Digital Innovations & Contemporary Research in SCIENCE, ENGINEERING & TECHNOLOGY
Vol. 14. No. 1, 2026 Series

Article Citation Format
Ekpo, M.E., Ituma, C. & Kekong, P.E. (2026): Assessment Of A Novel Deep Learning Framework Using Convolutional Neural Networks (CNN) And Ensemble Of Artificial Neural Networks (ANN) For Vulnerability Classification. Journal of Digital Innovations & Contemporary Research in Science, Engineering & Technology. Vol. 14, No. 1. Pp 37-56. www.isteams.net/digitaljournal dx.doi.org/10.22624/AIMS/DIGITAL/V14N1P3

Article Progress Time Stamps
Article Type: Research Article
Manuscript Received: 27th December, 2025
Review Type: Blind Peer
Final Acceptance: 2nd February, 2026

Assessment Of A Novel Deep Learning Framework Using Convolutional Neural Networks (CNN) And Ensemble Of Artificial Neural Networks (ANN) For Vulnerability Classification

[1]Ekpo, M.E., Ituma, C. (PhD) & Kekong, P.E.
[1]Department of Computer Science, Ebonyi State University, Abakaliki, Nigeria
[2]Department of Computer Science, Ebonyi State University, Abakaliki, Nigeria
[3]Dept of Mathematics & Computer Science, Federal University of Health Sciences, Otukpo, Nigeria
E-mails: [1]ekpom550@gmail.com; [2]ichinagolum@gmail.com; [3]piuskekong2019@gmail.com
Author's Phone NO: 08039590555

ABSTRACT

Introduction of Internet of Things (IoT) devices in healthcare is a major security concern with critical infrastructure exposed to cyber threats due to vulnerabilities at all levels of physical, network, and application. Conventional vulnerability management systems tend to be responsive in character, have high false-positive, and are not holistic in the multi-layer flaw detection approach. The paper will introduce Convolutional VulNet, which is a new deep learning architecture that combines the Convolutional Neural Networks (CNN) to extract features and an ensemble of Artificial Neural Networks (ANN) to classify the vulnerabilities robustly. The model has been incorporated into a Deep Packet Inspection (DPI) pipeline in order to analyze real-time network traffic and is backed by a rule-based scoring system to rank risks. The proposed system is trained and tested on a large dataset of 901,869 records of an entire hospital IoT network and the Common Vulnerabilities and Exposures (CVE) database, and achieves a state-of-the-art accuracy of 99.42%, which is significantly higher than previous models. The findings reveal that Convolutional VulNet is a better, stable, and viable solution to proactive vulnerability management in healthcare IoT that can ensure that sensitive patient information remains intact and that the necessary medical services are always accessible.
Keywords: Deep Packet Inspection; Vulnerability Management; Convolutional Neural Network; Ensemble Learning; Artificial Intelligence

Keywords: Deep Packet Inspection; Vulnerability Management; Convolutional Neural Network; Ensemble Learning; Artificial Intelligence

1. INTRODUCTION

Cyberattacks are malicious acts that seek to compromise network environments, using threat features such as viruses, malware, trojans, and botnets (Tasci, 2024). Recent studies have shown that 70% of CNIS have an average of 25 flaws (Baho and Abawajy, 2023), and a primary reason for over 300% increase in cyber-attacks (Bertino and Islam, 2023; Arampatzis, 2022).

To carry out a successful cyber-attack, Hassani et al. (2021) revealed that there must be vulnerability in the targeted facility. Vulnerability management is a risk-based approach applied to detect, prioritize, remediate, and mitigate flaws (Neshenko et al., 2019). The traditional method of vulnerability management involved the application of vulnerability scanning tools, a vulnerability scoring system, a patch management system, and an intrusion detection system (Aloui and Nfaoui, 2022); however, these approaches are more of a reactive solution and hence struggle to guarantee elements of computer network security, which are confidentiality, integrity, and availability. In addition, Perkel (2022) revealed that traditional vulnerability scanning tools are only able to detect 82% of vulnerabilities, thus having false positive results of 18%.

To solve this problem, Deep Packet Inspection (DPI) has emerged as a pivotal approach in network security, allowing thorough network analysis (Mike et al., 2022; Wenguang et al., 2020). The DPI has been applied to solve several cybersecurity problems such as threat detection and vulnerability management (Maria et al., 2020; Foreman et al., 2024; Son et al., 2023). For instance, Neshenko et al. (2019) address vulnerability in IoT facilities through packet prioritization, while Feng et al. (2022) focused on the detection of firmware vulnerabilities. In contrast, state-of-the-art ML and deep learning algorithms (DL) are applied in Yu et al. (2020) for vulnerability management in IoT, while Ahanger et al. (2022) improved IoT security through a survey of artificial intelligence (AI) systems for network vulnerability management.

Naeem et al. (2020) applied DL for vulnerability management in IoT and then tested the model on two datasets, which both reported over 61% accuracy. Meidan et al. (2020) focused on detecting vulnerabilities for telecommunication service providers using the long short-term memory algorithm, while convolutional neural networks were applied in Ullah et al. (2019) for vulnerability detection on IoT networks. While these studies have recorded significant success in detection and management of vulnerability for IoT network facilities, there is need to focus this energy for critical IoT facilities in the health care sector (Usak et al. 2020; Bhatt and Chkraborty, 2023).

The health sectors are an important area in the global economy that requires optimal security to protect patient health records. There is a need for a vulnerability management model that ensures that every flaw that can be exploited by an attacker is detected and reported for rapid incident response. In addition, Hulayyil et al. (2023) and Madanian et al. (2024) revealed that IoT facilities have different layers (physical, application, and perceptron), each with their unique vulnerability attributes, but recent vulnerability management models are never considered flows across these layers holistically, hence raising an issue of reliability in existing models. Finally, Madusha et al. (2022) submitted new vulnerabilities not capture in current vulnerability database like the Common Vulnerability Exploit (CVE), raises another concern in existing system.

In addition, the integration of cloud computing into health IoT also introduces additional risks from misconfiguration, while insider threats and the use of IoT devices in botnets exacerbate the situation (Madanian et al., 2024), thus raising a research question on what can be done to improve vulnerability detection performance in healthcare IoT facilities? To solve the research question and other issues raised in the background, this study proposes the design of a deep packet inspection model for vulnerability management in critical network infrastructure using artificial intelligence techniques

2. THE PROPOSED SYSTEM

The proposed system will be developed using methods such as the case study and experimental tested, the data collection, data processing, convolutional VulNet, deep packet inspection, system integration of the vulnerability management model. Figure 1 presents the block diagram of the proposed vulnerability management model for healthcare-IoT protection.

