A max-pooling layer was added to reduce the dimensions while keeping key features. This results in an output shape of (None, 5, 32) for better computational efficiency. Another convolutional layer with 64 filters was added to extract complex features that help to enhance feature representation and classification accuracy, reducing the sequence length to 3 and generating an output shape of (None, 3, 64). A flattening layer then converts the output into one-dimensional vectors. In the end, a dense layer was added along with a *softmax* activation function to perform classification.

The 1D-DCNN includes a convolutional layer with 32 filters, kernel size 1, and dilation rate of 2 to extract patterns and expand contextual scope, followed by another convolutional layer with 64 filters, kernel size 1, and dilation rate 8. These layers effectively handle feature extraction and representation. A flattening layer was added to reshape the output for integration with a dense layer, which uses *softmax* activation for classification. The *Adam* optimiser is used to minimise the loss function during neural network training.

Recurrent Neural Networks RNNs extend the feedforward networks with a hidden recurrent state. This includes temporal information through backpropagation over time. However, challenges such as vanishing gradients and high computational costs arise when processing long sequences. To address these, the Long Short Term Memory (LSTM) introduces a gating mechanism for longer memory and better control over weight updates. Another model is called GRU, a simpler variant of LSTM. GRU uses fewer parameters, making it faster and less computationally expensive. GRU utilises two gates: the update gate and the reset gate, which enable efficient retention of long-term dependencies in data. This design overcomes the limitations of traditional RNNs in handling sequential information. After an in-depth analysis of both mechanisms, the proposed optimised architectures used a reduced number of layers. This ensures computational efficiency and scalability for real-time applications.

The LSTM architecture is composed of an LSTM layer with the shape of (None, 64), where *None* indicates the variable batch size and *64* represents the number of output units. This layer has a total of 19,712 trainable parameters. Considering resource constraints, the architecture does not feature more hidden layers; instead, it is composed of a dense layer with an output shape of (None, 10) and 650 trainable parameters. The dense layer used the *softmax* activation function, which converts the learnt information into class probabilities.

The optimised GRU model followed an architecture similar to that of the proposed LSTM model, which is a single hidden layer and a dense layer. However, it used fewer trainable parameters compared to LSTM. The hidden GRU layer with a shape of (None,64) has 14,976 trainable parameters. This layer determines whether to retain or discard information from previous steps. The dense layer has an output shape (None, 10) with 650 parameters. This layer performs the attack classification based on the features learnt through the previous GRU layer.

Machine Learning Algorithms A systematic experimental methodology was implemented that involved training and evaluation of the most widely used algorithms, such as DT, RF, NB, KNN, and XGBoost, on both CICIoT2023 and CSE-CIC-IDS2018 datasets. Many ML models, including Logistic Regression, Linear Regression, K-means, and Support Vector Machine (SVM) were excluded from consideration during the experiments due to suboptimal performance or excessive detection time in preliminary testing on RCEs. Hyperparameter tuning and feature selection are important to achieve desirable results. Hyperparameter tuning for the selected ML algorithms was performed using the *GridSearchCV*

function. *GridSearchCV* performs a search through a predefined grid of hyperparameters using cross-validation. This process identifies optimal parameter combinations that maximise model performance while mitigating the risk of overfitting. The process involved defining relevant hyperparameters for each model, such as regularisation parameters, scaling values, solver algorithms, tree depth, and the number of neighbours. For example, in hyperparameter tuning of the RF model, key parameters that are considered to achieve the best results are the number of estimators, the maximum features per split and the splitting criteria that were systematically optimised. The models were evaluated on the datasets after hyperparameter tuning.

3.3 Evidence Acquisition

The evidence acquisition component of the framework is used to collect timestamped critical evidence upon attack detection. The collected evidence is organised into the following four categories: 1) system metadata containing the operating system and its version, architecture, processor cores, system uptime, timestamp, log-in history, and user information; 2) event and resource logs including memory dumps, system logs, event timelines, application logs, scheduled tasks, and resource usage; 3) network data comprising active network connections, network activity, network statistics, network connectivity, connected devices, device location, ARP tables, and captured network traffic (PCAP files); and 4) processes lists containing associated commands and user data, page tables, and kernel modules.

