Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 11, pp 174-186, November, 2026

Quantum-Powered Cyber Attack Detection and Prevention System

T. Debanga Saikia, P. Abhigna, P. Charan Teja, L. Harika, P. Samuel

Department of CSE-AI & ML, GMR Institute of Technology (GMRIT) – Deemed to be University, Rajam 532127, Andhra Pradesh, India

Email: abhignapedagadi2122@gmail.com, charantejpariserla@gmail.com, harika@gmritdu.edu.in, samuel@saec.ac.in

ABSTRACT:

The increasing complexity of cyberattacks creates a need for intelligent systems capable of accurate detection, explainable analysis, and adaptive response. This paper presents a Quantum-Powered Explainable Cyber Attack Detection and Prevention System, Q-XCADPS, an enterprise Security Operations Center that integrates classical machine learning, quantum machine learning, explainable artificial intelligence, and adaptive response mechanisms. The system detects Distributed Denial-of-Service attacks, SQL Injection, malware, brute-force attacks, and phishing URLs. Random Forest and XGBoost are evaluated along with Quantum Support Vector Classifier and Variational Quantum Classifier using a four-qubit architecture with PCA-based dimensionality reduction. Explainability is provided using SHAP-based feature analysis, while the adaptive response engine assigns threat scores and severity levels. Experimental results show that XGBoost achieves the highest performance with 90.50 percent accuracy and a weighted F1-score of 0.8651. The quantum models demonstrate the feasibility of quantum machine learning for cybersecurity detection.

Keywords: *Cybersecurity, Quantum Machine Learning, Explainable AI, Cyber Attack Detection, Q-XCADPS, QSVC, VQC, XGBoost, SHAP.*

1. INTRODUCTION

Cybersecurity has become an important concern with the increasing use of websites, computer networks, and online services. The continuous growth of network traffic has also increased the possibility of different types of cyber attacks. Attacks such as Distributed Denial of Service, SQL injection, malware, brute-force attacks, and phishing can affect system availability, steal sensitive information, and cause financial and operational damage. Therefore, detecting malicious activities at an early stage is important for protecting modern computer networks. Traditional Intrusion Detection Systems and machine learning techniques are commonly used to identify abnormal network activities. Classical machine learning models can provide effective detection performance, particularly for structured network data. However, large volumes of network traffic, complex feature relationships, and changing attack patterns create challenges for developing detection systems that are both accurate and efficient. These challenges have encouraged researchers to investigate new computational approaches for cybersecurity.Quantum Machine Learning (QML) has recently attracted attention as a possible approach for cybersecurity applications. QML combines machine learning techniques with quantum computing and provides new methods for representing and processing data. Quantum models such as Quantum Support Vector Classifier and Variational Quantum Classifier have been investigated for intrusion and anomaly detection. Previous research has also explored hybrid quantum-classical approaches for improving network security. However, the practical use of quantum machine learning for cybersecurity still presents several challenges. Current quantum computing systems have limitations related to the number of available qubits, circuit complexity, noise, and computational resources. Therefore, quantum models need to be evaluated carefully and compared with classical models to understand their practical advantages and limitations. Research on quantum intrusion detection also highlights the need to consider performance and computational efficiency when developing security applications.

Unlike a system that only classifies network traffic, the proposed framework is designed to support the complete process from detection and explanation to threat assessment and response. The main contributions of this work are:

- Integration of **cyber attack detection, quantum machine learning, explainable AI, threat assessment, and response mechanisms** within a single cybersecurity framework.
- Evaluation of two classical models (Random Forest, XGBoost) and two quantum models (QSVC, VQC) on network traffic and URL data, using a four-qubit architecture with PCA-based dimensionality reduction.

- SHAP-based feature-level explanations that allow security analysts to understand the reasons behind individual detection results.
- A threat score (0 to 100) with LOW, MEDIUM, HIGH and CRITICAL severity levels, configurable response actions, and a dashboard for monitoring.