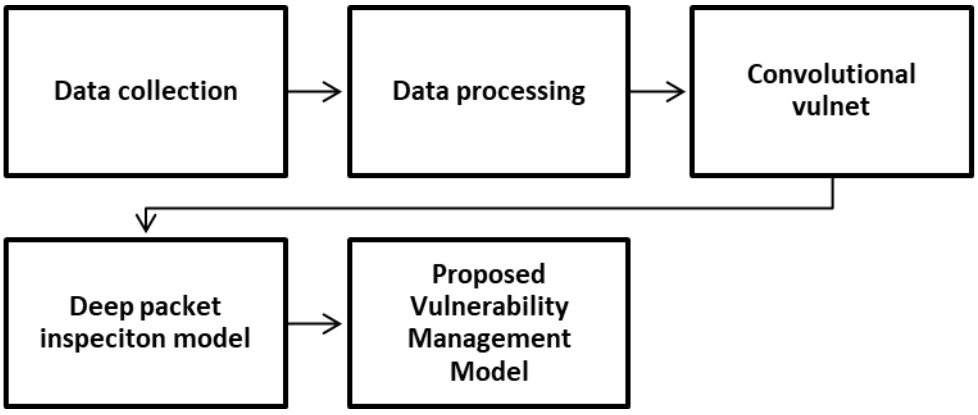

Figure 1: Block diagram of the research methods

2.1 Data Collection

Data used for this work was collected from primary and secondary sources. The primary source of data collection is the University of Uyo teaching hospital, information technology and communication department, Akwa Ibom state. The population size is 2010 to 2024 vulnerability records. This case study was selected for the study due to its reliance on IoT enables medical devices for the management of patient health information, making it a prime target for cyber threat. The vulnerability data collected span across all the IoT layers. For the perceptron/physical layer, vulnerabilities such as weak authentication, unencrypted communication data and outdated firmware were collected using Wireshark as the tool. For the network layer, vulnerabilities such as misconfiguration of router, firewall and insecure communication were collected using metasploit software, while for application layer software vulnerabilities, privilege escalation, SQL injection, cross-site scripting, and privilege escalation are vulnerabilities considered and collected using Burp suite kit. The collective integrated of this data collected formed the primary dataset. The sample size of the data collected is 22090 records. The secondary dataset considered is the Common Vulnerability and Exploit (CVE) which is a popular vulnerability data which constitutes both old and new vulnerabilities from IoT infrastructure. The CVE contained 890,660 samples of vulnerability across 12 attributes. The total sample size of data collected is 901869 records.

2.2 Data Processing

The data collected was processed using imputation techniques as the first data processing step to remove missing and duplicate values. Normalization approach was then applied to balance the data and also dimensionality reduction. Random data upsampling and downsampling was applied for class balance in the dataset, before auto encoder was applied for data transformation and feature extraction to make the dataset compatible for training deep learning model. Collectively these data processing steps were applied in preparation of the dataset for the training of the deep leaning model for the generation of the DPI.

2.3 Modelling of the Convolutional Neural Network

This work utilized CNN as the deep learning model of choice due to its super feature extraction capabilities. The CNN is made of input layer, convolutional layer, fully connected layer and output layer as shown in the architectural model of Figure 1.

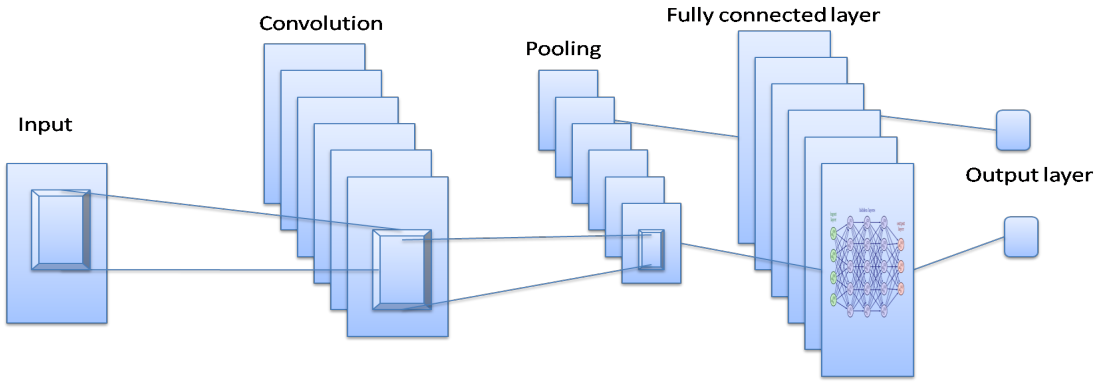

Figure 2: The CNN Architecture

CNN is input layer is responsible for the dimension of the input vulnerability data. The data are then feed to the convolutional layer through the application of convolutional scan performed with filters. The type of filter used for this process is the maximum pooling layers which identified the feature maps with the highest size and then extract. The Equation 1 represents the extracted feature maps (Sethi and Serrao, 2024).

$$F_m = \left[\frac{F_a + 2_b - k}{s}\right] + 1 \quad (1)$$

Where F_m is the output features, F_a is the input features, b is the convolutional padding, s is the strides size, k is the convolutional kernels size. The feature map extraction process to form the convolutional layer defined with Equation 2 (Ray, 2018);

$$C_O = ((w * h) + 1)\, nf) \quad (2)$$

Where Co is the total feature maps which formed the pixels in the convolutional layer, w is the filter weight, while h is the height, the number of filters used to extract the pixels are defined as nf and the bias term is 1. To compute the total number of convolutional extracted pixels in the next n convolutional layer, the model in Equation 3 was used (Sethi and Serrao, 2024);

$$C_O = ((w * h * np) + 1) * nf) \quad (3)$$

Where np is the number of pixels extracted and feed to the layer. The activation size is defined as Equation 4;

$$A_s = (w * h * d) \quad (4)$$

Where w is the weight, H is height and d is the depth of the image.

2.4 Modelling of the Ensemble Neural Network

The ensemble neural network integrated three neural network models. The building block of neural network constitutes the input layer which has neurons made of weights, bias and activation functions. A neuron is defined as Equation 5 (Nowak et al., 2024);

$$Y = f(wx_{ij} + b) \quad (5)$$

Where y is the output, x is the input matrix of size $(i * j)$, i is the data and j is the data features; w presents the weight of the neural network, b is the bias function and f represents the activation function. The neural network has three hidden layers defined as Equation 6 (Ntafloukas et al., 2022);

$$Y_L = f_l\,(w_l f_{l-1}(w_{l-1} f_{l-2}(\ldots..f_2(w_2 f_1(w_1 x + b_1) + b_2) \ldots..+b_{l-1}) + b_l) \quad (6)$$

Where y_l $donates$ the output, b_l is the bias of the hidden layer. The number of neurons in the input layer is determined by the 12 data attributes, while the activation function used is the hyperbolic tangent activation function. The architectural parameters like i is the neurons, n is the number of neurons, h is the hidden layers, h_n is the number of hidden layers, o is the output layer. Suppose the ensemble neural network is formed with the integration of three network