3.4 Data Integrity and Authentication

The DIA component of the proposed framework ensures compliance with legal requirements and forensic principles for admissible evidence, according to the ISO/IEC 27043:2015 standard. After the acquisition of evidence, a hash value is generated from the collected evidence using SHA-256 [12]. The hash value is then encrypted using a Rivest-Shamir-Adleman (RSA) private key [14]. On the server side, the DIA component verifies the digitally signed evidence using the corresponding public key, confirming that the evidence has not been tampered with and originates from a trusted source.

3.5 Forensic Analysis

The forensic analysis component of the framework is responsible for preserving evidence on a server after implementing DIA To examine the preserved evidence, Volatility and Wireshark were used to perform memory forensics and network traffic analysis.

4 Implementation and Results

The experiments comprehensively evaluate the framework on different RCEs and network attack scenarios. The evaluation involves a conscientious alignment of the proposed framework with the ISO/IEC 27043:2015 standard. As depicted in Figure 1, the proposed framework uses two environments, the server-side environment and RCE.

Server-Side Environment The server environment uses macOS 14.0 with a 3.49 GHz ARM-based processor and 16GB LPDDR5 RAM. AI models were developed in Python 3.11.5 using Keras, TensorFlow, and scikit-learn. All models were trained, optimised, and tested on the server prior to deployment in RCEs. The server also acted as a secure repository for evidence preservation using digital signing to ensure data integrity and authenticity (DIA), and supported forensic examination and analysis.

Resource Constrained Environments Multiple RCEs were emulated via Docker containers [10], replicating five devices: Raspberry Pi Zero 2 W, Raspberry Pi 3B+, ODROID-C2, Orange Pi 4, and NVIDIA Jetson Nano. All devices used Linux-based operating systems, as detailed in Table 1.

Table 1. Hardware specifications of devices used for simulation

Device Name	Memory (MB)	Storage (GB)	CPU (GHz)	Power Usage (W)	Processor
Raspberry Pi Zero 2 W	512	8	1.00	1.50	ARM Cortex-A53
Raspberry Pi 3B+	1024	16	1.40	5.00	ARM Cortex-A53
ODROID-C2	2048	16	1.50	3.00	ARM Cortex-A53
Orange Pi 4	3072	16	1.80	7.00	Cortex-A72 + Cortex-A53
NVIDIA Jetson Nano	4096	16	1.43	10.00	Cortex-A57 MPCore

To evaluate the robustness of the proposed framework, various attack traffic scenarios were simulated and transmitted to RCEs. The deployment of the proposed framework was implemented using Python 3.9.2, supported by various libraries such as Keras, TensorFlow, Scikit-learn, Pandas, Pyshark, Scapy, LiME, and cryptography. The Kafka stack was integrated to facilitate real-time streaming data pipelines and enable the deployment of a real-time NIDS.

4.1 Simulated Network Attack Traffic

Real-time network attack traffic generated using *Scapy*. Scapy was used to construct, decrypt, send, capture, and analyse packets using various protocols. The experiment replicated seven attack types from the CICIoT2023 dataset: DDoS, Brute Force, Spoofing, DoS, Reconnaissance, Web-based, and Mirai. Each attack

category included multiple scenarios. For example, web-based attacks included SQL injection, command injection, and Cross-Site Scripting (XSS).

In addition, a composite attack scenario randomly simulated seven attack types, automating the process to reveal the most effective method mix for broader and more potent threats. This setup assessed AI models and RCE behaviour when multiple attackers launched diverse assaults from different sources.

4.2 Network Intrusion Detection System

The first line of defence of the framework is the NIDS, where AI models were initially trained and evaluated within a server-side environment to develop optimised models. DL models were evaluated by monitoring accuracy and loss in both training and validation sets at each epoch. This facilitates effective anomaly detection within RCE in real-time data.

The model is compiled with an *Adam* optimiser and sparse categorical *cross-entropy* loss function. Adam optimiser was selected due to its adaptive estimation capabilities and low memory requirements, whereas cross entropy accurately evaluates the divergence between various attack types. Early stopping, a regularisation technique, was used to improve the ability of the DL model to generalise by avoiding excessive adaptation to the training data. This technique is integrated to determine the optimal point at which to stop the training process. The early stopping approach monitors the model's performance using a holdout validation set and a metric, such as loss, to ensure training stops when there is no further improvement.