The rest of the paper is organized as follows. Section 2 reviews related work and identifies the research gap. Section 3 describes the proposed system architecture. Section 4 presents the methodology and the classification models. Section 5 reports the experimental results. Section 6 discusses the findings, and Section 7 concludes with future scope.

2. RELATED WORK

Recent research has explored classical, quantum, and hybrid quantum-classical machine-learning approaches for cybersecurity and network intrusion detection. Several studies have investigated QSVC, VQC, QNN, quantum autoencoders, and hybrid architectures for intrusion and anomaly detection, demonstrating the potential of quantum feature encoding and quantum representation learning [1–5]. However, these approaches continue to face challenges related to quantum noise, limited qubit availability, computational overhead, scalability, and practical deployment.

Classical machine-learning and deep-learning techniques remain widely used for intrusion detection, IoT security, and network anomaly detection. Studies have demonstrated the effectiveness of models such as Random Forest, XGBoost, Gradient Boosting, CNN, RNN, LSTM, and Transformer-based architectures [6–10]. Although these approaches provide strong predictive performance and practical efficiency, they may require extensive feature engineering and often provide limited interpretability.

Explainable machine learning has therefore gained importance in cybersecurity applications, particularly for malicious-URL and phishing detection. Recent studies have incorporated SHAP, ensemble learning, and deep-learning techniques to improve both detection performance and the interpretability of security decisions [12–15]. These works demonstrate the importance of providing security analysts with understandable explanations rather than relying solely on model predictions.

Despite these advances, existing studies generally focus on individual aspects such as intrusion detection, phishing detection, quantum learning, or explainability. Limited work integrates classical and quantum detection, explainable AI, threat intelligence, risk assessment, real-time alerting, and automated response within a single end-to-end security operations framework. The proposed Q-XCADPS addresses this gap by combining classical ML and QML models with XAI, threat scoring, threat intelligence, real-time monitoring, and adaptive response mechanisms to provide an integrated and explainable cybersecurity platform.

The reviewed literature indicates that both classical and quantum machine-learning techniques have been extensively investigated for intrusion detection, anomaly detection, IoT security, and malicious URL classification. Existing quantum-based studies primarily investigate the feasibility of QSVM, QSVC, VQC, QNN, and hybrid quantum-classical architectures, with particular attention to quantum noise, circuit complexity, feature encoding, and computational requirements. Classical approaches continue to provide strong performance and practical efficiency, particularly for structured network and URL data. Explainable approaches using SHAP have also demonstrated the importance of interpreting model predictions in security-sensitive applications. Table 1 summarizes the reviewed studies.

Table 1. Summary of the reviewed literature

Ref.	Year	Approach / technique	Domain / dataset
[1]	2026	QATNet: amplitude encoding, shallow variational quantum circuit and attentive TabNet	Low-latency network intrusion detection; contemporary IDS datasets; quantum-noise conditions
[2]	2024	Quantum autoencoders with quantum one-class SVM, quantum Random Forest and quantum KNN	Network anomaly detection; KDD99, IoT-23, CIC-IoT-2023
[3]	2024	Ultra-lightweight quantum neural network with efficient NetFlow feature encoding	Network anomaly detection on noisy quantum computers; NF-UNSW-NB15
[4]	2022	QSVM and QCNN compared with conventional machine learning	Intrusion detection on large-scale network data