$N^m, m \in \{(\text{ANN\_A (1); ANN\_B(2), ANN\_C(3)})\ \}$ and Equation 6, each with three hidden layers. For the input $x \in R^d$, the m-th network is presented as Equation 7-10 (Ntafloukas et al., 2022);

$$h_1^{(m)} = f_1\, W_1^{(m)} x + b_1^{(m)} \quad (7)$$
$$h_2^{(m)} = f_2\, W_2^{(m)} h_1^{(m)} + b_2^{(m)} \quad (8)$$
$$h_3^{(m)} = f_3\, W_3^{(m)} h_2^{(m)} + b_3^{(m)} \quad (9)$$
$$y^m = f_{out}\, W_4^{(m)} h_3^{(m)} + b_4^{(m)} \quad (10)$$

Where $W_l^{(m)}$, $b_l^{(m)}$ are the weights and bias of the l-th layer in the network m, and f_1, f_2, f_3 are the hidden layers activation with $\mathbf{f_{out}}$ $\mathbf{which\ is\ the\ output\ activation\ function.}$ the weighed aggregation of the neuron network is presented with non negative weights of α_m as Equation 11 and the algorithm is presented in the stepwise section.

$$y_{ens}(X) = f_{ens}\ \alpha_1 y^1(x) + \alpha_2 y^2(x) + \alpha_3 y^3(x) \quad (11)$$

Stepwise of the ensemble neural network

1. Load data
2. Define and Build Three Independent ANN Architectures
3. Compile and Train Each ANN Separately
4. Generate Predictions from Each ANN
5. Combine Predictions Using Ensemble Technique
6. Soft Voting (Averaging Probabilities)
7. Evaluate the Ensemble Output
8. Model Comparison and Validation
9. Model Saving and Deployment

2.5 Modelling of the convolutional VulNet

The convolutional VulNet model is developed with the integration of the convolutional neural network model in Figure 1. The CNN model was applied for feature extraction, while the ensemble neural network model was applied for classification of the vulnerability. During the vulnerability classification, the average voting process is applied for the final decision of the vulnerability level in the healthcare environment. When vulnerability data are feed to the CNN, the feature vectors are extracted in one dimensional form, which was then applied to the ensemble model for training and classification of vulnerability. Figure 2 presents the architecture of the convolutional VulNet.

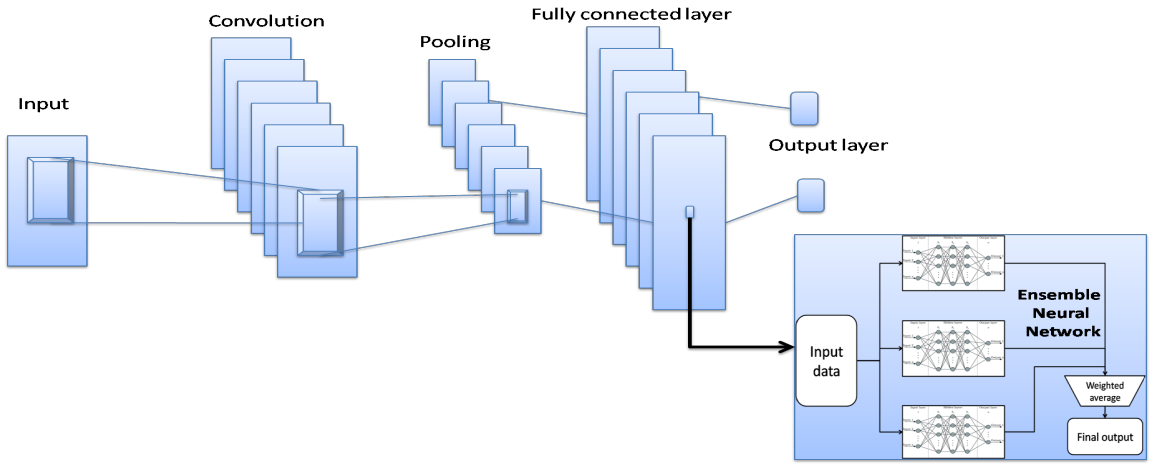

Figure 3: Architecture of the Convolutional VulNet

Convolutional Vulnet (CNN + Ensemble Integration) Stepwise

1. The CNN serves as the feature extractor from raw data.
2. Extracted features are passed to the ensemble neural network
3. The ensemble integrates multiple prediction outcomes
4. The VulNet outputs vulnerability classification results

2.6 Training of the convolutional VulNet

The training of the convolutional neural network with the ensemble neural network as the classifier, referred to as the Convolutional VulNet, was carried out using the CVE dataset. The dataset was pre-processed and then split into training, validation, and test sets in the ratio of 80:10:10 to ensure a balanced evaluation of the model. The convolutional layers were designed to automatically extract hierarchical vulnerability-related features, while the integrated ensemble neural network, placed within the classification layer, aggregated multiple neural decision boundaries to enhance detection accuracy and generalization. The implementation was carried out in Python 3.10 using the TensorFlow 2.x and Keras deep learning libraries, alongside scikit-learn for pre-processing and evaluation. GPU acceleration with CUDA-enabled Tensor Cores was employed to reduce training time. Data augmentation techniques such as normalization and feature scaling were applied to improve the robustness of the model. The training process adopted the Adam optimizer with an initial learning rate of 0.001, and the categorical cross-entropy loss function was used to handle multi-class classification of vulnerabilities.

Hyper parameters were carefully tuned, with a batch size of 64, a dropout rate of 0.5 in the fully connected layers to mitigate overfitting, and a ReLU activation function in hidden layers. The ensemble neural classifier employed three hidden layers with sizes 256, 128, and 64 neurons, respectively, before the SoftMax output layer. The model was trained for 50 epochs, with early stopping patience set to 10 epochs to prevent unnecessary computation when no improvement in validation loss was observed. Model evaluation metrics included accuracy, precision, recall, F1-score, and AUC-ROC, ensuring comprehensive assessment of vulnerability detection performance.

2.7 Development of the deep packet inspection model for vulnerability assessment

The Deep Packet Inspection (DPI) model for vulnerability assessment was developed by leveraging the trained convolutional VulNet, which combines convolutional neural networks with an ensemble neural network classifier. In this model, incoming data packets from network traffic are first captured and pre-processed into structured formats that preserve both content and metadata. The convolutional layers of the VulNet automatically extract discriminative spatial and temporal features from these packet representations, highlighting abnormal behaviours, irregular communication flows, and hidden attack signatures that may indicate security vulnerabilities. These extracted features are then passed to the ensemble neural network integrated within the classification layer, which aggregates multiple decision outcomes to provide a more reliable and generalized vulnerability classification.

Deep Packet Inspection (DPI) Model for Vulnerability Assessment Stepwise

1. Start
2. Real-time network traffic packets are captured
3. Packets are transformed into feature representations by the CNN.
4. CNN layers detect abnormal payload patterns and structural anomalies.
5. Ensemble NN classifies packets into vulnerability categories
6. The DPI model continuously monitors traffic
7. End

2.7.1 Development of the Vulnerability Decision Support Scoring System

To develop the vulnerability decision support scoring system, the data used was collected from the national vulnerability database (NVD, 2025) as reported in Table 1 with common vulnerability severity rating score. The data was then used to develop the rule-based decision support algorithm or vulnerability scoring.