In general, the RF model demonstrated the best performance among the selected models on the CICIoT2023 dataset. It achieved 99.58% accuracy, with precision and recall scores of 0.996 and an F1 score of 0.996. RF required 222.205 s to train with a prediction time of 2.243 s. However, considering resource utilisation and bandwidth constraints within RCE, the DT model outperformed RF. DT achieved an accuracy of 99.47% along with precision, recall, and F1 scores of 0.995. The DT model required significantly less time compared to RF. DT used 13.154 s to train the model and just 0.032 s for prediction, as shown in Table 2.

Among the DL models on the CICIoT2023 dataset, GRU outperformed the other DL models. GRU achieved 99.03% accuracy, with precision, recall, and F1 scores of 0.990. GRU required 202.856 s for training and 8.157 s for prediction across different attack types. LSTM model followed closely and achieved a slightly better accuracy of 99.05% with precision, recall, and F1 scores of 0.990, 0.991, and 0.990, respectively. However, LSTM demanded 226.543 s for training and a slightly longer prediction time of 9.001 s compared to GRU. The prediction time is crucial to deploy AI models for real-time NIDS within RCE.

On the CSE-CIC-IDS2018 dataset, DL models outperformed ML models across all evaluation metrics, including accuracy, precision, recall, and F1 score. Among the DL models, the 1D-DCNN demonstrated the best performance, achieving an accuracy of 99.99%, a precision of 1.000, recall of 0.991, and an F1 score of 0.996. ML models generally require less time for both training and prediction. KNN had the shortest training time at 0.073 s but exhibited the

Table 2. Performance Comparison of Optimised AI Models on CICIoT2023 and CSE-CIC-IDS2018 Datasets

	CICIoT2023						CSE-CIC-IDS2018					
Model	**Acc. (%)**	**Prec.**	**Recall**	**F1**	**Train (s)**	**Pred (s)**	**Acc. (%)**	**Prec.**	**Recall**	**F1**	**Train (s)**	**Pred (s)**
DT	99.47	0.995	0.995	0.995	13.154	0.032	96.89	0.969	0.968	0.969	5.506	0.008
RF	99.58	0.996	0.996	0.996	222.205	2.243	97.11	0.969	0.971	0.970	43.776	0.417
KNN	99.19	0.992	0.992	0.992	0.339	1250.771	95.42	0.944	0.954	0.945	0.073	45.346
NB	78.40	0.685	0.784	0.712	0.821	0.466	16.20	0.851	0.162	0.216	0.147	0.062
XGB	99.49	0.995	0.995	0.995	138.793	0.991	98.03	0.980	0.980	0.978	23.133	1.045
LSTM	99.05	0.990	0.991	0.990	226.543	9.001	99.88	1.000	0.998	0.999	26.559	0.909
GRU	99.03	0.990	0.990	0.990	202.856	8.157	99.88	1.000	0.998	0.999	26.559	0.858
1D-CNN	98.91	0.989	0.989	0.988	978.648	17.634	99.87	0.998	0.998	0.998	213.575	2.896
1D-DCNN	98.98	0.990	0.990	0.989	461.419	15.248	99.99	1.000	0.991	0.996	80.651	1.694

highest prediction time 45.346 s, due to its instance-based architecture that relies on distance calculations during inference. NB followed, with a training time of 0.147 s and prediction time of 0.062 s; however, it achieved the lowest accuracy among all models, with only 16.20%. Among the DL models, GRU stood out for its efficiency, requiring just 26.559 s for training and 0.858 s for prediction while still delivering strong performance, as shown in Table 2.

4.3 Framework Deployment

The proposed framework was deployed on various simulated devices to determine its performance within RCE.

The proposed framework was deployed on various simulated devices to evaluate its performance within RCE. Network attack traffic was generated using seven distinct attack types, with each attack simulated on different devices. For consistency, each attack type included 500 network packets of uniform packet size to allow a controlled evaluation of the performance of the framework in terms of attack detection, evidence acquisition, and resource utilisation. Extensive experiments were conducted to gather the essential knowledge for the development of the expert system, which is integrated as a component of the proposed framework. This knowledge enables the expert system to identify the most appropriate AI model based on the available resources within RCE. The results of the simulated reconnaissance attack on 5 different devices are shown in Table 3. Similar results were obtained when simulating other attacks on various devices.

