Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.075098

ARTICLE

Bridging AI and Cyber Defense: A Stacked Ensemble Deep Learning Model with Explainable Insights

Faisal Albalwy[1,*] **and Muhannad Almohaimeed**[2]

[1]Department of Cybersecurity, College of Computer Science and Engineering, Taibah University, Madinah, 42353, Saudi Arabia
[2]Department of Information Systems, College of Computer Science and Engineering, Taibah University, Madinah, 42353, Saudi Arabia

*Corresponding Author: Faisal Albalwy. Email: fbalwy@taibahu.edu.sa

Received: 24 October 2025; Accepted: 17 December 2025; Published: 12 March 2026

ABSTRACT: Intrusion detection in Internet of Things (IoT) environments presents challenges due to heterogeneous devices, diverse attack vectors, and highly imbalanced datasets. Existing research on the ToN-IoT dataset has largely emphasized binary classification and single-model pipelines, which often show strong performance but limited generalizability, probabilistic reliability, and operational interpretability. This study proposes a stacked ensemble deep learning framework that integrates random forest, extreme gradient boosting, and a deep neural network as base learners, with CatBoost as the meta-learner. On the ToN-IoT Linux process dataset, the model achieved near-perfect discrimination (macro area under the curve = 0.998), robust calibration, and superior F1-scores compared with standalone classifiers. Interpretability was achieved through SHapley Additive exPlanations–based feature attribution, which highlights actionable drivers of malicious behavior, such as command-line patterns, process scheduling anomalies, and CPU usage spikes, and aligns these indicators with MITRE ATT&CK tactics and techniques. Complementary analyses, including cumulative lift and sensitivity-specificity trade-offs, revealed the framework's suitability for deployment in security operations centers, where calibrated risk scores, transparent explanations, and resource-aware triage are essential. These contributions bridge methodological rigor in artificial intelligence/machine learning with operational priorities in cybersecurity, delivering a scalable and explainable intrusion detection system suitable for real-world deployment in IoT environments.

KEYWORDS: Cybersecurity; IoT intrusion detection; stacked ensemble learning; deep learning; explainable AI (XAI); probability calibration; SHAP interpretability; ToN-IoT dataset; MITRE ATT&CK

1 Introduction

The rapid proliferation of Internet of Things (IoT) deployments across critical infra-structures has expanded the cyberattack surface and introduced heterogeneous, high velocity telemetry that challenges conventional intrusion detection systems (IDSs) [1–3]. Signature-based and rule-centric approaches struggle to keep pace with evolving tactics, techniques, and procedures (TTPs), and often fail to generalize across diverse device types and operational contexts [4,5]. Consequently, there is growing interest in machine learning–driven IDSs that can learn discriminative patterns from rich host and network traces, and provide timely, risk-aware decisions suitable for security operations centers (SOCs) [1,5–8].

Within this landscape, the ToN-IoT corpus has emerged as a commonly used benchmark for IoT cyber analytics, providing multimodal artifacts (e.g., host processes, network flows, and sensor signals) that enable both host- and network-centric detection studies [9–11]. In the present work, we focused on the

Linux process scheduling dataset, which comprise 503,019 process-level records with 16 telemetry fields (e.g., CMD, PID, CPU, Status/State, and PRI/NICE), capturing both normal and multiple attack categories (distributed denial of service [DDoS], password-based DoS [DoS-Password], injection, scanning, and cross-site scripting [XSS]). This dataset provides a tractable but expressive view of host behavior for learning attack-relevant signatures.

Despite the steady research progress, three limitations recur in the literature. First, many recent ToN-IoT IDS studies train and evaluate their models mainly in a binary normal-versus-attack setting, which can overstate performance because misclassification between specific attack types is hidden when all attacks are merged into a single class [12,13].

Second, recent ToN-IoT IDS studies typically report only classification metrics such as accuracy, precision, recall and F1-score and do not evaluate how well their predicted probabilities are calibrated [12–14]. Yet in security-critical intrusion-detection deployments, mis-calibrated probabilities can lead to poor threshold choices and unstable alert volumes, and both the general ML literature and IDS-specific work show that modern classifiers often produce over-confident, poorly calibrated scores unless explicit calibration is applied [15,16].

Third, explainability in intrusion detection is still predominantly addressed via post-hoc, local explanation methods (e.g., LIME/SHAP), and the X-IDS literature notes that black-box decisions and non-stakeholder-aligned explanations limit analyst trust and operational actionability; consequently, feature attributions are often not explicitly mapped to threat-grounded host artifacts such as commands (CMD), process identifiers (PID), process/thread state, and resource-usage footprints (CPU/memory/disk) [17–19].

A synthesis of recent ToN-IoT studies showed that these gaps hinder methodological comparability and hamper deployment-oriented decision-making.

To address these issues, we propose a stacking ensemble deep learning model for multiclass intrusion detection on ToN-IoT Linux processes. The framework integrates complementary base learners, namely random forest (RF), XGBoost, and a deep neural network (DNN), with a CatBoost meta-learner trained on out-of-fold (OOF) predictions to reduce variance and improve generalization. We further employed SHapley Additive exPlanations (SHAP) to produce threat-grounded explanations that highlight operation-ally meaningful drivers (e.g., CMD, PID, CPU, Status/State, and temporal regularities).

This study makes four contributions to IoT intrusion detection:

- Propose a calibrated stacking ensemble deep learning framework that integrates tree-based and neural learners for multiclass ToN-IoT detection, trained using stratified OOF stacking to limit leakage and reduce variance.
- Provide a comprehensive assessment beyond accuracy/F1, including ROC-AUC, calibration, sensitivity–specificity (Youden), and cumulative lift, to support risk-aware thresholding and workload planning.
- Employ SHAP-based global and local attributions to reveal actionable host-level indicators (e.g., command-line tokens, process churn, CPU anomalies), thereby facilitating SOC triage, justification, and rule hardening.
- Contextualize our framework within recent ToN-IoT studies by synthesizing their advances in multiclass robustness, calibration, and interpretability. Unlike prior work, we explicitly demonstrate how our stacked ensemble deep learning approach addresses these gaps, thereby bridging the divide between re-search-grade IDS prototypes and deployment-ready analytics for SOC environments.

The remainder of the paper is organized as follows: Section 2 reviews related work on ToN-IoT IDSs and positions our approach. Section 3 details the dataset, preprocessing, and model design, including stacking and calibration. Section 4 reports comprehensive results and interpretability analyses. Section 5 discusses operational implications and threat-informed interpretations. Section 6 presents the study limitations, future directions, and conclusions

2 Related Works

The growing body of research on intrusion detection using the ToN-IoT dataset has produced a range of methodological perspectives. This section reviews the most relevant contributions, highlighting their key achievements and limitations, and situates our work within this evolving landscape.

2.1 Binary Classification Approaches

Early research on the ToN-IoT dataset focused predominantly on binary classification, establishing performance baselines for distinguishing normal traffic from malicious traffic. Gad et al. [20] developed a distributed intrusion detection framework across cloud-fog-edge architectures, achieving an accuracy higher than 99% with XGBoost in binary tasks. Their comprehensive evaluation across multiple deployment layers pro-vides valuable insights into computational trade-offs, although the reliance on traditional metrics without probabilistic assessment limits operational applicability.

Similar binary-focused approaches by Naeem et al. [21] and Kaddour et al. [22] achieved accuracy rates exceeding 95%, demonstrating the feasibility of basic attack detection. However, these studies share common limitations: simplified feature sets that may miss complex attack signatures; absence of cross-validation procedures, raising overfitting concerns; and limited exploration of decision thresholds critical for operational deployment. While binary classification provides an important foundation, oversimplification of the problem space constrains practical applicability, where diverse attack types require distinct response strategies.

