

AI Model for Early Detection of Cyber Attacks in IoT

Sangita Sri Harshitha*, Dr. D. Latha

Computer Science and Engineering, Adikavi Nannaya University, Rajamahendravaram, Andhra Pradesh, India.

ABSTRACT

The number of IoT devices in homes, offices, and factories are now full of small internet-connected devices, and a large share of them run with weak passwords and firmware that never gets patched. Attackers know this. Botnets, DDoS floods, and quiet unauthorized access have become routine on IoT networks, and the usual intrusion detection tools tend to raise an alarm only after the damage has started. This paper takes a different route: network traffic is treated as a sequence rather than a pile of unrelated records. A sliding window gathers ten consecutive flow records at a time, and a Bidirectional Long Short-Term Memory (BiLSTM) network with an attention layer reads each window, picking up the slow, gradual signs of trouble before the attack itself lands. Training and testing use the CICIDS2017 dataset under a strict time-ordered split. The result is a practical and scalable way to catch attacks on IoT networks early instead of cleaning up after them.

Keywords: - Internet of Things, LSTM, Attention Mechanism, Deep Learning, Network Security, CICIDS2017

INTRODUCTION

Cameras, smart locks, thermostats, baby monitors, factory sensors: ordinary objects like these now sit on the internet in the billions, and together they make up what we call the Internet of Things, or IoT. Cheap hardware keeps the numbers growing. Cheap hardware also keeps the security weak. A typical IoT device ships with a default password, receives few or no updates, and was never designed with an attacker in mind. Once one device is compromised, it becomes a stepping stone toward everything else on the network.

The classic abuse of hijacked IoT devices is the botnet. Mirai is the textbook case it crawled the internet for devices with open Telnet ports and factory logins, took them over by the thousand, and then aimed all of that combined traffic at chosen websites until they fell over. What matters for this paper is the timeline. An attack like that is not instantaneous. There is a scanning phase, then a break-in, then a quiet stretch where the infected device checks in with its control server, and only after all of that does the loud part begin. Each early phase leaves faint traces in network traffic. Catch those

traces and the attack can be stopped while it is still cheap to stop. Most detection tools do not work this

way. They inspect one traffic record at a time, compare it against a rule or a trained model, and decide on the spot. The trouble is that a slow port scan or a brief check in with a command server rarely looks strange in isolation. The oddness only shows up across a run of records, as a pattern that builds. A per-record classifier has no memory of what came before, so it misses exactly the behaviour that early detection depends on.

Our model reads traffic in short sequences instead. Ten consecutive records form a window, and the whole window goes into a Bidirectional Long Short-Term Memory (BiLSTM) network. An attention layer on top learns which records inside the window carried the most weight in the decision, so every prediction comes with a rough pointer to its evidence rather than a bare label. We train on the CICIDS2017 dataset, which has labeled normal and attack traffic, and we test on data that comes strictly later in time than anything seen during training. That setup is closer to how such a system would actually be deployed.

This paper contributes: (1) an end-to-end pipeline that turns raw network traffic into short labeled windows and classifies them with a BiLSTM plus attention; (2) an evaluation split ordered by time, not shuffled at random, to better match deployment conditions; (3) a decision threshold tuned on validation data instead of

Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

an assumed 0.5; and (4) a frank account of the results, covering the weak spots along with the strong ones.

Section II reviews related work. Section III describes the system as a whole, Section IV covers the model in detail, and Section V reports the experiments.

References close the paper.

II. LITERATURE REVIEW

A. Early Approaches to Detecting Attacks

The first generation of intrusion detection tools relied on signatures. Traffic was matched against a hand-maintained list of known attack fingerprints, which worked until an attacker did something the list had never seen, and the list itself had to be updated by hand forever. Machine learning loosened this constraint by letting a model learn the patterns from data. Random Forest, Support Vector Machines, and Multi-Layer Perceptrons have all been applied this way, usually with each traffic flow flattened into a row of numbers and labeled normal or attack. Accuracy can be respectable. Even so, every flow is judged on its own, and nothing that happened one record earlier is available to the model.