The primary objective of these experiments is to dynamically select the most appropriate AI model based on the available resources. If sufficient resources were not available, the framework refrained from engaging AI-based NIDS and allowed the RCE to prioritise its primary task.

In general, the DT model demonstrated better performance in terms of resource utilisation, detection time, and evidence collection. DT required only 16.5% of CPU usage and 276 MB of memory to process 500 network packets associated with suspicious activity. In addition, the DT model presented exceptional real-time capabilities by detecting a single attack packet in 0.3 ms and

Table 3. AI Models Resource Utilisation on Raspberry Pi-3B+ Device for Reconnaissance Attack on 500 Network Packets

	Device			Model			Time		
Model	**CPU Usage (%)**	**Memory Usage (MB)**	**Power Usage (mW)**	**CPU Usage (%)**	**Memory Usage (MB)**	**Power Usage (mW)**	**Detection Time per Packet (s)**	**Total Detection Time (s)**	**Memory Dump Time (s)**
DT	18.8	597	114.0	16.5	276	57.1	0.0003	0.1659	2.91
RF	28.3	638	95.0	24.4	398	55.1	0.0014	0.9287	3.46
KNN	87.2	857	267.0	84.5	694	138.0	0.0674	38.1184	4.07
NB	22.1	597	89.7	16.9	269	47.1	0.0003	0.1833	3.59
XGB	20.3	590	104.0	17.1	279	54.2	0.0004	0.2401	3.10
LSTM	53.3	588	103.0	47.7	315	65.4	0.0371	20.6388	3.00
GRU	52.4	582	106.0	47.6	320	65.2	0.0455	20.3733	2.98
1D-CNN	51.7	576	95.8	47.0	319	57.0	0.0393	20.3423	5.32
1D-DCNN	53.6	598	80.2	46.9	314	50.9	0.0404	20.2980	2.98

0.1659 s for 500 network packets with high accuracy. DT also consumed minimal power during attack detection, which makes it highly efficient for RCEs. The RF model followed DT in performance but required slightly more computational resources. RF consumed 24.4% of CPU and 398 MB of memory while achieving a detection time of 0.0014 s per packet and 0.9287 s for 500 network packets. Although RF required more computational resources, its performance remained competitive, making it suitable for scenarios where additional resources are available.

In the case of DL models, resource utilisation and detection times were relatively similar across all models. However, the 1D-DCNN model outperformed other DL models in terms of efficiency. It utilised 46.9% CPU and 314 MB of memory, consuming 50.9 mW of power to detect a network attack packet. On average, 1D-DCNN required 0.0404 s per packet and processed 500 packets in 20.2980 s.

These findings highlight the trade-offs between computational efficiency and detection capabilities in different models, providing valuable insights into selecting the most appropriate AI model based on the available resources in RCE.

Extensive Testing To evaluate the scalability of the proposed framework, a series of experiments were conducted that analysed its performance based on the varying volume of network packets. The experiments were carried out on the five

aforementioned devices, with the number of packets gradually increasing from 500 to 5000. DT and GRU models were selected to evaluate performance, and the results of the experiments are presented in Table 4 and Table 5, respectively.

Table 4. DT Performance with Various Number of Network Packets on Raspberry Pi-3B+ Real-Time Network Traffic

No of Packets	Features Extraction Time (s)	Packet Size (Bytes)	Detection Time (s)	Total Time With AI (s)	Total Time Without AI (s)	Bandwidth With AI (Mbps)	Bandwidth Without AI (Mbps)	Power Consumption (mW)	Memory Dump Time (s)	Max Memory Utilisation (MB)
500	1.31	417,853	0.20	2.18	0.88	1.45	10.45	107	2.98	687
1000	2.32	864,580	0.33	4.12	1.80	1.60	3.67	205	3.39	717
2000	5.12	1,685,576	0.64	8.68	3.56	1.48	3.61	252	3.84	727
3000	7.74	2,524,791	0.97	12.89	5.15	1.49	3.74	333	4.15	733
4000	10.72	3,357,845	1.31	17.52	6.81	1.46	3.76	350	5.76	740
5000	14.20	4,234,019	2.00	22.97	8.77	1.41	3.68	326	3.67	756

The experiments considered critical factors such as network traffic, time, computational resources, and bandwidth utilisation, both with and without AI integration. The results demonstrated that as the number of packets increased, the processing time per packet remained relatively similar. The comparative results highlight the ability of the proposed framework to handle larger network traffic efficiently.