2.2 Multiclass Detection Challenges

The transition from binary to multiclass classification revealed significant performance gaps in the existing approaches. Oseni et al. [23] explicitly demonstrated this challenge, observing accuracy degradation from 99.15% in binary settings to 90.55% in multiclass settings. This substantial decrease highlights the increased complexity of distinguishing between diverse attack categories—a critical requirement for actionable threat intelligence. While establishing important benchmarks, their work relied primarily on traditional machine learning algorithms without exploring more advanced strategies that might mitigate class-specific weaknesses.

Cao et al. [24] advanced multiclass detection using both conventional and deep learning methods, achieving competitive performance on the ToN-IoT dataset. However, when applied to alternative datasets, their models showed significant performance degradation, which suggests potential overfitting to dataset-specific characteristics. This generalization challenge, combined with the absence of calibration mechanisms, limits the confidence in deployment scenarios in which data distributions may shift.

Moustafa et al. [19] provide a detailed class-wise analysis that revealed extreme performance disparities: Some classes achieved near-perfect detection, whereas others experienced a decrease to 20%–60%, yielding a weighted true-positive rate lower than 70%. These results underscore the heterogeneity of attack patterns in IoT environments and the inadequacy of single-model approaches for capturing diverse threat behaviors. The contribution of this study lies in exposing these disparities, although it stopped short in proposing ensemble or hierarchical strategies to address them.

2.3 Deep Learning and Ensemble Advances

Recent work has investigated more sophisticated architectures to address the limitations of traditional approaches. Shaker and Al-Musawi [25] investigated deep learning models on the NF-ToN-IoT variant, achieving 96.37% accuracy on specific datasets. However, performance varied dramatically across datasets (decreasing to 68%–83%), raising questions about model stability and the role of dataset characteristics in determining effectiveness. Their exclusive focus on single deep learning architectures without any model integration represents a missed opportunity for variance reduction.

A notable advancement came from Sneha and Prasad [26], who introduced an explainable model combining convolutional neural networks (CNN), artificial neural networks, and least absolute shrinkage and selection operator with local interpretable model-agnostic explanations interpretability. Achieving 98.7% accuracy, with F1-scores exceeding 98%, their work demonstrates the potential of such methods while addressing the critical need for model interpretability. The integration of threshold calibration to reduce false positives shows that operational awareness is often lacking in purely academic studies. Nevertheless, their reliance on hard voting rather than meta-learning and their focus on local rather than global interpretability leaves room for methodological refinement.

Eren and Küçük [27] contributed automated feature generation techniques, reporting improved F1-scores and AUC metrics. While their feature engineering approach provides valuable preprocessing insights, unclear reporting regarding binary compared with multiclass results and the absence of overall accuracy metrics complicate direct comparisons with other studies. This methodological ambiguity is symptomatic of broader reproducibility challenges in the field.

2.4 Research Gaps and Positioning

Analysis of the existing literature revealed consistent methodological gaps that our work addressed. First, most studies rely on single-model architectures or basic ensembles without exploring sophisticated meta-learning strategies. Second, evaluation typically focuses on raw accuracy metrics without considering probabilistic reliability through calibration analysis, which is critical for risk-based security decisions. Third, interpretability remains either absent or limited to local explanations, missing opportunities for actionable security insights through global feature importance analysis.

Table 1 summarizes the comparative landscape, highlighting how prior works, while making important contributions, exhibit limitations in handling multiclass complexity, ensuring probabilistic reliability, and providing interpretable insights.

The evolution of ToN-IoT intrusion detection research reveals a trajectory from simple binary classification toward sophisticated multiclass approaches with interpretability requirements. While early work established feasibility baselines, recent studies have increasingly recognized the need for more advanced methods, probabilistic calibration, and explainable predictions. Our stacking ensemble deep learning framework synthesizes these advances while addressing persistent gaps, particularly the integration of meta-learning for improved generalization, calibration for risk-based decision-making, and global interpretability for operational insights. This comprehensive approach bridges the gap between academic performance metrics and deployment-ready security solutions, advancing both the methodological sophistication and practical applicability of IoT IDSs.

Table 1: Comparison of existing intrusion detection approaches on the ToN-IoT dataset

Work	Year	Method type	Key strengths	Primary limitations
Gad et al. [20]	2022	Single models (XGBoost and RF)	99% binary accuracy and distributed architecture	Limited multiclass performance and no calibration
Naeem et al. [21]	2022	Binary classifiers	High binary accuracy	Binary-only and limited features
Kaddour et al. [22]	2022	Simple ML	Strong binary results	No validation procedures
Oseni et al. [23]	2021	Simple ML	Explicit binary/multi comparison	Significant multiclass degradation
Cao et al. [24]	2023	Simple ML + DL	Strong ToN-IoT performance	Poor cross-dataset generalization
Moustafa et al. [19]	2022	Deep learning (CNN/RNN)	Detailed class-wise analysis	Poor accuracy for some classes, TPR < 70%
Shaker et al. [25]	2022	Single deep learning	96.37% on a specific dataset	High variance across datasets
Sneha and Prasad [26]	2024	CNN + ANN + LASSO	98.7% accuracy and interpretability	Hard voting and local explanations only
Eren and Küçük [27]	2023	Automated features	Feature engineering insights	Unclear evaluation metrics

3 Materials and Methods

3.1 Study Design and Data Source

This study used the Linux process scheduling dataset from the ToN-IoT project, a comprehensive repository of telemetry data collected from IoT devices. The raw cleaned dataset originally contained 503,019 instances across 16 features, capturing both normal system behavior and various types of cyberattacks, including normal, DDoS, DoS, Password, injection, scanning, and XSS. The dataset features comprise ts (data collection timestamp), PID (Process Identifier), TRUN (Number of threads in state 'running'), TSLPI (Number of interruptible sleeping threads), TSLPU (Number of uninterruptible sleeping threads), POLI (Process scheduling policy), NICE (Process nice value), PRI (Numerical process priority), RTPR (Real-time process priority), CPUNR (CPU core number where the process is executing), Status (Process status indicator for newly started processes), EXC (Process exit code or fatal signal), State (Current process state), CPU (CPU time consumption in system mode), CMD (Process command name), and type (Attack type label). These features, collected using Linux system monitoring tools, provide a detailed view of process behavior and system resource utilization, supporting the development and evaluation of machine learning models for intrusion detection and attack classification.

3.2 Data Preparation

The raw dataset underwent extensive preprocessing to ensure quality and balance. The 'Type' column, representing attack categories, was encoded using LabelEncoder for numerical processing. Feature normalization was performed using StandardScaler to standardize data distributions. The dataset was split using stratified sampling into 70% training and 30% testing sets, ensuring that the attack distribution remained consistent across splits.

To address class imbalance, SMOTE (Synthetic Minority Oversampling Technique) was applied, generating synthetic samples for minority classes only on the training data. Specifically, for each fold of the stratified five-fold cross-validation, SMOTE was applied to the training portion of that fold after the train–test split and before fitting the base learners, while the corresponding validation fold remained unmodified. Similarly, for the final evaluation, SMOTE was applied to the training partition (70%) only, and the held-out test set (30%) was kept in its original, imbalanced form. This strategy avoids information leakage and reduces overfitting caused by oversampling the minority classes in the evaluation data.

All 16 features were retained for model training, as preliminary analysis indicated that each contributed relevant information for distinguishing between attack types. Feature scaling and normalization were applied to improve model convergence and performance. Dimensionality reduction was not explicitly required, given the moderate number of features, but preprocessing ensured that the dataset was optimized for both tree-based models and neural networks.