B. Deep Learning for Network Security

Deep learning removed some of the manual feature work, instead of hand-picking which measurements matter, a network can learn useful patterns from raw or lightly processed data. McDermott, Majdani, and Petrovski applied deep networks directly to IoT botnet traffic, borrowing word-embedding ideas from natural language processing to turn packet data into inputs a network could digest. The Long Short-Term Memory architecture of Hochreiter and Schmidhuber gave networks a working memory across time steps, and Graves and Schmidhuber later found that reading a sequence in both directions beats reading it forward only. That bidirectional variant is the backbone of the model in this paper.

C. Attention Mechanisms

A plain LSTM compresses an entire sequence into one fixed-size summary before deciding, and the compression can drown out the one or two moments that actually mattered. Attention, introduced by Bahdanau, Cho, and Bengio for machine translation and pushed much further by Vaswani et al. in

“Attention Is All You Need,” assigns each time step its own learned weight instead of treating all steps the same. Language tasks adopted the idea almost universally. Network security has been slower to pick it up, even though an attack window tends to hinge on a small number of unusual records, which is precisely the situation attention was built for.

D. Early-Stage and Fast Detection of IoT Botnets

Some recent work targets early detection directly rather than waiting for a full attack to unfold. Wei, Xie, and Diao built a lightweight framework for the IoT edge that works with a five-second detection window and never reads packet payloads. Nguyen et al. tested a collaborative setup in which classic algorithms such as Support Vector Machines and K-Nearest Neighbors run across multiple participants to spot botnets sooner. A related paper by the same group mixed static and dynamic analysis with sub-graph features rooted in program similarity, raising precision for industrial IoT. Bojarajulu, Tanwar, and Singh paired Bidirectional Gated Recurrent Units with recurrent networks under a bio-inspired tuning scheme they call SMIE and reported further gains in accuracy.

Amara Korba, Diaf, and Ghamri-Doudane took a different angle altogether, train only on normal traffic and flag whatever deviates from it. Their strongest model reaches a 100% detection rate on packet-based traffic with a false positive rate of just 1.53%, and the one-class setup is attractive wherever labeled attack data is scarce. Our setting is not that setting.

CICIDS2017 comes with labeled attacks, so we train a supervised model instead and spend the extra design effort on sequence modeling and attention, letting the classifier learn from runs of records rather than single ones.

E. Summary of Related Work

Two gaps run through this body of work. Classifiers, including many deep ones, still mostly judge individual records, which handicaps them against attacks that ramp up gradually. And attention, for all its dominance in language tasks, remains uncommon in intrusion detection research. The model described in the rest of this paper was built around both observations, and Section IV explains exactly how.

III. SYSTEM ARCHITECTURE

Fig. 1 lays out the pipeline from raw traffic to a final verdict. The system does three jobs, it prepares the data, it cuts the data into small time windows and runs them through the model, and it converts the model's output into a plain answer, benign traffic or attack traffic.

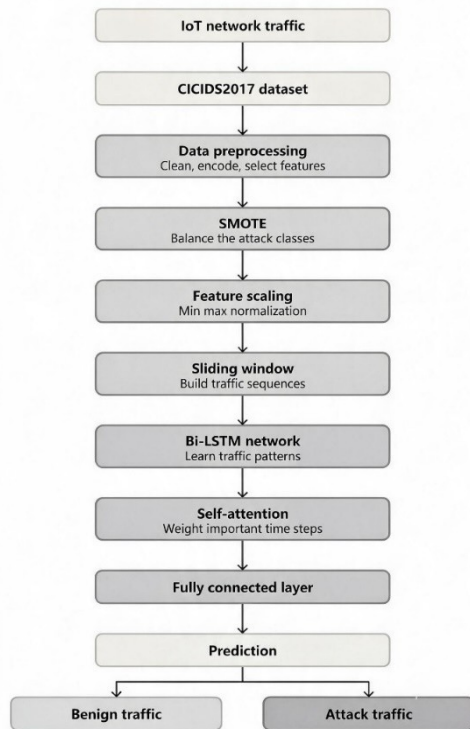


Figure 1: End-to-end Detection Pipeline, from Raw IoT Network Traffic to the Final Benign/Attack Prediction.

The raw material is the CICIDS2017 dataset, real network traffic with several kinds of attacks mixed in. Preprocessing removes broken and missing values and encodes the features into a form the model can read. Attack traffic is far rarer than normal traffic, so during training SMOTE generates extra attack examples; without this step the model could score well by predicting “normal” almost blindly. Min-max scaling then squeezes every feature into the same 0-to-1 range, so a feature with large raw numbers cannot dominate the others just by being large.