The dynamic selection of AI models based on available resources was evaluated through multiple tests. A simulated application was developed using Python that utilised interactive sliders that enable dynamic adjustment of resource utilisation. The testing also validated that if RCE requires exceeded computational resources for its primary task, the framework would bypass attack detection and evidence collection to ensure uninterrupted operation. After the simulated application, different resource-intensive applications were executed within RCE to observe their impact on model selection in real time.

Table 5. GRU Performance with Various Number of Network Packets on Raspberry Pi-3B+ Real-Time Network Traffic

No of Packets	Features Extraction Time (s)	Packet Size (Bytes)	Detection Time (s)	Total Time With AI (s)	Total Time Without AI (s)	Bandwidth With AI (Mbps)	Bandwidth Without AI (Mbps)	Power Consumption (mW)	Memory Dump Time (s)	Max Memory Utilisation (MB)
500	1.34	417,853	20.65	2.22	0.88	1.44	3.64	128	4.01	644
1000	2.25	864,580	42.42	4.05	1.80	1.63	3.67	168	3.11	674
2000	5.10	1,685,576	81.24	8.67	3.56	1.48	3.61	283	3.85	709
3000	9.46	2,524,791	129.16	14.62	5.15	1.32	3.74	340	3.19	708
4000	12.23	3,357,845	175.54	19.04	6.81	1.35	3.76	335	3.08	784
5000	16.79	4,234,019	213.64	25.57	8.77	1.26	3.68	345	3.61	717

Real-Time Feature Extraction The feature extraction module in the proposed framework was deployed for real-time feature extraction after network traffic capture. This module is capable of processing network packets from IEEE 802.11 (WiFi), Ethernet, and Zigbee communication standards. These communication standards were selected to ensure real-time compatibility with the IoT network, as the majority of IoT devices use these protocols for data transmission.

The module extracts features from three key protocols: TCP/IP (Transmission Control Protocol/Internet Protocol), UDP (User Datagram Protocol), and ARP, which identify the essential attributes of the network packet data for potential attack detection. After processing PCAP-stored packets, it derives the same 40 features used to train the ML and DL models, structuring them into input vectors according to the expected model input shape. These are streamed in real time via Kafka, where producers transmit them to consumers for AI-based attack detection, which returns a numeric class label representing the predicted attack type. This design ensures minimal impact on network bandwidth during prediction. Upon detection, the Kafka consumer activates the evidence acquisition component.

Evidence Acquisition Evidence acquisition is the crucial component of the framework to ensure a comprehensive and systematic collection of evidence within the RCE. The process begins with the collection of system metadata

that includes evidence files containing user information (active users and login history with timestamps) and system details such as timestamps, operating system specifications, machine architecture, physical cores, and memory statistics (total and available memory). After collecting system metadata, the component collects detailed event and resource logs. These logs include a system log file that records all events, requests, and activities on the system, an event timeline file to establish when specific events occurred, an application log file that provides behavioural information about running software, and a scheduled tasks file that documents tasks set to execute at defined times. Memory-related evidence is also collected, including memory dump files (snapshots of physical memory at a given moment) and a page table file that includes buffered and cached memory, swap memory, active and inactive memory, paging information, kernel memory usage, and system parameters such as *CommitLimit*, *Committed_ AS*, *VmallocTotal*, and *VmallocUsed*. The memory detail provides valuable insights into the system's run-time state and resource use at the time of evidence collection.