3.3 Model Development

This study employed both ensemble learning methods and deep learning architectures to build predictive models for intrusion detection. Initially, individual classifiers were trained and evaluated to establish baseline performance. The algorithms included the following:

- Ensemble methods: RF, extreme gradient boosting (XGBoost), and CatBoost
- Deep learning methods: CNN, DNN, TabNet, and multilayer perceptron (MLP)

Each model was implemented using the *scikit-learn*, *xgboost*, *catboost*, and *pytorch* frameworks in Python. The deep learning models were trained with standardized hyperparameters, including ReLU activations, dropout regularization, the Adam optimizer, and categorical cross-entropy loss, while ensemble methods were trained using their standard multiclass configurations. Specifically, DNN involved an input layer with all features, followed by three fully connected hidden layers. Each hidden layer used ReLU activation, and a dropout layer with a rate of 0.3 was applied after the first hidden layer to mitigate overfitting while keeping the model lightweight. The DNN model was trained with the Adam optimizer (learning rate = 0.001), a batch size of 32, and up to 20 epochs, using 10% of the training data for validation.

The performance of these algorithms was individually assessed in terms of accuracy, precision, recall, and F1-scores. The results showed that the top four learners, namely RF, XGBoost (XGB), CatBoost, and DNN, were selected for stacking.

The stacking strategy followed a two-step process with fivefold stratified cross-validation:

- Base Learners (Level 0 models): RF, XGBoost, and DNN were trained independently on each training fold, producing OOF predictions.
- Meta Learner (Level 1 model): A CatBoost classifier was trained on the concatenated OOF predictions to generate the final predictions. Test set predictions were averaged across folds to yield the final performance scores.

CatBoost was selected as the meta-learner for several reasons. First, it is a gradient-boosted decision tree algorithm that is known to produce well-calibrated probability estimates, which is essential in our deployment-oriented design where SOCs consume risk scores rather than hard labels [28]. Second, CatBoost naturally handles heterogeneous inputs and non-linear interactions, which is appropriate here because the meta-learner receives concatenated probability vectors from tree-based and neural base models [29]. For these reasons, CatBoost is an appropriate choice for aggregating the OOF predictions from RF, XGBoost, and DNN.

To make the training procedure fully transparent and facilitate replication, we summarize the complete stacking pipeline in Algorithm 1. The algorithm shows how the dataset is split into training and test partitions, how stratified K-fold cross-validation with SMOTE is used to generate out-of-fold (OOF) probability predictions from the base learners (RF, XGBoost, and DNN), and how these OOF predictions are used to train the CatBoost meta-learner. The final part of the algorithm describes the inference stage, where the retrained base learners and the CatBoost meta-learner are applied to new Linux process events to produce calibrated multiclass probability outputs.

Algorithm 1: Training and inference procedure of the stacking model for IDS

```
Input:
    D             ← full labeled dataset
    K = 5         ← number of CV folds
    BaseModels    ← {RF, XGBoost, DNN}
Output:
     Trained base models and CatBoost meta-learner
1:   Split D into D_train (70%) and D_test (30%) using stratified sampling
2:   Let C be number of classes
3:   Initialize OOF matrix Z of size |D_train| × (3C)
4:   Create K stratified folds {F1, . . ., FK} from D_train
5:   for each fold k do
6:        D_k ← training portion; F_val ← validation portion
7:        D_k* ← Apply SMOTE to D_k
8:        Train RF_k, XGB_k, DNN_k on D_k*
9:        for each sample x ∈ F_val do
10:           z(x) ← concat(RF_k.proba(x),
                            XGB_k.proba(x),
                            DNN_k.proba(x))
11:           Store z(x) in Z
12:   end for
13: Train CatBoost_meta on (Z, D_train.labels)
14: // Retrain base models on full training set
15: D_train* ← Apply SMOTE to D_train
16: Train RF_full, XGB_full, DNN_full on D_train*
17: function Predict(x_new):
18:       z_new ← concat(RF_full.proba(x_new),
                         XGB_full.proba(x_new),
                         DNN_full.proba(x_new))
19:       return CatBoost_meta.predict(z_new)
```

The integration of OOF predictions with stratified cross-validation provided two key benefits: It prevented information leakage into the meta learner while preserving the overall class distribution across folds, resulting in more reliable and unbiased performance estimates. This stacked ensemble deep learning approach leveraged the complementary strengths of tree-based ensemble methods and deep neural networks, while the CatBoost meta learner provided robust integration of predictions to enhance generalization in IoT intrusion detection.

3.4 Model Validation and Evaluation

The model performance was first evaluated for each individual classifier to establish baseline results. For both ensemble methods (RF, XGBoost, and CatBoost) and deep learning models (CNN, DNN, TabNet, and MLP), we computed standard classification metrics, including accuracy, precision, recall, and F1-scores, to compare their predictive capabilities across different attack types.

Following this, the ensemble deep learning stacking model, which comprised RF, XGBoost, and DNN as base learners and CatBoost as the meta-learner was validated using the held-out test set. Performance was assessed using the following measures:

- Overall metrics: accuracy, weighted-average precision, recall, and F1-score;
- ROC analysis: macro-averaged ROC curves and AUC values to evaluate overall discrimination capability;
- Calibration analysis: calibration curves to assess the alignment of predicted probabilities with observed outcomes;
- Cumulative lift analysis: lift curves to quantify the improvement of the stacking model over the random selection, benchmarked against the ideal "perfect" scenario;
- Sensitivity-specificity trade-off: evaluation of sensitivity and specificity across thresholds, with the optimal cutoff determined by Youden's J statistic;
- Statistical significance testing: paired Wilcoxon signed-rank tests applied to fold-wise macro-AUC values to evaluate whether the stacking model's improvements over baseline classifiers are statistically meaningful.

This multifaceted evaluation framework allowed for rigorous comparison between the individual models and the ensemble deep learning stacking model, demonstrating the benefits of combining complementary learners for enhanced generalization and robustness in IoT intrusion detection.

3.5 Feature Importance Assessment

The impacts of the individual features on the predictions of the stacking model was further assessed using SHAP. SHAP applies game-theoretic principles to quantify the contribution of each feature to the model output, assigning an optimal importance score based on cooperative interactions. Mean absolute SHAP values were computed across all samples and classes, allowing the ranking of features according to their influence on the CatBoost meta-learner. This analysis provided interpretable insight into the most critical factors that drive predictions of cyberattack types. The proposed model is illustrated in Fig. 1.

3.6 Hardware and Software Specifications

The study was conducted using Jupyter Notebook (Anaconda 3) with Python (version 3.9) on a MacBook Pro running the operating system Big Sur version 11.7.10, equipped with a 16 GB RAM and an Intel® Core™ i9 CPU @ 2.3 GHz. This hardware configuration was adequate for executing the full training and evaluation workflow, including cross-validation, model stacking, and post hoc analyses.