Windowing comes next. Rather than handing the model single records, we group ten consecutive records into a window and slide that window forward one record at a time across the whole dataset. This is

the design choice that gives the model a short memory of recent traffic history, which is the ingredient missing from most of the work reviewed in Section II.

Each window then flows through the BiLSTM network, the self-attention layer, and finally a small fully connected head that produces a single prediction.

Section IV walks through those three pieces in detail.

METHODOLOGY

A. Building the Traffic Windows

Flow records from CICIDS2017 are cleaned first: rows with missing or corrupt values are dropped, and so are columns that are constant or near-duplicates of each other, since they carry no information. Min-max scaling is fitted on the training portion only, which stops statistics from later traffic leaking backwards into training. The data keeps its original time order and is cut into overlapping ten-record windows, advancing one record per step. A window's label, attack or benign, comes from a simple majority vote over the records inside it. The model therefore judges a short stretch of traffic as one unit instead of judging a record in isolation.

B. Balancing the Data

The rarity of attacks is good news for a network operator and bad news for training. A model can post high raw accuracy on such data by answering “normal” nearly every time, and it will learn to do exactly that if nothing stops it. SMOTE stops it, building synthetic attack samples by interpolating between real attack samples that sit close together in feature space. The synthesis touches training data only. Validation and test sets keep their natural imbalance, because the whole point of testing is to see how the model behaves in the world it will actually face.

C. The BiLSTM and Attention Layer

Each window, of size $T = 10$ time steps by N features, passes through two stacked Bidirectional LSTM layers with 128 units in the first layer and 64 in the second. The LSTM cell at each time step t computes the following gated updates, where σ denotes the sigmoid function and $\odot$ the Hadamard (element-wise) product:

Forget gate:

$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \quad (1)$$

Input gate:

$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)$	(2)
---	-----

Cell candidate:

$\tilde{c}_t = \tanh(W_c[h_{t-1}; x_t] + b_c)$	(3)
--	-----

Cell state:

$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$	(4)
---	-----

Output gate:

$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o)$	(5)
---	-----

Hidden state:

$h_t = o_t \odot \tanh(c_t)$	(6)
------------------------------	-----

In the equations above, $W_\bullet$ and $b_\bullet$ are learnable weight matrices and bias vectors respectively. The Bidirectional LSTM reads the same sequence in both the forward and backward directions, producing two hidden-state sequences that are concatenated at each step:

$\vec{h}_t = \text{LSTM}_f(x_t, \vec{h}_{t-1}), \quad \overleftarrow{h}_t = \text{LSTM}_b(x_t, \overleftarrow{h}_{t+1})$	(7)
--	-----

$h_t = [\vec{h}_t \parallel \overleftarrow{h}_t]$	(8)
---	-----

A self-attention layer sits on top of the BiLSTM. It assigns a scalar alignment score e_t to each time step using a small learned function parameterized by weight matrix W_a , bias b_a and context vector v :

$e_t = v^T \tanh(W_a h_t + b_a)$	(9)
----------------------------------	-----

The scores are normalized into attention weights α_t using a softmax so that they sum to one:

$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^T \exp(e_k)}$	(10)
---	------

A single context vector c is formed as a weighted sum of all hidden states, giving more influence to the time steps the scoring function rated highly:

$c = \sum_{t=1}^T \alpha_t h_t$	(11)
---------------------------------	------

The context vector is passed through a dense output layer with sigmoid activation to produce the attack probability for the window:

$\hat{y} = \sigma(W_{out} c + b_{out})$	(12)
---	------

The practical effect is easy to picture. Suppose nine records in a window look normal and one looks like the opening of a port scan. The scoring function pushes most of the weight α_t onto the odd record, so the context vector c is dominated by it. The stored weights can be read back afterwards to see which record drove the decision.

D. Picking the Decision Threshold

The network outputs a probability, not a verdict. The naive move is to call anything above 0.5 an attack, but when positives are rare that cutoff is rarely the best one. We sweep candidate cutoffs on a held-out validation set instead and keep whichever gives the best F1-score. On this run the search landed on 0.3805.