The component also captures network-related data to provide network activity on RCE. This includes PCAP files that sequentially record network packets, a connected device file that contains information about connected devices and details about the kernel IP routing table, an active connection file consisting of IP addresses, port numbers, and the statuses of active connections, and a file of network connectivity that includes IP addresses, subnet masks, and MAC addresses for network interfaces. The geolocation data file provides the longitude, latitude, and address of the devices. In the end, the framework gathers process-related information, including a process list file that provides details of active processes along with their usernames, process IDs (PIDs), names, and command-line arguments, and a kernel module information file that outlines installed modules, their sizes, and dependencies. This comprehensive collection ensures that no critical aspect of the state or activity of the system is overlooked. Once the evidence collection process is complete, the framework triggers the DIA component.

Data Integrity and Authentication In the RCE, the DIA component generated unique private and public keys after attack detection and evidence collection. SHA-256 was used to create a unique hash for each evidence file mentioned in Section 4.3. The private key was used to encrypt the hash to generate digitally signed evidence.

In the server-side environment, the signed evidence was decrypted using the corresponding public key to retrieve the original hash value. Simultaneously, the SHA-256 hash was compared to the received evidence hash value to verify that the evidence had not been tampered with. This process validated the integrity of the evidence to ensure that it was in accordance with the standards and admissibility of the evidence in legal proceedings. After the evidence was validated, it was stored on a server for forensic analysis.

5 Discussion

The adaptability of RCEs is rapidly increasing due to cost-effective hardware and widespread connectivity. However, their inherent vulnerabilities, such as outdated security, weak authentication, minimal security considerations during development, insecure network services, etc., make them more vulnerable to infiltration than traditional computing systems. The evolving threat landscape continues to challenge effective digital forensic investigations in RCEs. Despite existing forensic tools, the complex architecture of RCE demands innovative solutions to bridge the gap between ideal forensic procedures and real-world constraints. AI models have been adopted for the detection and analysis of cybersecurity incidents. However, AI-based forensic readiness faces challenges in RCE due to resource constraints, power consumption, and the effects on network bandwidth. The proposed AI-based optimised NFR framework improves forensic readiness by automating incident detection and response. This reduces response time, minimises human intervention, and improves attack detection, evidence acquisition, preservation, and analysis. Quantitative simulation helps the framework to dynamically select the most appropriate AI model based on the available resources, including CPU, memory, and power. The framework is designed to capture and process network packets to extract features for AI models from different communication standards and protocols in real time. Its versatility and scalability make it compatible with most RCEs. The framework prioritises strategic planning, as discussed in ISO/IEC 27043:2015. It integrates attack detection, evidence acquisition, validity checks, and secure evidence storage to ensure comprehensive tamper-proof digital evidence for legal proceedings.

Real-Time Deployment Feasibility in Resource-Constrained Environment The effectiveness of real-time NFR depends not only on detection accuracy, but also on its ability to identify attacks promptly, since evidence collection begins immediately after detection. Lightweight AI models that can run on RCEs and detect attacks promptly are essential; any delay can lead to critical packet loss and missed forensic evidence, ultimately compromising the proactive approach. Shoukat et al. [22] reported detection times of 0.079 ms, 0.087 ms, 0.084 ms on N-BaIoT, Edge-IIoTset, and CIC-IDS2017 using a hardware configuration of an Intel Core i5-6200U CPU @ 2.30 GHz, 12 GB RAM. However, this study did not evaluate performance on constrained devices such as the Raspberry Pi Zero 2 W. Wang et al. [27] reported a detection time of 0.0172 ms and achieved an accuracy of 99.90% on the CSE-CIC-IDS2018 dataset. However, the authors did not evaluate their model on low-end devices.

In comparison, the proposed framework achieved detection times of 0.3 ms per packet on real-time traffic and was successfully deployed on a variety of resource constrained devices, as listed in Table 1. The offline evaluation further showed 0.00019 ms and 0.001 ms detection times on CICIoT2023 and CSE-CIC-IDS2018 datasets. The comparison of the proposed framework with the existing state-of-the-art is shown in Table 6.