Table 1: Common Vulnerability Severity Rating (NVD, 2025)

Severity	Score	Risk level	Impact
Low	0.0-1.5	Low	Low
Very Low	1.6-3.9	Low	Low
Medium	4.0-6.9	Medium	Medium
High	7.0-8.0	High	High
Very high	9.0-10	Critical	Very high

2.7.2 Development of Vulnerability Assessment and Management System

The vulnerability assessment and management system was developed as an integrated framework that combines both classification and decision-making functionalities to ensure robust identification, evaluation, and prioritization of vulnerabilities.

The system architecture is composed of two primary modules of deep packet inspection-based vulnerability classification engine, and rule-based vulnerability decision scoring system. The classification engine leverages the ConvulNet deep learning model to analyze packet-level features and identify potential vulnerabilities with high accuracy. This stage ensures that traffic anomalies, misconfiguration, and exploit patterns are effectively detected and mapped into known vulnerability categories. ConvulNet's ability to extract hierarchical patterns from structured datasets enhances detection performance, particularly for zero-day and complex vulnerabilities.

The second stage of the system integrates the decision scoring mechanism, which was designed using a rule-based approach with a scale of 1–10. Once vulnerabilities are classified, the decision engine assigns severity scores based on predefined rules that reflect parameters such as exploitability, impact on confidentiality, integrity, and availability, as well as contextual system importance. This scoring provides a quantifiable metric for ranking vulnerabilities, enabling security analysts to prioritize remediation efforts effectively. The integration of these modules produces a unified vulnerability assessment and management system capable of not only identifying and classifying vulnerabilities but also evaluating their severity in real time.

Stepwise Development of Vulnerability Assessment and Management System

1. Start
2. Collect raw packet data
3. Extract relevant features with CNN
4. Normalize and encode the dataset into a structured format
5. Use the trained ConvulNet model to classify packets into different vulnerability
6. Feed classification results into the decision scoring system.
7. Apply predefined rules (scale of 1–10)
8. Assign a severity score to detected vulnerability.
9. Rank vulnerabilities based on severity scores.
10. Group them into categories:
11. Low Risk (1–3) ⟶ low severity
12. Medium Risk (4–6) ⟶ High severity
13. High Risk (7–8) ⟶ High severity
14. Critical Risk (9–10) ⟶ critical severity
15. Log results into a centralized vulnerability management dashboard.
16. Send alerts to system administrators with detailed assessment reports.
17. Continuous Learning and Feedback
18. End

3. SYSTEM RESULTS

The system testing presents the results of the CNN extractor, the results of the ensemble neural network training, and the overall system performance in terms of classification accuracy, precision, recall, and loss evaluation. During the testing phase, the CNN model was first employed as a feature extractor, generating high-level feature representations from the input dataset. These extracted features served as the input to the ensemble neural network, which combined the outputs of three independently trained artificial neural networks to improve generalization and reduce the likelihood of overfitting.

The ensemble network was tested using both validation and unseen test datasets to measure its robustness. The results indicated that the CNN extractor successfully captured critical features necessary for accurate classification, while the ensemble architecture demonstrated improved predictive stability compared to individual models. The system achieved consistent accuracy across multiple testing rounds, with relatively lower variance in loss values, confirming the effectiveness of the ensemble approach. Overall, the testing phase validates that the integration of the CNN feature extractor with the ensemble neural network provides a reliable and scalable solution for the intended application.

3.1 Result of the CNN extractor training

This section presents the results of the CNN extractor during the training process considering accuracy and loss. The training loss, validation loss and accuracy across 60 epochs were recorded in Table 2.

Table 2: Result of the CNN extractor training

Epoch	Train_Loss	Val_Loss	Accuracy
1	0.215563	0.249828	0.839862
2	0.203198	0.208221	0.847596
3	0.183709	0.230981	0.843351
4	0.160688	0.172524	0.849875
5	0.174298	0.210902	0.846382
6	0.173623	0.206231	0.857462
7	0.171043	0.191613	0.847185
8	0.129441	0.173197	0.861445
9	0.162488	0.205982	0.856092
10	0.148	0.152202	0.877926
11	0.15403	0.202416	0.861341
12	0.122488	0.160883	0.865162
13	0.134127	0.196792	0.868759
14	0.113719	0.141303	0.87983
15	0.133102	0.151839	0.87367
16	0.097138	0.090316	0.865271
17	0.106382	0.082898	0.873898
18	0.11027	0.177311	0.856226
19	0.09071	0.154138	0.871307
20	0.084722	0.123694	0.872559
21	0.097131	0.114892	0.87754
22	0.105881	0.147879	0.872396
23	0.075575	0.126579	0.875894
24	0.090935	0.139048	0.877467
25	0.069845	0.097713	0.880886

Epoch	Train_Loss	Val_Loss	Accuracy
26	0.116795	0.153509	0.871018
27	0.082934	0.122822	0.879126
28	0.024405	0.072518	0.87966
29	0.075929	0.079573	0.885131
30	0.078548	0.113867	0.881017
31	0.060095	0.075292	0.869561
32	0.060721	0.084094	0.877531
33	0.05566	0.07212	0.888472
34	0.076552	0.087369	0.886851
35	0.03422	0.076525	0.885497
36	0.030582	0.042836	0.884979
37	0.038551	0.043687	0.888009
38	0.085389	0.110051	0.884315
39	0.046948	0.104113	0.880769
40	0.041214	0.065867	0.883853
41	0.041357	0.097777	0.876178
42	0.093057	0.086712	0.884331
43	0.001	0.036627	0.893448
44	0.067963	0.203012	0.882346
45	0.035325	0.053051	0.877086
46	0.064939	0.093993	0.884931
47	0.038765	0.031338	0.874113
48	0.058797	0.09401	0.893065
49	0.016901	0.050638	0.896381
50	0.068261	0.102539	0.865927
51	0.03186	0.082776	0.877402
52	0.010232	0.016387	0.883166
53	0.00304	0.058008	0.86466
54	0.042928	0.109713	0.883527
55	0.026397	0.068428	0.887719
56	0.052838	0.057705	0.890184
57	0.021845	0.007526	0.875003
58	0.016448	0.08307	0.876535
59	0.032456	0.055595	0.885237
60	0.032456	0.055595	0.885237

Table 2 reveal a consistent trend of performance improvement across the epochs. At the initial stages (epochs 1–10), the CNN exhibited relatively higher training and validation losses (e.g., Train Loss = 0.2156, Val Loss = 0.2498, Accuracy = 83.9% at epoch 1), reflecting the early learning phase of the model. However, by epoch 10, the accuracy had increased to 87.7%, accompanied by a notable reduction in both training and validation losses, signifying effective feature learning. As training progressed (epochs 11-30), the model stabilized further, with the validation loss reducing consistently and the accuracy improving beyond 88%. By epoch 28, the model achieved an accuracy of 87.96% with very low training and validation losses (Train Loss = 0.0244, Val Loss = 0.0725), indicating that the CNN was learning generalized features effectively.