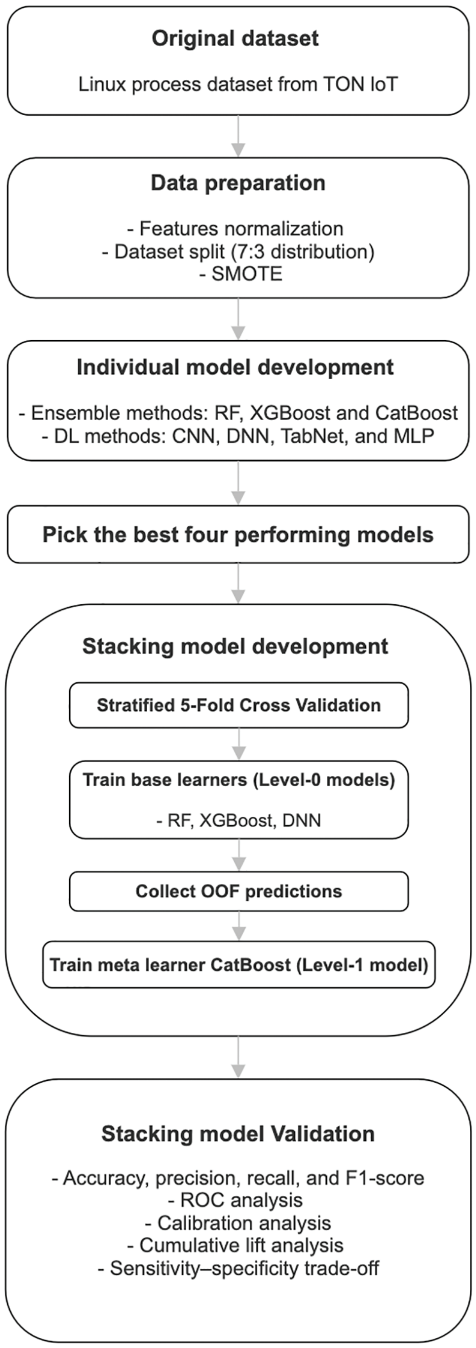

Original dataset
Linux process dataset from TON IoT
Data preparation
- Features normalization
- Dataset split (7:3 distribution)
- SMOTE
Individual model development
- Ensemble methods: RF, XGBoost and CatBoost
- DL methods: CNN, DNN, TabNet, and MLP
Pick the best four performing models
Stacking model development
Stratified 5-Fold Cross Validation
Train base learners (Level-0 models)
- RF, XGBoost, DNN
Collect OOF predictions
Train meta learner CatBoost (Level-1 model)
Stacking model Validation
- Accuracy, precision, recall, and F1-score
- ROC analysis
- Calibration analysis
- Cumulative lift analysis
- Sensitivity–specificity trade-off

Figure 1: Overview of the proposed stacking model methodology

4 Results

4.1 Baseline Data Assessment

Using the open-source Linux process scheduling dataset from the ToN-IoT repository, this study encompassed 503,019 instances with 16 features after preprocessing. Of these instances, 369,950 (73.5%) represented normal activity, whereas 133,065 (26.5%) were categorized as malicious, including attack types, such as DDoS, DoS-Password, injection, scanning, and XSS. The instances in the cohort were randomly stratified into a training set (n = 352,110) and a test set (n = 150,905), following a 7:3 distribution. The variable distributions between the two datasets exhibited no statistically significant disparities, confirming the representativeness of the test set.

4.2 *Algorithm Screening*

We initially evaluated a set of ensemble and deep learning classifiers individually to determine their predictive performances. The candidate algorithms included RF, XGBoost, CatBoost, CNN, DNN, TabNet, and MLP. Each model was trained and validated using stratified fivefold cross-validation, and performance was assessed in terms of accuracy, precision, recall, and F1-score. Table 2 presents the performance metrics for each classifier. Among the evaluated models, RF, XGBoost, CatBoost, and DNN demonstrated the best predictive performances and were selected for further exploration.

Table 2: Performance of individual classifiers

Performance → **Classifier ↓**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
RF	95.8	90.7	89.1	89.9
XGBoost	95.4	92.7	82.7	87.2
CatBoost	94.9	92.1	94.9	94.9
CNN	85.3	86.5	85.3	85.5
DNN	88.43	75	70	71
Tabnet	86	88	86	86
MLP	88	89.9	89	89

4.3 *Model Performance and Feature Importance*

Following the initial screening of the candidate algorithms, three high-performing classifiers, namely RF, XGBoost, and DNN, were selected as base learners, and CatBoost was chosen as the meta-learner for the ensemble deep learning stacking model. The key set of hyperparameter configurations for all base learners and the meta-learner is provided in Table 3.

The model was developed using a two-stage procedure. In the first stage, the base learners were independently trained on the training dataset using fivefold stratified cross-validation to ensure a balanced representation across attack categories. In the second stage, OOF predictions from these models were aggregated and provided as inputs to the CatBoost meta-learner, which generated the final predictions. This biphasic approach leveraged the complementary strengths of tree-based and neural models, enhancing the ability of the model to generalize beyond the training data.

Our ensemble deep learning stacking model demonstrated superior predictive capability compared with standalone classifiers, substantially outperforming them across multiple evaluation metrics. When applied to the test dataset, the stacking model achieved an overall accuracy of 97%, accompanied by a weighted average precision of 96%, recall of 96%, and F1-score of 97%, underscoring its robustness in balancing sensitivity and specificity. To better understand how the model behaves across different attack types, we computed per-class precision, recall, and F1-scores for the model on the test set. Table 4 reports these metrics for the seven classes (Normal, DDoS, DoS, Password, Injection, Scanning, and XSS). The model obtained consistently strong results across all categories, achieving near-perfect performance on high-support classes such as Normal (F1 = 0.98) and DDoS (F1 = 0.99), as well as on Password, Scanning, and XSS attacks (all with F1 = 1.00).

Table 3: Configurations for stacking framework

Model	Hyperparameters
Random Forest (Base Learner)	n_estimators = 50, criterion = gini, max_depth = None, min_samples_split = 2, min_samples_leaf = 1, bootstrap = True, n_jobs = 1, random_state = 42
XGBoost (Base Learner)	n_estimators = 50, tree_method = hist, eval_metric = mlogloss, learning_rate = 0.1, max_depth = 6, n_jobs = 1, random_state = 42
DNN Architecture (Base Learner)	Layer1: Dense(48, relu), Dropout(0.25) Layer2: Dense(24, relu) Output: Dense(num_classes, softmax) optimizer = Adam (lr = 0.001), loss = sparse_categorical_crossentropy epochs = 10, batch_size = 32, early_stopping_patience = 3
CatBoost (Meta Learner)	Iterations = 1000, learning_rate = 0.03, depth = 6, loss_function = MultiClass, verbose = 0, random_state = 42

Table 4: Per-class precision, recall, and F1-scores of the model on the test set. The model demonstrates strong and consistent performance across all seven classes, with near-perfect detection for high-support classes (Normal, DDoS, Password, Scanning, and XSS)

Class	Precision	Recall	F1-Score
Normal	0.98	0.97	0.98
DDoS	0.99	0.99	0.99
DoS	0.90	0.87	0.89
Password	1.00	1.00	1.00
Injection	0.87	0.91	0.89
Scanning	1.00	1.00	1.00
XSS	1.00	1.00	1.00

Slightly lower performance is observed for DoS (F1 = 0.89) and Injection attacks (F1 = 0.89), primarily due to their more subtle behavioral patterns in Linux process telemetry. Nonetheless, even for these more challenging classes, the model shows clear improvements over the best individual base learner (Table 2), demonstrating that meta-learning effectively mitigates class-specific weaknesses and enhances overall robustness.

To assess computational feasibility, we measured the training and inference times of all components in the stacking framework. Across five folds, Random Forest, XGBoost, and DNN base learners required an average of 215.38, 55.56, and 440.48 s per fold, respectively, with a total cross-validation time of 3785.73 s. The CatBoost meta-learner trained in 528.63 s. In contrast, inference was highly efficient: the model processed the test set in 6.93 s, corresponding to a per-sample latency of only 0.0134 ms. These results demonstrate that, while training is computationally intensive—as expected for a hybrid approach—the final model supports near real-time inference suitable for SOC environments.

To assess whether the improvements of the stacking model were statistically meaningful, we conducted Wilcoxon signed-rank tests using the fold-wise macro-AUC values from the five-fold cross-validation. The stacking model consistently achieved higher AUCs across all folds (mean = 0.9983) compared with the Random Forest (0.9946), XGBoost (0.9957), and DNN (0.9819) baselines. Because the stacking model outperformed each baseline on every fold, the Wilcoxon test statistic was 0.0 for all comparisons, yielding *p* = 0.0625. While this value lies slightly above the conventional 0.05 threshold due to the limited sample size (five folds), the results nonetheless indicate a clear and stable trend in favor of the stacking model.