Table 6. Proposed Model Performance Comparison with Existing Studies

Ref.	Low-End Device	Dataset	Acc. (%)	Train (s)	Detect (ms)	Pwr Usage (mW)
Shoukat et al. [22]	✗	CIC-IDS2017	98.57	–	0.084	–
Vellela et al. [24]	✗	CSE-CIC-IDS2018	99.30	882.00	–	–
Zhang et al. [27]	✗	CSE-CIC-IDS2018	99.90	74.88	0.0172	–
Seth et al. [20]	✗	CSE-CIC-IDS2018	97.72	5.06	0.13801	–
ElSayed et al. [6]	✓	CICIoT2023	93.60	–	–	7500
Kharoubi et al. [9]	✗	CICIoT2023	99.17	–	0.060	–
Alzahrani et al. [1]	✓	CICIoT2023	99.10	–	6760.00	6040
Wang et al. [26]	✗	CICIoT2023	93.13	708.40	6.40	–
Proposed Model	✓	**CICIoT2023**	99.58	222.20	0.00019	114
Proposed Model	✓	**CSE-CIC-IDS2018**	99.99	80.65	0.00100	635

5.1 Limitations and Future Work

An AI-based optimised NFR framework was introduced, demonstrating robustness, accuracy, integrity, and authentication. However, certain design and testing limitations should be acknowledged. The framework was not deployed on real devices, so future research should involve implementing and evaluating it on representative devices to validate its performance in realistic settings. The privacy of data collected during forensic investigations is beyond the scope of this study. The time required for dynamic model selection is not addressed and will be explored in future work. Additionally, the proposed approach relied on TensorFlow and Kafka, which require more computational resources than lightweight options, e.g., MQTT and TensorFlow Lite. Future implementations should consider AI frameworks customised for RCEs.

6 Conclusion

This paper introduces an AI-based optimised NFR framework for RCE. The framework integrates five ML and four DL models for attack detection, followed by evidence collection that is categorised into system metadata, event and resource logs, network data, and process information. The proposed framework achieved an optimal balance between the use of computational resources, the bandwidth of the network, and the accurate detection of attacks. The expert system was integrated, which utilised the knowledge of extensive experiments carried out in different simulated environments to find the appropriate AI model for attack detection based on available computational resources on RCE. The DIA component of the framework effectively validated the evidence using a robust digital signing technique. This ensures that the evidence remains unmanipulated and admissible for legal proceedings. The framework was designed and developed to enhance the NFR standard outlined in ISO/IEC 27043:2015 through the application of AI.

Bibliography

[1] Alzahrani, H., Sheltami, T., Barnawi, A., Imam, M., YASAR, A.: A Lightweight Intrusion Detection System using Convolutional Neural Network and Long Short-Term Memory in Fog Computing (2024)

[2] Atlam, H.F., Alenezi, A., Alassafi, M.O., Alshdadi, A.A., Wills, G.B.: Security, Cybercrime and Digital Forensics for IoT, pp. 551–577. Springer International Publishing, Cham (2020)

[3] Breitinger, F., Hilgert, J.N., Hargreaves, C., Sheppard, J., Overdorf, R., Scanlon, M.: DFRWS EU 10-Year Review and Future Directions in Digital Forensic Research. Forensic Science International: Digital Investigation **48**, 301685 (03 2024). https://doi.org/https://doi.org/10.1016/j.fsidi.2023.301685

[4] Canadian Institute for Cybersecurity: A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018), `https://registry.opendata.aws/cse-cic-ids2018/`, last accessed 02 June 2022.

[5] Darabseh, A., Kbar, G., Almulhem, A.: Network Forensics Readiness: A Survey. Journal of Digital Forensics, Security and Law **11**(2), 61–76 (2016)

[6] ElSayed, Z., Elsayed, N., Bay, S.: A Novel Zero-Trust Machine Learning Green Architecture for Healthcare IoT Cybersecurity: Review, Analysis, and Implementation. In: SoutheastCon 2024. pp. 686–692 (2024)

[7] Fagbola, F.I., Venter, H.S.: Smart Digital Forensic Readiness Model for Shadow IoT Devices. Applied Sciences **12**(2) (2022). https://doi.org/10.3390/app12020730

[8] Kebande, V.R., Mudau, P.P., Ikuesan, R.A., Venter, H., Choo, K.K.R.: Holistic Digital Forensic Readiness Framework for IoT-enabled Organizations. Forensic Science International: Reports **2**, 100117 (2020)