In the later epochs (31–60), the CNN reached its peak performance, where both training and validation losses converged towards near-zero values. For instance, at epoch 43, the CNN achieved its highest recorded accuracy of 89.34% with an exceptionally low training loss of 0.001 and validation loss of 0.0366. Similarly, accuracy values remained above 88% consistently between epochs 33 and 60, showing strong stability in learning. Figure 3 presents the graphical analysis of the CNN extractor training accuracy.

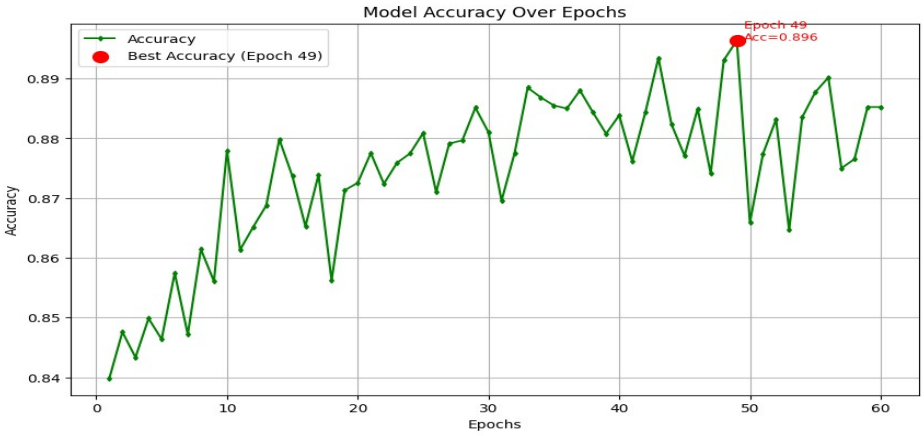

Figure 4: Accuracy of the CNN training performance

Figure 5: Training and validation loss

Figure 3 presents the accuracy performance of the CNN extractor across 60 epochs. The results during the training process were generated when the model reaches stopping criteria which is the epoch 49 (point where the best version of the model was generated). From the results the accuracy of the CNN extractor is 0.896, which is 89.6%. The results implied that the CNN was able to correctly extract features maps of vulnerability to train the neural network models. The loss function recorded 0.051 which is very good as it is approximately zero and implied that the error when identifying features for extraction is tolerable.

3.2 Result of the Ensemble Neural Network Training

This section presents the result of the neural network training process. The training was done experimentally by training the three different neural network algorithms and then computing the average as the ensemble classifier for the detection of vulnerabilities. The result of the first neural network training ANN_A was reported in Table 3.

Table 3: Result of the ANN_A Training

Epoch	Train_Accuracy	Train_Loss	Val_Accuracy	Val_Loss
1	0.3155	1.3652	0.3938	1.332
2	0.4768	1.1845	0.4953	1.2505
3	0.5641	1.0104	0.5503	1.0822
4	0.6323	0.8535	0.49	0.9475
5	0.6756	0.7386	0.4769	0.9244
6	0.7296	0.6277	0.574	0.7576
7	0.7843	0.5267	0.7398	0.5104
8	0.84	0.424	0.8838	0.3551
9	0.8831	0.3362	0.9195	0.2735
10	0.899	0.2756	0.952	0.2102
11	0.911	0.2371	0.9581	0.1771
12	0.9206	0.2098	0.9565	0.1611
13	0.93	0.1835	0.9517	0.1546
14	0.9306	0.181	0.9637	0.1351
15	0.9327	0.1656	0.9507	0.1412
16	0.9343	0.1597	0.9608	0.1245
17	0.9388	0.1492	0.964	0.1137
18	0.9475	0.136	0.9712	0.1024
19	0.9451	0.1347	0.9675	0.0968
20	0.9405	0.1346	0.9715	0.0912
21	0.9516	0.1256	0.9763	0.0845
22	0.9501	0.1201	0.9755	0.086
23	0.955	0.1147	0.9781	0.0799
24	0.9553	0.1113	0.9744	0.079
25	0.9556	0.107	0.9784	0.0748

Epoch	Train_Accuracy	Train_Loss	Val_Accuracy	Val_Loss
26	0.957	0.1042	0.9824	0.0689
27	0.9604	0.0972	0.9797	0.0663
28	0.9593	0.0977	0.9904	0.0589
29	0.9627	0.0907	0.9819	0.0604
30	0.9653	0.0861	0.9869	0.058

The performance results of ANN_A in Table 3 across the 30 training epochs demonstrate a strong and consistent learning pattern. At the beginning of training, the model started with a very low accuracy of about 31.6% on the training set and 39.4% on the validation set, which is expected since the network was essentially making near-random predictions at that stage. As training progressed through the early epochs, both training and validation accuracies improved significantly, with training accuracy reaching above 70% and validation accuracy climbing above 55% by the sixth epoch. This steady rise in accuracy, accompanied by a noticeable decline in both training and validation losses, indicates that the model quickly began capturing meaningful feature representations from the extracted inputs.

During the middle stages of training, roughly between the 7th and 15th epochs, the model's performance improved further, achieving over 93% training accuracy and consistently surpassing 95% validation accuracy by the 14th epoch. At this point, validation accuracy actually exceeded training accuracy, which is often a positive sign, suggesting that the model was not only learning but also generalizing well to unseen data. The validation loss remained slightly lower than the training loss, showing no signs of severe overfitting, although a small performance gap began to appear, which is typical in neural networks as they start to converge.

In the final training phase, from epochs 16 to 30, the network stabilized at a very high-performance level. Training accuracy capped at around 96.5%, while validation accuracy peaked at an impressive 99.0% at the 28th epoch, before finishing around 98.7%. The validation loss continued to decrease, reaching as low as 0.058, which is evidence of strong generalization. During the testing of the model, the accuracy recorded 0.9931 which is 99.31 success and loss of 0.0176. Overall, the results show that ANN_A is a well-trained and robust model. It demonstrates a smooth and reliable learning trajectory, avoids overfitting despite extended training, and achieves excellent generalization with very high validation accuracy. This makes it a strong candidate for inclusion in the ensemble neural network since it contributes not only accurate but also stable predictions across unseen data. Table 4 presents the result of the ANN_B training.

Table 4: Result of the ANN_B Training

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1/40	0.6877	0.7491	0.9739	0.0817
2/40	0.9610	0.1168	0.9891	0.0416
3/40	0.9755	0.0730	0.9859	0.0366
4/40	0.9792	0.0564	0.9896	0.0287
5/40	0.9839	0.0437	0.9901	0.0257
6/40	0.9864	0.0362	0.9912	0.0235
7/40	0.9874	0.0331	0.9901	0.0234
8/40	0.9895	0.0293	0.9915	0.0221
9/40	0.9893	0.0282	0.9896	0.0236
10/40	0.9895	0.0262	0.9909	0.0225
11/40	0.9906	0.0249	0.9904	0.0224
12/40	0.9897	0.0265	0.9896	0.0248
13/40	0.9915	0.0247	0.9896	0.0233

The result presented in Table 4: Result of the ANN_B Training clearly demonstrates the progressive improvement of the network's learning ability across epochs. At the beginning of training (Epoch 1), the model's training accuracy was relatively low at 0.6877, with a high training loss of 0.7491, indicating that the network was just beginning to learn the underlying patterns in the data. Interestingly, the validation accuracy at this stage was already very high (0.9739), with a low validation loss (0.0817). This suggests that the features extracted from the CNN provided a strong and separable representation, allowing ANN_B to generalize well even in the early stages.