Additional evaluations highlighted the reliability of the stacking ensemble deep learning model. The calibration curve in Fig. 2 shows close alignment between the predicted and observed probabilities, indicating that the model is well-calibrated across most of the probability range. For predicted probabilities between 0.2 and 0.8, the calibration curve remains very close to the diagonal, while a slight overconfidence is observed for probabilities above 0.9. This behavior is acceptable in our SOC-driven setting, where high scores are handled as high-risk alerts. The macro-averaged ROC curve in Fig. 3 further confirms the strong discriminative power of the model, with an AUC of 0.9983.

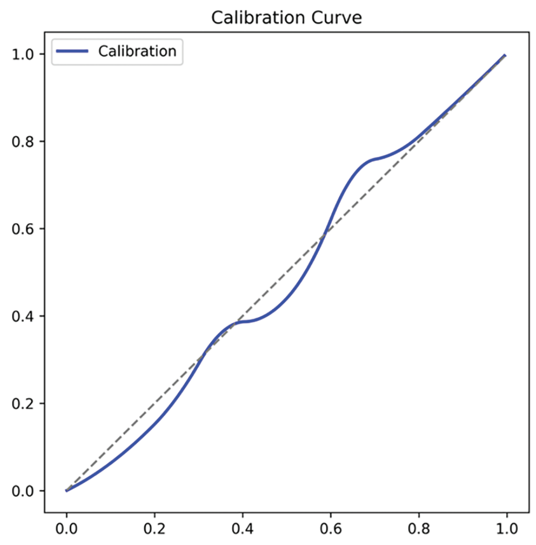


Figure 2: Calibration curve of the proposed stacking model. The curve demonstrates close alignment between the predicted and observed probabilities, indicating that the model produces well-calibrated outputs across the probability range

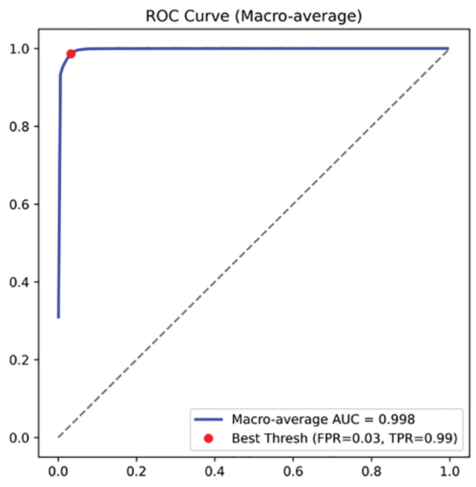


Figure 3: Macro-average ROC curve of the stacking model. The model achieved an area under the curve (AUC) of 0.9983, reflecting the near-perfect discrimination between the normal and attack classes

Analyses of sensitivity and specificity across the thresholds (Fig. 4) further revealed that the optimal operating point was obtained at a probability cutoff of 0.16, corresponding to a sensitivity of 0.994, specificity of 0.985, and Youden's J of 0.979. As illustrated in the plot, sensitivity remained consistently high across the range of thresholds, while specificity gradually increased with stricter cutoffs. The two curves converged near the optimal point, indicating a balanced trade-off between minimizing false negatives and false positives. This threshold achieved an excellent balance between true positive and true negative rates, reinforcing the robustness of the decision boundary of the model. Meanwhile, the cumulative lift curve presented in Fig. 5 demonstrates the advantage of the stacker model over the random selection, achieving a maximum lift of 7.0. The lift curve of the model was nearly indistinguishable from the perfect lift line, indicating that the stacker prioritized positive instances almost as effectively as the ideal classifier. This result highlights the exceptional discriminative ability and capacity of the model to identify a large proportion of attacks within a small fraction of the population. These findings confirm that the ensemble deep learning stacking model can achieve an excellent balance between true positive and true negative rates.

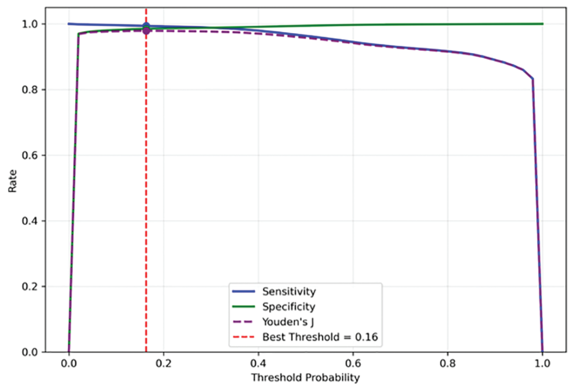


Figure 4: Sensitivity and specificity as functions of probability thresholds. The optimal cutoff was identified at 0.16, yielding a sensitivity of 0.994, specificity of 0.985, and a Youden's J of 0.979, highlighting a balanced trade-off between true positives and true negative rates

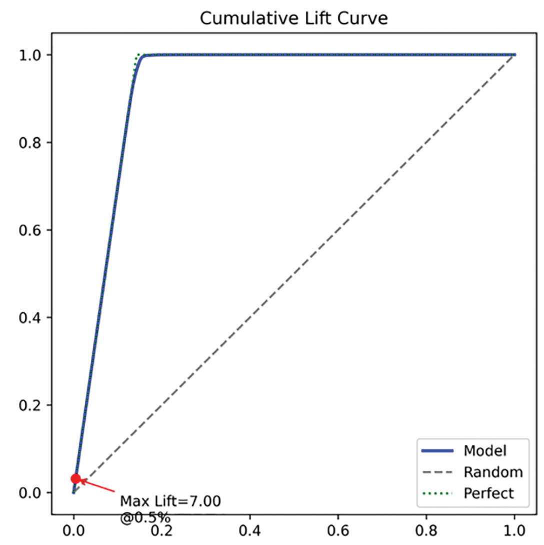
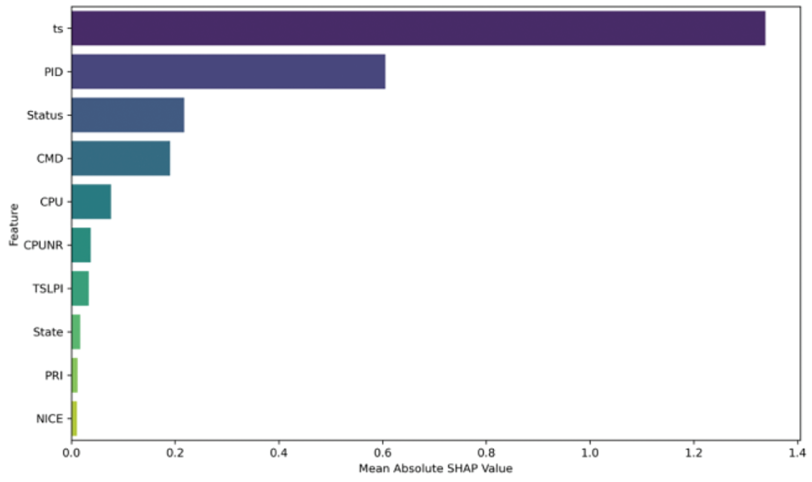
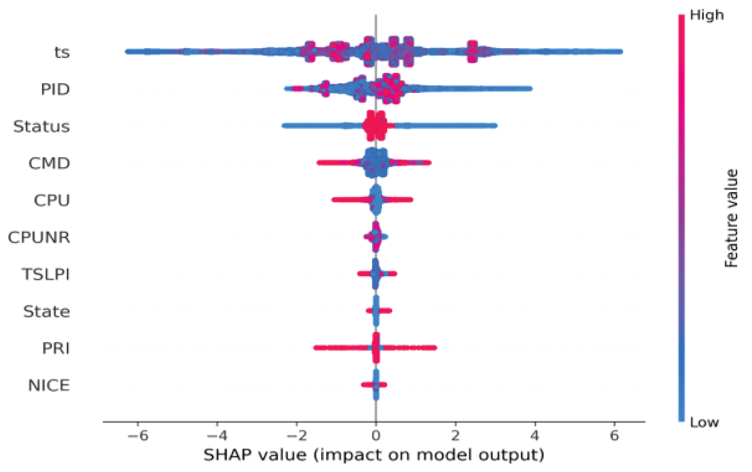