[9] Kharoubi, K., Cherbal, S., Mechta, D., Gawanmeh, A.: Network Intrusion Detection System Using Convolutional Neural Networks: NIDS-DL-CNN for IoT Security. Cluster Computing **28**(4), 219 (2025)

[10] Merkel, D.: Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal (239), 2 (2014)

[11] Montasari, R., Carpenter, V., Hill, R.: A road map for digital forensics research: a novel approach for establishing the design science research process in digital forensics. International Journal of Electronic Security and Digital Forensics **11**(2), 194–224 (2019)

[12] National Institute of Standards and Technology: Secure Hash Standard. Federal Information Processing Standards Publication FIPS PUB 180-4, U.S. Department of Commerce (2015)

[13] Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A.: CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. Sensors **23**(13) (2023)

[14] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (Feb 1978). https://doi.org/10.1145/359340.359342
[15] Rizal, R., Selamat, S.R., Mas'ud, M.Z., Widiyasono, N.: Enhanced Readiness Forensic Framework for the Complexity of Internet of Things (IoT) Investigation Based on Artificial Intelligence. Journal of Advanced Research in Applied Sciences and Engineering Technology **50**(1), 121–135 (2025)
[16] Rizvi, S., Scanlon, M., Mcgibney, J., Sheppard, J.: Application of Artificial Intelligence to Network Forensics: Survey, Challenges and Future Directions. IEEE Access **10**, 110362–110384 (2022)
[17] Rizvi, S., Scanlon, M., McGibney, J., Sheppard, J.: Deep learning based network intrusion detection system for resource-constrained environments. In: Digital Forensics and Cyber Crime. pp. 355–367. Springer Nature Switzerland, Cham (2023)
[18] Rizvi, S., Scanlon, M., McGibney, J., Sheppard, J.: Pushing network forensic readiness to the edge: A resource constrained artificial intelligence based methodology. In: 2024 Cyber Research Conference - Ireland (Cyber-RCI). pp. 1–8 (2024). https://doi.org/10.1109/Cyber-RCI60769.2024.10939120
[19] Sadineni, L., Pilli, E.S., Battula, R.B.: Ready-IoT: A Novel Forensic Readiness Model for Internet of Things. In: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT). pp. 89–94 (2021)
[20] Seth, S., Singh, G., Kaur Chahal, K.: A Novel Time Efficient Learning-based Approach for Smart Intrusion Detection System. Journal of Big Data **8**(1), 111 (2021)
[21] Shahin, M., Maghanaki, M., Hosseinzadeh, A., Chen, F.F.: Advancing Network Security in Industrial IoT: A Deep Dive into AI-Enabled Intrusion Detection Systems. Advanced Engineering Informatics **62**, 102685 (2024)
[22] Shoukat, S., Gao, T., Javeed, D., Saeed, M.S., Adil, M.: Trust my IDS: An Explainable AI Integrated Deep Learning-based Transparent Threat Detection System for Industrial Networks. Computers & Security **149**, 104191 (2025)
[23] Valjarević, A., Venter, H., Petrović, R.: ISO/IEC 27043: 2015—Role and Application. In: 2016 24th Telecommunications Forum (TELFOR). pp. 1–4. IEEE (2016)
[24] Vellela, S.S., D, R., Purimetla, N.R., Thalakola, S., Vuyyuru, L.R., Vatambeti, R.: Cyber Threat Detection in Industry 4.0: Leveraging GloVe and Self-ttention Mechanisms in BiLSTM for Enhanced Intrusion Detection. Computers and Electrical Engineering **124**, 110368 (2025)
[25] Waguespack, K.M., Smith, K.J., Muliri, O.A., Vijayakanthan, R., Ali-Gombe, A.: MARS: The First Line of Defense for IoT Incident Response. Forensic Science International: Digital Investigation **49**, 301754 (2024)
[26] Wang, Z., Chen, H., Yang, S., Luo, X., Li, D., Wang, J.: A Lightweight Intrusion Detection Method for IoT Based on Deep Learning and Dynamic Quantization. PeerJ Computer Science **9**, e1569 (2023)
[27] Zhang, H., Zhang, B., Huang, L., Zhang, Z., Huang, H.: An Efficient Two-Stage Network Intrusion Detection System in the Internet of Things. Information **14**(2) (2023). https://doi.org/10.3390/info14020077