RESULT

A. Dataset and Setup

All experiments use CICIDS2017, which mixes benign traffic with DoS, DDoS, port scans, brute force attempts, botnet activity, infiltration, and web attacks. We collapse the labels into a binary problem, benign versus attack, and split the cleaned data 70/15/15 into training, validation, and test while preserving row order, so test traffic always comes after training traffic in time. Training ran with a batch size of 256 and the Adam optimizer at an initial learning rate of 0.001, with early stopping once validation loss had stalled for 10 straight epochs. The test split produced 29,991 windows, and only 452 of them, roughly one and a half percent, are attack windows. Real traffic looks like this, and it makes the evaluation unforgiving: a constant “normal” guess is already right about 98.5% of the time.

RESULTS

Table I lists the test-set numbers at the tuned cutoff of 0.3805, and Fig. 2 charts the same values. Accuracy and AUC-ROC are both very high, while precision, recall, and F1-score for the attack class land in the 70 to 78 percent band.

BiLSTM + attention model

Table I: Test-set Performance of the Proposed

Metric	Value
Accuracy	99.18%
Precision (ATTACK)	70.71%
Recall (ATTACK)	77.43%
F1-score (ATTACK)	0.739

Metric	Value
AUC-ROC	0.995
False Positive Rate	0.49%
False Negative Rate	22.57%

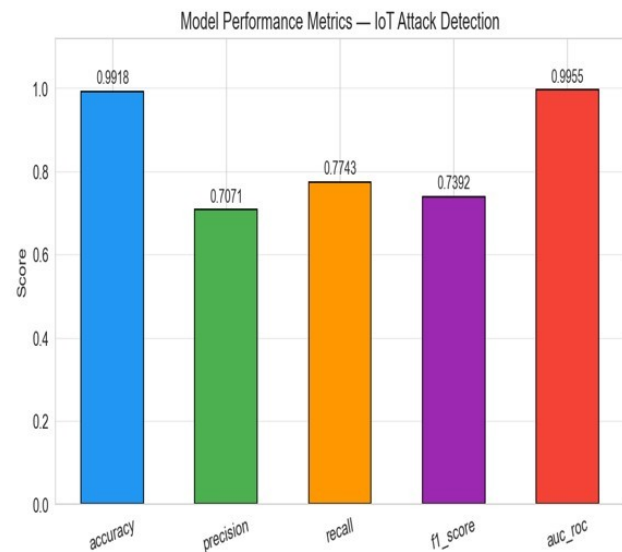


Figure 2: Model Performance Metrics on the Held-Out Test Set.

C. Confusion Matrix Analysis

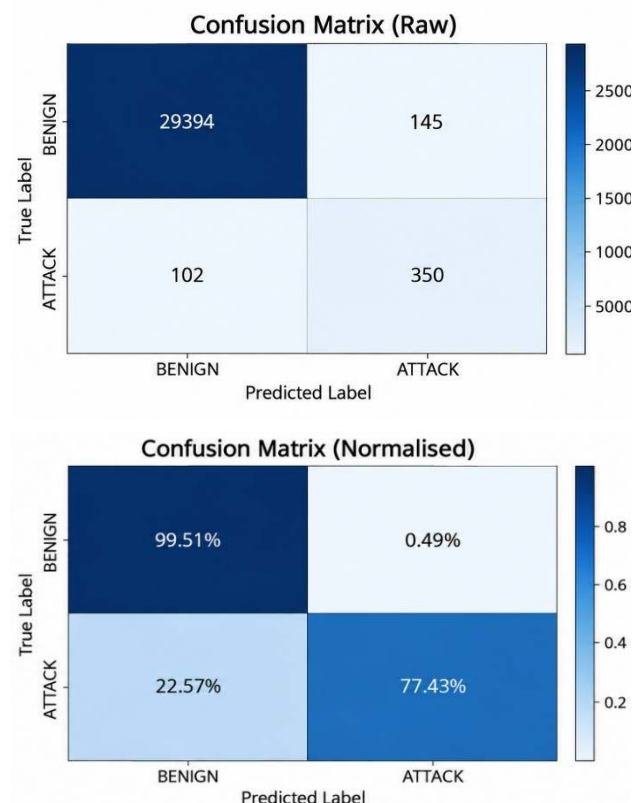


Figure 3: Confusion Matrix on the Held-Out Test Set.