Figure 5: Cumulative lift curve. The stacking model exhibits a curve closely approaching the ideal "perfect" line, substantially outperforming random selection and achieving a maximum lift of 7.0 at the optimal threshold

SHAP values were computed to assess feature importance in the stacking model, as shown in Fig. 6. The top five influential features, namely ts, PID, status, CMD, and CPU, were ranked by according to their mean absolute SHAP values. These features likely capture both operational load and behavioral anomalies in the system, which are critical for distinguishing between normal activity and various types of attacks. The plot illustrates the magnitude and direction of the impact of each feature on the model output, with red indicating high feature values and blue indicating low feature values. Positive SHAP values increase predicted attack risk, while negative values indicate a favorable outcome.



(**A**) Feature importance ranking via SHAP



(**B**) SHAP summary plot of feature contributions

Figure 6: SHAP-based interpretability of the stacking ensemble deep learning: (**A**) Ranking of the top 10 most influential features based on mean absolute SHAP values. (**B**) Summary plot showing the distribution of SHAP contributions for each feature across the test cohort

5 Discussion

This study advances intrusion detection on the ToN IoT dataset by demonstrating a stacking ensemble deep learning that integrates high-performing base learners (RF, XGBoost, and a DNN) and targets multi-class discrimination with probabilistic reliability and explainability. The model delivers near-perfect discrimination (macro AUC ≈ 0.998) while maintaining well-calibrated posterior probabilities, which is crucial operationally because SOC playbooks consume probabilities—rather than raw labels—for triage and thresholding. Reliability diagrams, risk-sensitive threshold tuning, and the use of SHAP to expose decision drivers collectively move the contribution from "good ML" to deployable cyber defense, where decision quality, analyst trust, and governance needs are equally important [15,30–32].

A threat informed reading of the most predictive features provides external validity for the explanations. High SHAP attributions on the command field (CMD) are consistent with execution/defense evasion behaviors in MITRE ATT&CK T1059 (and the Unix shell subtechnique T1059.004) and align with recent CMD-based intrusion studies that document tokens, such as bash-c, shell metacharacters, and chained spawns, as high-value indicators [33–35]. Downloader patterns (e.g., curl, wget) match T1105 (ingress tool transfer) and have been empirically associated with the staging of payloads prior to persistence [36,37]. Temporal regularities in ts support interpretations around scheduled jobs (T1053.003, Cron) and mirror the beacon like polling found in APT tooling, where fixed or jittered inter-arrival times create detectable time series signatures [38,39]. Bursts of short lived process creation and process lineage reuse—captured via PID features—are compatible with scripted chains and fork bombs, such as local DoS, as discussed in host-based threat detection/audit provenance literature and operational guidance on mitigation (e.g., ulimit) [40,41]. Elevated or sustained CPU/CPUNR footprints reflect resource abuse signatures (e.g., cryptomining), corroborated by host-level detection studies relying on CPU/memory/execution profiles [42,43]. Finally, deviations in scheduling markers (PRI, NICE) and process-state distributions (Status, State) carry diagnostic value in Linux: Priority manipulation can throttle noisy tasks or accelerate time-critical payloads, while unusual accumulations of sleeping/defunct processes or persistent input/output waits are forensically meaningful signals that merit correlation with execution artifacts [44–47].

From an operational standpoint, probability calibration ensures that model outputs are interpretable as risk scores, enabling SOCs to set thresholds aligned with risk appetite and service-level objectives. In practice, deep learning/ensemble classifiers may exhibit overconfidence or underconfidence; Platt or isotonic calibration is well established to correct these effects, with supportive evidence spanning foundational work and recent applied studies [15,30–32]. Complementing calibration, the cumulative lift profile shows that a small fraction of highest-ranked events yields a disproportionate share of true positives, which is valuable for triage when analyst capacity is constrained. SHAP explanations further assist forensic validation and analyst trust by linking alerts to concrete causes (e.g., which command pattern, scheduling change, and periodicity) and facilitate feedback into defensive rules, such as command-line hardening or rate-limiting in the presence of resource-abuse signatures.

Positioned against prior ToN IoT studies, this work aimed to complement earlier contributions while addressing recurring gaps. Binary-only evaluations in single-model pipelines reported strong accuracy but did not test multiclass settings where overlap and minority classes complicate detection. Other works have examined multiclass classification but emphasized a narrow set of metrics or left probability calibration and model interpretability underspecified. By contrast, the ensemble in this study reduces variance and was evaluated with a broader suite—ROC AUC, calibration, and lift—to improve methodological comparability and operational relevance. The explicit use of SHAP and its mapping to recognized TTPs (e.g., T1059, T1105, and T1053.003) adds threat aware explainability, which is increasingly required for auditability and defensible decision making in enterprise environments [33,36,38].

At the system level, calibrated risk scores can be tied to organizational governance (e.g., escalation matrices and risk thresholds) to ensure that model outputs inform not only technical operations but also compliance and assurance workflows. SHAP-based narratives are audit ready and help justify containment or suppression decisions, while feature insights translate into preventive and detective controls (e.g., command-line controls, PID lifecycle monitoring, and CPU-based throttling). The consistent performance of the ensemble design across heterogeneous feature datasets supports scalability to diverse IoT deployments without sacrificing explainability or decision quality. Overall, these properties demonstrate how advanced machine learning/artificial intelligence approaches can be translated into implementable and policy-aware cybersecurity solutions that address evolving threats and the complex interdependencies of modern digital environments.

6 Conclusions

This paper presents a stacking ensemble deep learning model for multiclass intrusion detection on the ToN IoT Linux process dataset, integrating tree-based learners (RF and XGBoost) with a DNN as base models and a CatBoost meta-learner. The pipeline incorporated stratified OOF training, probability calibration, and post hoc explainability (SHAP), operating on 503,019 process-level observations with 16 host telemetry features (e.g., CMD, PID, CPU, Status/State, and PRI/NICE). This design was motivated by the need to translate high predictive power into decision quality outputs usable in security operations.

Empirically, the model delivered near perfect discrimination (macro AUC $\approx$ 0.9983) with well calibrated probabilities, a balanced operating point at a threshold of 0.16, and a maximum cumulative lift $\approx$ 7.0, which indicate that a small fraction of highest-ranked events captured a disproportionately large share of true attacks. SHAP analyses consistently elevated CMD, PID, Status/State, CPU, and temporal markers among the most influential signals, supporting threat informed interpretations of the decisions of the model. Collectively, these findings show that the proposed system achieves not only strong headline metrics but also the probabilistic reliability and explainability required for triage, escalation, and auditing in practical SOC workflows.

Beyond point estimates, the discussion established how these explanations map to operational security behaviors: command-line tokens and process spawn chains that reflect execution tactics; periodicity consistent with scheduled jobs or beacon-like polling; process churn indicative of scripted activity; and resource-abuse footprints suggestive of compute-bound implants. In this way, the outputs of the model can be connected to recognizable defensive levers, such as command-line hardening, PID lifecycle monitoring, and CPU-based throttling, and to policy constructs, such as risk thresholds and escalation matrices. These properties indicate a concrete path from advanced machine learning/artificial intelligence to implementable, policy aware cybersecurity analytics that acknowledge the complexity and interdependencies of modern digital environments.