As training progressed, there was a rapid increase in accuracy, with the model reaching 0.9610 accuracy by Epoch 2 and a corresponding significant drop in training loss to 0.1168. At the same time, validation accuracy improved further to 0.9891, and validation loss decreased to 0.0416, indicating that the network was not only fitting the training data but also maintaining strong generalization to unseen samples. From Epochs 3 to 8, the training accuracy consistently improved, reaching values above 0.98, while both training and validation losses decreased steadily, showing a healthy convergence pattern.

Between Epochs 9 and 13, the model's performance stabilized with accuracy values fluctuating slightly between 0.9893–0.9915 and validation accuracy ranging between 0.9896–0.9909. The training and validation losses in this stage remained low, hovering around 0.022–0.028, which reflects that the network had reached a plateau in its learning, converging to a high-performance solution. The absence of sharp divergence between training and validation performance confirms that the network was not overfitting during these epochs. The testing result recorded 0.9928 accuracy and loss of 0.0195.

Overall, the training history of ANN_B demonstrates that the model achieved very high accuracy (≈ 99%) with minimal loss values after only a few epochs, thanks to the effective feature representation provided by the CNN. The results highlight that ANN_B is highly efficient in learning from extracted features, achieving a good balance between fitting the training data and maintaining strong generalization on validation data. This strong consistency between training and validation metrics validates the robustness and reliability of ANN_B as a component of the ensemble framework. Table 5 presents the training result of the ANN_C.

Table 5: Training result of the ANN_C

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1/40	0.6026	0.9814	0.9501	0.2094
2/40	0.9512	0.1921	0.9720	0.1177
3/40	0.9708	0.1164	0.9779	0.0827
4/40	0.9766	0.0859	0.9829	0.0643
5/40	0.9824	0.0663	0.9901	0.0534
6/40	0.9834	0.0583	0.9909	0.0463
7/40	0.9849	0.0501	0.9920	0.0415
8/40	0.9879	0.0433	0.9923	0.0377
9/40	0.9872	0.0403	0.9923	0.0351
10/40	0.9882	0.0381	0.9923	0.0329
11/40	0.9887	0.0362	0.9915	0.0314
12/40	0.9909	0.0321	0.9917	0.0299
13/40	0.9906	0.0308	0.9917	0.0293
14/40	0.9909	0.0292	0.9915	0.0280
15/40	0.9905	0.0278	0.9915	0.0273
16/40	0.9902	0.0279	0.9909	0.0267
17/40	0.9906	0.0269	0.9920	0.0262
18/40	0.9918	0.0260	0.9915	0.0258
19/40	0.9924	0.0243	0.9917	0.0250
20/40	0.9915	0.0246	0.9917	0.0246
21/40	0.9921	0.0236	0.9920	0.0244
22/40	0.9922	0.0229	0.9915	0.0239
23/40	0.9924	0.0227	0.9915	0.0242
24/40	0.9932	0.0218	0.9920	0.0232
25/40	0.9932	0.0215	0.9904	0.0236
26/40	0.9935	0.0215	0.9920	0.0230
27/40	0.9929	0.0208	0.9912	0.0232
28/40	0.9936	0.0204	0.9909	0.0231
29/40	0.9931	0.0199	0.9915	0.0226
30/40	0.9931	0.0198	0.9904	0.0229
31/40	0.9932	0.0201	0.9915	0.0219
32/40	0.9935	0.0198	0.9912	0.0220
33/40	0.9930	0.0193	0.9912	0.0219
34/40	0.9937	0.0180	0.9909	0.0219
35/40	0.9930	0.0189	0.9907	0.0218
36/40	0.9931	0.0191	0.9912	0.0213
37/40	0.9933	0.0189	0.9907	0.0218
38/40	0.9936	0.0178	0.9907	0.0215
39/40	0.9938	0.0176	0.9915	0.0210
40/40	0.9927	0.0191	0.9907	0.0210

Training Result of ANN_C in Table 5 demonstrates a smooth and consistent improvement in the model's learning behavior across the 40 epochs. At the beginning of training (Epoch 1), ANN_C achieved a moderate training accuracy of 0.6026 with a relatively high training loss of 0.9814. However, its validation performance was already quite strong, with 0.9501 validation accuracy and a relatively low validation loss of 0.2094. This early strong generalization suggests that the input features (extracted from the CNN) were already highly discriminative, giving the model a head start compared to training accuracy. From Epoch 2 to 5, the network experienced a steep rise in training accuracy, reaching 0.9824, while training loss decreased significantly to 0.0663. At the same time, validation accuracy improved from 0.9720 to 0.9901, with validation loss steadily reducing from 0.1177 to 0.0534. This phase shows that the model quickly learned to map the extracted features to the correct outputs, achieving near-optimal generalization within just a few epochs.

Between Epochs 6 and 15, ANN_C continued to refine its learning. Training accuracy stabilized above 0.98, with training loss reducing gradually to around 0.0278. Validation accuracy during this stage consistently remained above 0.990, while validation loss is around 0.026–0.030, reflecting strong and balanced learning. Importantly, there were no signs of overfitting, as the validation metrics closely tracked the improvements in training performance. From Epoch 16 to 40, the model converged toward its optimal performance. Training accuracy reached 0.9938 (Epoch 39) with a minimal loss of 0.0176, while validation accuracy remained stable around 0.991–0.992, with a validation loss as low as 0.0210. This plateau suggests that the network had fully exploited the discriminative power of the extracted CNN features and achieved peak performance. The minor fluctuations in later epochs are expected and reflect normal training dynamics rather than instability.

3.3 Comparative Analysis

This section compared the results obtained with the three different ANN models and then ensemble model as reported in Figure 5 and 6 respectively.