Fig. 3 gives the raw counts. Of 452 real attack windows, the model caught 350 and missed 102. Of

29,539 normal windows, it wrongly flagged 145. The 102 misses are the figure to worry about in a security setting, since an undetected attack can do real damage while a false alarm costs an analyst a few minutes of checking. The false alarms are not free either. A false positive rate of 0.49% sounds tiny until it is multiplied across nearly thirty thousand windows, at which point it becomes 145 separate alerts that somebody has to sit down and review.

D. ROC Curve Analysis

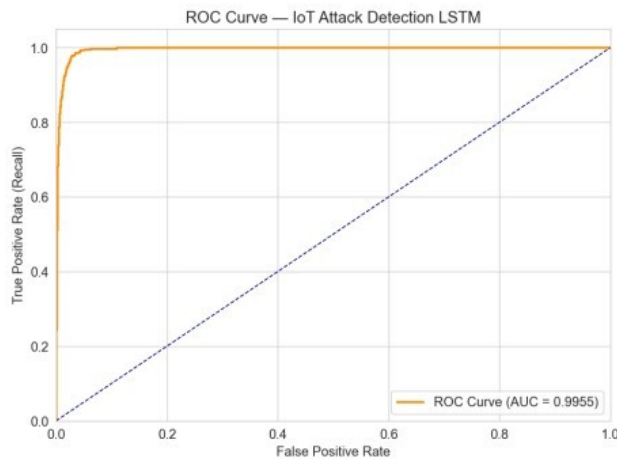


Figure 4: ROC Curve on the Test Set (AUC = 0.995).

The AUC-ROC of 0.995 says the model ranks attack windows above normal ones almost every time: draw one of each at random and the attack window gets the higher score in about 99.5% of draws, across every cutoff we could have chosen. Turning that ranking skill into one sharp yes-or-no answer is where performance drops, and the gap between the two is where future effort will pay off most.

E. Comparison with Related Work

It helps to place these numbers beside the related work from Section II. The one-class models of Amara Korba et al., trained on normal traffic alone, report a packet-based AUC of 99.23% and a bidirectional-flow AUC of 92.23% on the IoT-23 dataset. Our AUC-ROC of 0.995 sits near their packet figure and above their flow figure, and it was obtained under a time-ordered split, normally a tougher test than the cross-validation they used. Precision and recall cannot be compared head to head, though. The datasets differ, CICIDS2017 against IoT-23, the attack mixes differ, and the learning problems differ, since their models never see a labeled attack while ours trains on them directly. The narrower takeaway still stands: on

ranking ability, a supervised sequence-plus-attention model holds its own, which supports reaching for sequence models whenever labeled attack data exists.

DISCUSSION

The gap between the strong AUC-ROC and the more modest F1-score is the most important thing in these results. A high AUC means the model has genuinely learned to separate attack traffic from normal traffic. A modest F1 at one fixed cutoff means that separation is not yet clean enough to trust an automatic yes-or-no answer without a person checking. Several things plausibly feed the gap. The test set holds very few attack windows, so every single error moves precision and recall by a visible amount. The test data also comes strictly after the training data, and traffic patterns drift, so a threshold tuned on earlier traffic will sit slightly wrong on later traffic. And the majority vote labeling means a window straddling the border between normal and attack behaviour is genuinely ambiguous; some of these windows would be hard for a human labeler too, let alone a model.

G. Ethical Considerations

A tool like this belongs beside a human analyst, not in place of one. With 102 missed attacks and 145 false alarms across roughly 30,000 test windows, the model is not yet trustworthy enough to block traffic on its own in any setting where a missed attack carries real consequences. This study also used a public research dataset rather than traffic collected from real homes or devices, which sidesteps privacy problems here. Anyone applying the approach to genuine IoT traffic would need to settle how that traffic is collected, stored, and anonymized before fitting the first model.

CONCLUSION

This paper presented a Bidirectional LSTM with attention for early detection of cyber attacks in IoT network traffic. Grouping consecutive traffic records into short windows, and letting the model learn which records inside each window matter most, reaches 99.18% accuracy and an AUC-ROC of 0.995 on a realistic, time-ordered test split of CICIDS2017. The precision and recall numbers show real room left to improve. Rather than waving at future improvements in the abstract, we have named the specific gap, ranking ability versus fixed-threshold decisions, and the concrete steps we intend to take at it.