Positioned against prior ToN-IoT research, the contribution addresses recurrent gaps identified in the literature: dependence on single-model pipelines, emphasis on binary tasks, narrow metric sets, limited calibration, and minimal interpretability. By combining meta-learning with a broader evaluation suite (ROC-AUC, calibration, and lift) and threat grounded explainability, the approach advances both methodological comparability and operational relevance while remaining complementary to earlier results on the same corpus.

Several constraints warrant attention. The present experiments are restricted to the Linux process dataset of ToN-IoT. Extending validation to Windows, network flow, and IoT sensor modalities is needed to assess cross-domain generalization. While SHAP provides actionable post hoc interpretability, the real-time cost of generating explanations and the potential susceptibility to adversarial manipulation remain open questions. Additionally, SHAP's reliability is not assured in all scenarios: its attributions may become unstable when features are highly correlated, as is common in network datasets. SHAP explanations also rely on the choice of background distribution, meaning that oversampling or class imbalance can affect attribution values. Moreover, SHAP reflects how the model uses features, not their causal influence, and explanations produced for the model may reflect interactions between base-model outputs rather than direct relationships with raw input features. Therefore, in this work SHAP explanations should be interpreted as supportive insights rather than definitive causal indicators. Finally, although the model is computationally efficient for offline training and near real time inference, deploying on resource constrained IoT edge nodes may require additional optimization or model distillation.

We plan to (i) evaluate the model across all ToN-IoT modalities to test scalability; (ii) explore cost-sensitive training aligned with organizational risk appetites; (iii) implement online calibration and drift detection to preserve probability reliability under distribution shift; and (iv) study robustness against adversarial attacks and data poisoning. In parallel, we aim to translate stable SHAP patterns (e.g., anomalous command forms or CPU spikes) into preventive policies and automated response rules, further tightening the loop between model outputs and defensive action.

Overall, the study demonstrates that a carefully engineered stacking ensemble deep learning, validated with comprehensive metrics, supported by calibrated probabilities, and explained through threat aware feature attributions, can be a reliable decision-support component for secure information systems, providing a practical bridge from research-grade IDS modeling to operational cybersecurity practice.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Faisal Albalwy and Muhannad Almohaimeed jointly contributed to conceptualization, methodology, software implementation, validation, formal analysis, investigation, data curation and writing. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The ToN-IoT Linux process dataset used in this study is publicly available from the ToN-IoT project repository. Processed data and scripts used for training and evaluation are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Abbreviations

IDS	Intrusion Detection System
IoT	Internet of Things
SOC	Security Operations Center
RF	Random Forest
XGB/XGBoost	Extreme Gradient Boosting
CatBoost	Categorical Boosting
DNN	Deep Neural Network
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
DL	Deep Learning
ML	Machine Learning
OOF	Out-of-Fold
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
SHAP	SHapley Additive exPlanations

SMOTE	Synthetic Minority Oversampling Technique
TPR	True Positive Rate
DoS	Denial of Service
DDoS	Distributed Denial of Service
XSS	Cross-Site Scripting
APT	Advanced Persistent Threat
MITRE ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
ReLU	Rectified Linear Unit (activation function)
ts	Data collection timestamp
PID	Process Identifier
TRUN	Number of threads in state 'running'
TSLPI	Number of interruptible sleeping threads
TSLPU	Number of uninterruptible sleeping threads
POLI	Process scheduling policy
NICE	Process nice value
PRI	Numerical process priority
RTPR	Real-time process priority
CPUNR	CPU core number where the process is executing
Status	Process status indicator for newly started processes
EXC	Process exit code or fatal signal
State	Current process state
CPU	CPU time consumption in system mode
CMD	Process command name

References

1. Ullah F, Turab A, Ullah S, Cacciagrano D, Zhao Y. Enhanced network intrusion detection system for Internet of Things security using multimodal big data representation with transfer learning and game theory. Sensors. 2024;24(13):4152. doi:10.3390/s24134152.
2. Alotaibi B. A survey on industrial Internet of Things security: requirements, attacks, AI-based solutions, and edge computing opportunities. Sensors. 2023;23(17):7470. doi:10.3390/s23177470.
3. Abdel Wahab O. Intrusion detection in the IoT under data and concept drifts: online deep learning approach. IEEE Internet Things J. 2022;9(20):19706–16. doi:10.1109/jiot.2022.3167005.
4. Guo Y. A review of Machine Learning-based zero-day attack detection: challenges and future directions. Comput Commun. 2023;198:175–85. doi:10.1016/j.comcom.2022.11.001.
5. Rakine I, El Guemmat K, Ouahabi S, Atouf I, Talea M. IoT intrusion detection: a review of ML and DL-based approaches. In: Proceedings of the 2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET); 2024 May 16–17; Fes, Morocco. p. 1–7. doi:10.1109/IRASET60544.2024.10548107.
6. Alrefaei A, Ilyas M. Using machine learning multiclass classification technique to detect IoT attacks in real time. Sensors. 2024;24(14):4516. doi:10.3390/s24144516.
7. Ohtani T, Yamamoto R, Ohzahata S. IDAC: federated learning-based intrusion detection using autonomously extracted anomalies in IoT. Sensors. 2024;24(10):3218. doi:10.3390/s24103218.
8. Neto ECP, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA. CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors. 2023;23(13):5941. doi:10.3390/s23135941.
9. Belarbi O, Spyridopoulos T, Anthi E, Mavromatis I, Carnelli P, Khan A. Federated deep learning for intrusion detection in IoT networks. In: Proceedings of the GLOBECOM 2023—2023 IEEE Global Communications Conference; 2023 Dec 4–8; Kuala Lumpur, Malaysia. p. 237–42. doi:10.1109/globecom54140.2023.10437860.
10. Khan NW, Alshehri MS, Khan MA, Almakdi S, Moradpoor N, Alazeb A, et al. A hybrid deep learning-based intrusion detection system for IoT networks. Math Biosci Eng. 2023;20(8):13491–520. doi:10.3934/mbe.2023602.