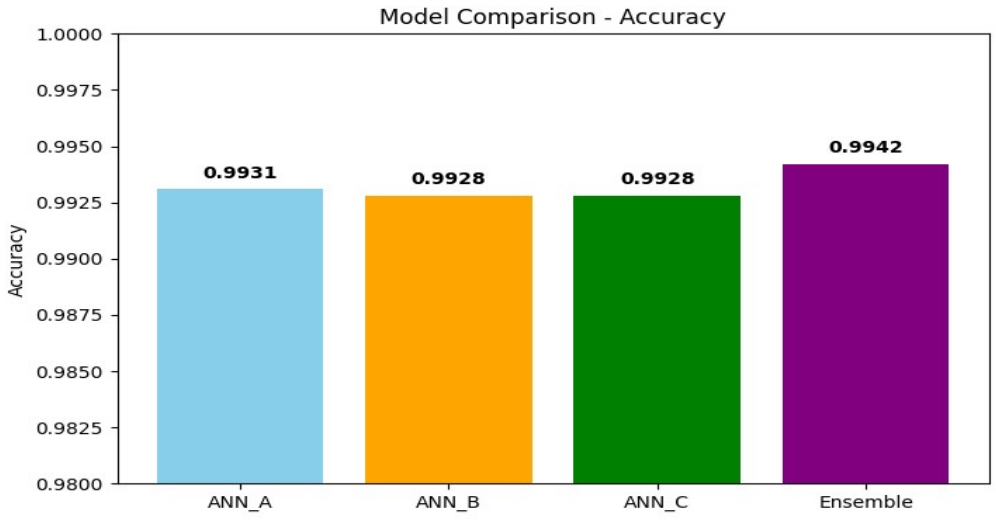

Figure 6: Comparative Accuracy

The analysis of the graphs and values shows a very strong and consistent performance across the three individual ANN models, with the ensemble integration providing an additional improvement for accuracy in Figure 5 and loss in Figure 6. For ANN_A, the accuracy is 0.9931 with a corresponding loss of 0.0176. This combination indicates excellent classification performance and efficient learning, as the model achieves high accuracy while maintaining a low error value.

ANN_B records an accuracy of 0.9928 and the highest loss value among the three, 0.0195. Although its accuracy is still very strong, the slightly higher loss suggests that the model's internal error during optimization was a bit larger compared to the others, which may imply less stability during training. ANN_C achieves the same test accuracy as ANN_B (0.9928) but with a lower loss of 0.0180, indicating better optimization and more balanced learning compared to ANN_B. This suggests ANN_C converged more efficiently despite ending with the same predictive accuracy. The ensemble model demonstrates the best overall performance, with the highest accuracy (0.9942) and the lowest loss (0.0165). This result confirms the strength of ensemble learning, as the averaging of probability outputs smoothes out the small weaknesses of each individual model, reduces variance, and achieves a more stable and reliable prediction.

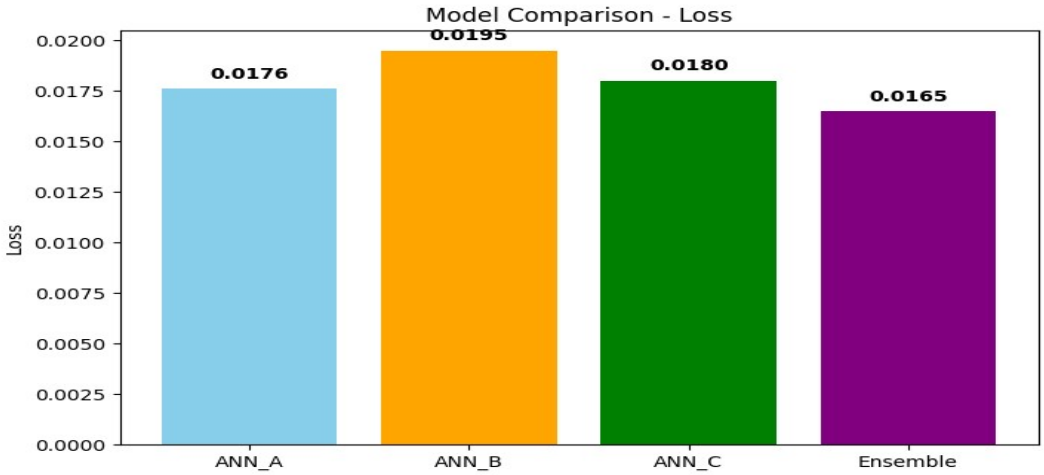

Figure 7: Comparative Loss Results

In summary, the graphs show that while all three ANN models individually perform well above 99% accuracy, the ensemble model clearly enhances both performance and stability. It achieves the lowest error rate and highest predictive power, making it the most reliable approach for vulnerability classification in this context. In summary, the ensemble's performance highlights the effectiveness of combining multiple ANNs for improved classification stability. It minimizes the weaknesses of individual models, achieving near-perfect accuracy while providing confidence in the reliability of its predictions across all severity categories.

3.4 Comparative Analysis with Existing Algorithms

This section compare the results of the four convolutional vulnet models developed with ANN_A, ANN_B, ANN_C and the ensemble model with other results in literature review. The results are reported in Table 6.

Table 6: Comparative results with other works in literature review

Models	Technique	Accuracy
Nowak et al., (2023)	PNN	91.00
Jeon and Kim (2021)	LSTM+GRU	96.11
Ogundairo and Broklyn (2024)	CNN	90.00
Subhan et al., (2022)	CNN+LSTM	91.00
Fu et al., (2021)	BiLSTM	88.55.
	LSTM	88.48
An et al., (2023)	V-CNN	95
ConVulNet (our model)	ANN_A	99.31
	ANN_B	99.28
	ANN_C	99.28
	Ensemble	99.42

The comparative results presented in Table 4.5 clearly highlight the superiority of our convolutional vulnet over existing works in the literature. The earlier models demonstrate good performance but generally fall within the 88-96% accuracy range. For example, Fu et al. (2021) achieved accuracies of 88.55% with BiLSTM and 88.48% with LSTM, while Ogundairo and Broklyn (2024) obtained 90% using CNN. Similarly, hybrid models like Subhan et al. (2022) with CNN+LSTM improved accuracy to 91%, and Jeon and Kim (2021) reached 96.11% using LSTM+GRU, representing one of the stronger baselines. An et al. (2023) also advanced the performance with a V-CNN model achieving 95% accuracy, which is competitive but still below your results.

In contrast, our models ANN_A (99.31%), ANN_B (99.28%), and ANN_C (99.28%) all outperform these state-of-the-art techniques by a notable margin. Even more significant is the ensemble model, which achieved the highest accuracy of 99.42%, surpassing all other individual and hybrid models in the comparison. This result demonstrates the effectiveness of the ensemble approach, as it not only improves stability but also reduces prediction variance compared to the standalone ANNs.

4. CONCLUSION

The explosion of IoT devices in healthcare like smart monitors, wearable sensors, and connected medical tools has made things very convenient, but it's also opened the door to some serious security headaches. These vulnerabilities lurk across every layer: the physical hardware, the networks linking it all, and the apps running the show. Old-school vulnerability management tools? They're mostly reactive, glued to dusty databases, and prone to missing threats or crying wolf with false alarms, leaving hospitals and clinics wide open to attacks that could disrupt life-saving care. Therefore, in this study, it is set out to develop and test a smarter, faster system tailored for healthcare IoT, blending artificial intelligence with deep packet inspection (DPI) which is vigilant traffic cop scanning every bit of network data in real time. At its heart is our new creation, Convolutional VulNet: a deep learning setup that uses a Convolutional Neural Network (CNN) to extract key patterns from the data, then hands them off to a team of Artificial Neural Networks (ANNs) working together like a dream ensemble for spot-on decisions.