VII. FUTURE WORK

Several next steps are already queued. An ablation comparing the model with and without the attention layer would show what attention actually buys. Testing on IoT-specific datasets beyond CICIDS2017 would show whether the results travel. A loss function built for imbalanced data could reduce the reliance on SMOTE, and varying the window size would reveal how sensitive the results are to the choice of ten. Extending the output from a plain attack flag to the name of the specific attack type is a natural upgrade. Finally, real IoT traffic keeps changing, and a model trained once will slowly go stale; periodically re-tuning the decision cutoff on fresh traffic looks like a cheap and promising way to keep pace with that drift.

VIII. CODE AVAILABILITY

The dataset utilized in this research is publicly accessible through its respective online repository. To ensure reproducibility and facilitate further research, the complete implementation of the proposed AI-based IoT cyber-attack detection model is made available through a public GitHub repository.

Dataset:<https://www.unb.ca/cic/datasets/ids-2017.html>

SourceCode:<https://github.com/SangitaSriHarshitha/AI-Model-for-Early-Detection-of-Cyber-Attacks-in-IoT-Networks-.git>

REFERENCES

1. A. Amara Korba, A. Diaf, and Y. Ghamri-Doudane, "AI-Driven Fast and Early Detection of IoT Botnet Threats: A Comprehensive Network Traffic Analysis Approach," *arXiv:2407.15688*, Jul. 2024. doi:10.1109/IWCMC61514.2024.10592525
2. S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," version 1.0.0, Zenodo, Jan. 2020. doi:10.5281/zenodo.4743746
3. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116. doi:10.5220/0006639801080116
4. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. doi:10.1613/jair.953
5. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735
6. A. Graves and J. Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, 2005. doi:10.1016/j.neunet.2005.06.042
7. D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015. doi:10.48550/arXiv.1409.0473
8. A. Vaswani et al., "Attention Is All You Need," in *Proc. 31st Conf. Neural Information Processing Systems (NeurIPS)*, 2017. doi:10.48550/arXiv.1706.03762
9. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. doi:10.1023/A:1010933404324
10. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794. doi:10.1145/2939672.2939785
11. M. Antonakakis et al., "Understanding the Mirai Botnet," in *Proc. 26th USENIX Security Symp.*, 2017, pp. 1093–1110
12. C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet Detection in the Internet of Things Using Deep Learning Approaches," in *Proc. 2018 Int. Joint Conf. Neural Networks (IJCNN)*, Jul. 2018, pp. 1– 8. doi:10.1109/IJCNN.2018.8489489
13. C. Wei, G. Xie, and Z. Diao, "A Lightweight Deep Learning Framework for Botnet Detecting at the IoT Edge," *Computers & Security*, 2023, Art. no. 103195. doi:10.1016/j.cose.2023.103195

14. G. L. Nguyen, B. Dumba, Q.-D. Ngo, H.-V. Le, and T. N. Nguyen, "A Collaborative Approach to Early Detection of IoT Botnet," *Computers & Electrical Engineering*, vol. 97, 2022, Art. no. 107525. doi:10.1016/j.compeleceng.2021.107525
15. T. N. Nguyen, Q.-D. Ngo, H.-T. Nguyen, and G. L. Nguyen, "An Advanced Computing Approach for IoT-Botnet Detection in Industrial Internet of Things," *IEEE Trans. Industrial Informatics*, vol. 18, no. 11, pp. 8298–8306, 2022. doi:10.1109/TII.2022.3152814
16. B. Bojarajulu, S. Tanwar, and T. P. Singh, "Intelligent IoT-Botnet Attack Detection Model with Optimized Hybrid Classification Model," *Computers & Security*, vol. 126, 2023, Art. no. 103064. doi:10.1016/j.cose.2022.103064
17. N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019. doi:10.1016/j.future.2019.05.041
18. R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *Proc. IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35. doi:10.1109/SPW.2018.00013

HOW TO CITE: Sangita Sri Harshitha, Dr. D. Latha, AI Model for Early Detection of Cyber Attacks in IoT, *Int. J. Sci. R. Tech.*, 2026, 3 (10), 881-888. <https://doi.org/10.5281/zenodo.23391650>