11. Elsayed RA, Hamada RA, Abdalla MI, Elsaid SA. Securing IoT and SDN systems using deep-learning based automatic intrusion detection. Ain Shams Eng J. 2023;14(10):102211. doi:10.1016/j.asej.2023.102211.
12. Alotaibi Y, Ilyas M. Ensemble-learning framework for intrusion detection to enhance internet of things' devices security. Sensors. 2023;23(12):5568. doi:10.3390/s23125568.
13. Soni, Remli MA, Mohd Daud K, Al Amien J. Ensemble learning approach to enhancing binary classification in Intrusion Detection System for Internet of Things. Int J Electron Telecommun. 2024;465–72. doi:10.24425/ijet.2024.149567.
14. Elsayed R, Hamada R, Hammoudeh M, Abdalla M, Elsaid SA. A hierarchical deep learning-based intrusion detection architecture for clustered Internet of Things. J Sens Actuator Netw. 2023;12(1):3. doi:10.3390/jsan12010003.
15. Guo C, Pleiss G, Sun Y, Weinberger QK. On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning; 2017 Aug 6–11; Sydney, Australia. p. 1321–30.
16. Jamgharyan T, Yarovikova O. Research on classifier calibration methods in network infrastructure. In: Proceedings of "Computer Science and Information Technologies" International Conference; 2025 Sep 22–26; Yerevan, Armenia. p. 303–6. doi:10.51408/csit2025_73.
17. Pawlicki M, Pawlicka A, Kozik R, Choraś M. The survey on the dual nature of xAI challenges in intrusion detection and their potential for AI innovation. Artif Intell Rev. 2024;57(12):330. doi:10.1007/s10462-024-10972-3.
18. Neupane S, Ables J, Anderson W, Mittal S, Rahimi S, Banicescu I, et al. Explainable intrusion detection systems (X-IDS): a survey of current methods, challenges, and opportunities. IEEE Access. 2022;10:112392–415. doi:10.1109/access.2022.3216617.
19. Moustafa N, Ahmed M, Ahmed S. Data analytics-enabled intrusion detection: evaluations of ToN_IoT linux datasets. In: Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom); 2020 Dec 29–2021 Jan 1; Guangzhou, China. p. 727–35. doi:10.1109/trustcom50675.2020.00100.
20. Gad AR, Haggag M, Nashat AA, Barakat TM. A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset. Int J Adv Comput Sci Appl. 2022;13(6):548–63. doi:10.14569/ijacsa.2022.0130667.
21. Naeem F, Malik AW, Khan SA, Jabeen F. Enhancing intrusion detection: leveraging federated learning and hybrid machine learning algorithms on ToN_IoT dataset. In: Proceedings of the 2023 International Conference on Frontiers of Information Technology (FIT); 2023 Dec 11; Islamabad, Pakistan. p. 73–8.
22. Kaddour H, Das S, Bajgai R, Sanchez A, Sanchez J, Chiu SC, et al. Evaluating the performance of machine learning-based classification models for IoT intrusion detection. In: Proceedings of the 2024 IEEE Opportunity Research Scholars Symposium (ORSS); 2024 Apr 15–Jul 15; Atlanta, GA, USA. p. 84–7. doi:10.1109/orss62274.2024.10697949.
23. Oseni A, Moustafa N, Creech G, Sohrabi N, Strelzoff A, Tari Z, et al. An explainable deep learning framework for resilient intrusion detection in IoT-enabled transportation networks. IEEE Trans Intell Transp Syst. 2023;24(1):1000–14. doi:10.1109/tits.2022.3188671.
24. Cao Z, Zhao Z, Shang W, Ai S, Shen S. Using the ToN-IoT dataset to develop a new intrusion detection system for industrial IoT devices. Multimed Tools Appl. 2025;84(16):16425–53. doi:10.1007/s11042-024-19695-7.
25. Shaker BN, Al-Musawi BQ, Hassan MF. A comparative study of IDS-based deep learning models for IoT network. In: Proceedings of the 2023 International Conference on Advances in Artificial Intelligence and Applications; 2023 Nov 18–20; Wuhan, China. New York, NY, USA: The Association for Computing Machinery (ACM); 2023. p. 15–21. doi:10.1145/3603273.3635058.
26. Sneha M, Prasad GR. Transparent ensemble deep learning for intrusion detection in industrial Internet of Things. In: Proceedings of the 2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC); 2024 Oct 24–25; Davangere, India. p. 1–7. doi:10.1109/icicec62498.2024.10808331.
27. Eren KK, Küçük K. Improving intrusion detection systems for IoT devices using automated feature generation based on ToN_IoT dataset. In: Proceedings of the 2023 8th International Conference on Computer Science and Engineering (UBMK); 2023 Sep 13–15; Burdur, Turkiye. p. 276–81. doi:10.1109/ubmk59864.2023.10286655.
28. Nguyen TM, Vo HH, Yoo M. Enhancing intrusion detection in wireless sensor networks using a GSWO-CatBoost approach. Sensors. 2024;24(11):3339. doi:10.3390/s24113339.

29. Cai Y, Yuan Y, Zhou A. Predictive slope stability early warning model based on CatBoost. Sci Rep. 2024;14(1):25727. doi:10.1038/s41598-024-77058-6.
30. Niculescu-Mizil A, Caruana R. Predicting good probabilities with supervised learning. In: Proceedings of the 22nd International Conference on Machine Learning—ICML'05; 2005 Aug 7–11; Bonn, Germany. New York, NY, USA: The Association for Computing Machinery (ACM); 2005. p. 625–32. doi:10.1145/1102351.1102430.
31. Niculescu-Mizil A, Caruana R. Obtaining calibrated probabilities from boosting. In: Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence; 2025 Jul 21–25; Rio de Janeiro, Brazil. p. 413–20.
32. Phelps N, Lizotte DJ, Woolford DG. Using Platt's scaling for calibration after undersampling—limitations and how to address them. arXiv:2410.18144. 2024.
33. MITRE ATT&CK. Command and Scripting Interpreter, Technique T1059—Enterprise | MITRE ATT&CK [Internet]. [cited 2025 Nov 20]. Available from: https://attack.mitre.org/techniques/T1059/.
34. Alageel A, Maffeis S. EarlyCrow: detecting APT malware command and control over HTTP(S) using contextual summaries. In: Information security. Cham, Switzerland: Springer International Publishing; 2022. p. 290–316. doi:10.1007/978-3-031-22390-7_18.
35. Lin J, Guo Y, Chen H. Intrusion detection at scale with the assistance of a command-line language model. In: Proceedings of the 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Supplemental Volume (DSN-S); 2024 Jun 24–27; Brisbane, Australia. p. 73–9.
36. MITRE ATT&CK. T1105: ingress Tool Transfer [Internet]. [cited 2025 Nov 20]. Available from: https://attack.mitre.org/techniques/T1105/.
37. Roy S, Panaousis E, Noakes C, Laszka A, Panda S, Loukas G. SoK: the MITRE ATT&CK framework in research and practice. arXiv:2304.07411. 2023.
38. MITRE ATT&CK. Scheduled Task/Job [Internet]. [cited 2025 Nov 20]. Available from: https://attack.mitre.org/techniques/T1053/003/.
39. Abu Talib M, Nasir Q, Bou Nassif A, Mokhamed T, Ahmed N, Mahfood B. APT beaconing detection: a systematic review. Comput Secur. 2022;122:102875. doi:10.1016/j.cose.2022.102875.
40. Shah KD, Patel KV. Security against fork bomb attack in linux based systems. Int J Res Advent Technol. 2019;7(4):125–8. doi:10.32622/ijrat.74201911.
41. Xu B, Gong Y, Geng X, Li Y, Dong C, Liu S, et al. ProcSAGE: an efficient host threat detection method based on graph representation learning. Cybersecurity. 2024;7(1):51. doi:10.1186/s42400-024-00240-w.
42. Gomes F, Correia M. Cryptojacking detection with CPU usage metrics. In: Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA); 2020 Nov 24–27; Cambridge, MA, USA. p. 1–10. doi:10.1109/nca51143.2020.9306696.
43. Naseem F, Aris A, Babun L, Tekiner E, Uluagac AS. MINOS: a lightweight real-time cryptojacking detection system. In: Proceedings 2021 Network and Distributed System Security Symposium; 2021 Feb 21–25; Virtual. p. 1–15. doi:10.14722/ndss.2021.24444.
44. GeeksforGeeks. Priority of Process in Linux | Nice Value [Internet]. [cited 2025 Nov 20]. Available from: https://www.geeksforgeeks.org/linux-unix/priority-of-process-in-linux-nice-value/.
45. Zhang H, Li B, Yu S, Chang C, Li J, Yang B. ProcGCN: detecting malicious process in memory based on DGCNN. PeerJ Comput Sci. 2024;10:e2193. doi:10.7717/peerj-cs.2193.
46. Singh N, Rebeiro C. LEASH: enhancing micro-architectural attack detection with a reactive process scheduler. arXiv:2109.03998. 2021.
47. Butt MA, Akram M. A new intuitionistic fuzzy rule-based decision-making system for an operating system process scheduler. SpringerPlus. 2016;5(1):1547. doi:10.1186/s40064-016-3216-z.