Then, the system is further fed with a massive dataset of 901,869 vulnerability records pulled straight from a real hospital's IoT setup and the gold-standard Common Vulnerabilities and Exposures (CVE) database. The system also packs in a live DPI engine to sniff out issues on the fly and a straightforward rule-based scorer to rank threats by urgency, helping teams prioritize what to fix first. The results attained from the study reported that the CNN performed feature extraction at 89.6% accuracy. Then, the ensemble neural network achieved 99.42% accuracy on fresh test data with a rock-bottom loss rate of just 0.0165 which basically, it barely makes mistakes. When the proposed model was compared with existing models, such as LSTM+GRU setups (96.11% accuracy) or basic CNNs (90%), the proposed model pulled way ahead, proving it's not just hype. Bottom line: Convolutional VulNet delivers a tough, precise, and easy-to-deploy toolkit for tackling vulnerabilities head-on in healthcare IoT. It shows how mixing CNN smarts with ensemble teamwork and DPI can supercharge threat hunting, keeping patient info safe and medical services running smoothly.

REFERENCES

Ahanger, T. A., Aljumah, A., & Atiquzzaman, M. (2022). State-of-the-art survey of artificial intelligent techniques for IoT security. *Computer Networks, 206*, Article 108771. https://doi.org/10.1016/j.comnet.2022.108771

Alaoui, R. L., & Nfaoui, E. H. (2022). Deep learning for vulnerability and attack detection on web applications: A systematic literature review. *Future Internet, 14*(4), Article 118. https://doi.org/10.3390/fi14040118

An, J. H., Wang, Z., & Joe, I. (2023). A CNN-based automatic vulnerability detection. *EURASIP Journal on Wireless Communications and Networking, 2023*, Article 28. https://doi.org/10.1186/s13638-023-02255-2

Arampatzis, A. (2022, December 29). *Top 10 vulnerabilities that make IoT devices insecure*. Venafi. https://venafi.com/blog/top-10-vulnerabilities-make-iot-devices-insecure/

Baho, S. A., & Abawajy, J. (2023). Analysis of consumer IoT device vulnerability quantification frameworks. *Electronics, 12*(5), Article 1176. https://doi.org/10.3390/electronics12051176

Bertino, E., & Islam, N. (2017). Botnets and internet of things security. *Computer, 50*(2), 76–79. https://doi.org/10.1109/MC.2017.62

Bhatt, V., & Chakraborty, S. (2023). Improving service engagement in healthcare through internet of things based healthcare systems. *Journal of Science and Technology Policy Management, 14*(1), 53–73. https://doi.org/10.1108/JSTPM-03-2021-0040

Fu, X., Gu, Z., Han, W., Qian, Y., & Wang, B. (2021). Exploring security vulnerabilities of deep learning models by adversarial attacks. *Wireless Communications and Mobile Computing, 2021*, Article 9969867. https://doi.org/10.1155/2021/9969867

Hassani, H. L., Bahnasse, A., Martin, E., Roland, C., Bouattane, O., & Diouri, M. E. (2021). Vulnerability and security risk assessment in a IIoT environment in compliance with standard IEC 62443. *Procedia Computer Science, 191*, 33–40. https://doi.org/10.1016/j.procs.2021.07.008

Hulayyil, S. B., Li, S., & Xu, L. (2023). Machine-learning-based vulnerability detection and classification in Internet of things device security. *Electronics, 12*(18), Article 3927. https://doi.org/10.3390/electronics12183927

Jeon, S., & Kim, H. K. (2021). AutoVAS: An automated vulnerability analysis system with a deep learning approach. *Computers & Security, 106*, Article 102308. https://doi.org/10.1016/j.cose.2021.102308

Madanian, S., Chinbat, T., Subasinghage, M., Airehrour, D., Hassandoust, F., & Yongchareon, S. (2024). Health IoT threats: Survey of risks and vulnerabilities. *Future Internet, 16*(11), Article 389. https://doi.org/10.3390/fi16110389

Madushan, H., Salam, I., & Alawatugoda, J. (2022). A review of the NIST lightweight cryptography finalists and their fault analyses. *Electronics, 11*(24), Article 4199. https://doi.org/10.3390/electronics11244199

Meidan, Y., Sachidananda, V., Peng, H., Sagron, R., Elovici, Y., & Shabtai, A. (2020). A novel approach for detecting vulnerable IoT devices connected behind a home NAT. *Computers & Security, 97*, Article 101968. https://doi.org/10.1016/j.cose.2020.101968

Naeem, H., & Alalfi, M. H. (2020). Identifying vulnerable IoT applications using deep learning. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 582–586). IEEE. https://doi.org/10.1109/SANER48275.2020.9054817

Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., & Ghani, N. (2019). Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials, 21*(3), 2702–2733. https://doi.org/10.1109/COMST.2019.2910750

Ntafloukas, K., McCrum, D. P., & Pasquale, L. (2022). A cyber-physical risk assessment approach for Internet of things enabled transportation infrastructure. *Applied Sciences, 12*(18), Article 9241. https://doi.org/10.3390/app12189241

Nowak, M. R., Walkowski, M., & Sujecki, S. (2023). Support for the vulnerability management process using conversion CVSS base score 2.0 to 3.x. *Sensors, 23*(4), Article 1802. https://doi.org/10.3390/s23041802

Ogundairo, O., & Broklyn, P. (2024). Automated vulnerability assessment using machine learning. *EasyChair Preprint no. 14347*.

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences, 30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Seara, J. P., & Serrão, C. (2024). Automation of system security vulnerabilities detection using open-source software. *Electronics, 13*(5), Article 873. https://doi.org/10.3390/electronics13050873

Subhan, F., Wu, X., Bo, L., Sun, X., & Rahman, M. (2022). A deep learning-based approach for software vulnerability detection using code metrics. *IET Software, 16*(6), 516–526. https://doi.org/10.1049/sfw2.12066

Taşcı, B. (2024). Deep-learning-based approach for IoT attack and malware detection. *Applied Sciences, 14*(18), Article 8505. https://doi.org/10.3390/app14188505

Usak, M., Kubiatko, M., Shabbir, M. S., Dudnik, O. V., Jermsittiparsert, K., & Rajabion, L. (2020). Health care service delivery based on the Internet of things: A systematic and comprehensive study. *International Journal of Communication Systems, 33*(2), Article e4179. https://doi.org/10.1002/dac.4179

Wenguang, S., Mykola, B., Przystupa, K., Beshley, H., Kochan, O., Pryslupskyi, A., Pieniak, D., & Su, J. (2020). A software deep packet inspection system for network traffic analysis and anomaly detection. *Sensors, 20*(3), Article 642.

Yu, M., Zhuge, J., Cao, M., Shi, Z., & Jiang, L. (2020). A survey of security vulnerability analysis, discovery, detection, and mitigation on IoT devices. *Future Internet, 12*(2), Article 27. https://doi.org/10.3390/fi12020027