Ref.	Year	Approach / technique	Domain / dataset
[5]	2026	Benchmark of QSVC, VQC and hybrid quantum-classical neural network	Intrusion detection under ideal and noisy conditions; ToN-IoT, NSL-KDD
[6]	2025	Optimized Decision Tree, Random Forest, KNN and Gradient Boosting with data balancing	Network intrusion detection
[7]	2026	Feature selection and optimization with RNN and MLP; VQC-based QNN extension	IoT intrusion detection; NSL-KDD, UNSW-NB15
[8]	2026	Quantum-inspired hyperparameter optimization of CNN, GAN, GRU, LSTM and Transformer	DoS detection in wireless sensor networks; WSN-DS
[9]	2025	QCNN-ID: PCA with quantum convolutional and classical CNN components	Healthcare IoT intrusion detection; BoT-IoT and healthcare IoT datasets
[10]	2023	Optimized quantum neural network with elliptic curve cryptography	Intrusion detection with data protection
[11]	2025	Quantum-driven Zero Trust architecture with QNN-based anomaly detection	Dynamic anomaly detection for 7G networks
[12]	2026	Explainable ML and DL with feature selection and SHAP	Malicious URL detection
[13]	2026	LR, RF, Gradient Boosting, XGBoost and SVM compared with a DNN; SHAP	Fraudulent URLs on Arabic social media; 1,600 URLs
[14]	2025	XGBoost, LSTM and semantic analysis ensemble	Phishing URL classification
[15]	2026	Random Forest compared with a deep-learning model	Real-time phishing URL detection; 50,000 labelled URLs

3. PROPOSED SYSTEM ARCHITECTURE

3.1 Overall architecture

The Quantum-Powered Cyber Attack Detection and Prevention System follows a sequence of processing stages, beginning with input collection and ending with threat assessment, explanation, and response generation. The system accepts network traffic data or a website URL as input to identify potentially malicious activities. Network traffic data contain information about communication patterns, packet statistics, protocol behaviour, and suspicious activity indicators, whereas URL data contain structural and domain-related characteristics that may indicate phishing or other malicious behaviour.

The extracted input features undergo data preprocessing to ensure that they are suitable for classification. This stage includes data cleaning, categorical feature encoding, numerical normalization, and dimensionality reduction where required. For the quantum classification pipeline, Principal Component Analysis (PCA) reduces the feature representation to four principal components. This reduction makes the input more manageable for quantum circuit processing.

After preprocessing, the data are passed to the selected classification model. The system implements two classical machine-learning models, Random Forest and XGBoost, and two quantum machine-learning models, Quantum Support Vector Classifier (QSVC) and Variational Quantum Classifier (VQC). Each model analyses the input features and generates a prediction based on the patterns learned during training. The outputs are compared to

evaluate the suitability of classical and quantum approaches for cyberattack detection.

Following classification, the system assigns a threat score ranging from 0 to 100 and categorizes the detected activity into LOW, MEDIUM, HIGH, or CRITICAL severity levels. The score and severity category are used to prioritize security events. SHAP-based explainability is incorporated to identify the features influencing the model's predictions, thereby providing additional information about why an activity has been classified as legitimate or suspicious.

When a potential threat is detected, the system generates an alert containing the prediction and relevant threat information. Depending on the configured response policy, actions such as monitoring, event logging, rate limiting, temporary blocking, suspicious URL blocking, or critical alert generation can be selected. VirusTotal integration supplies additional threat-intelligence information for suspicious URLs when available. The detection results, explanations, and security events are stored in MongoDB, while the React-based dashboard displays the results for monitoring. Flask manages backend requests and coordinates communication between the frontend, detection models, and storage components.

3.2 Technology stack

Table 2. Technology stack of the proposed system

Layer	Technology	Purpose
Front end	React dashboard	Displays detection results, explanations and security events for monitoring.
Back end	Flask	Manages backend requests and coordinates the frontend, detection models and storage components.
Database	MongoDB	Stores detection results, explanations and security events.
Preprocessing	Data cleaning, categorical encoding, normalization; PCA	Prepares input features; reduces them to four principal components for quantum processing.
Classical ML	Random Forest, XGBoost	Classification of network traffic and URL features.
Quantum ML	QSVC, VQC,QSVC	Quantum classification of the four PCA components.
Explainability	SHAP (TreeExplainer)	Feature-level explanation of model predictions.
Threat assessment	Threat score (0 to 100); LOW, MEDIUM, HIGH, CRITICAL	Prioritizes detected security events.
Threat intelligence	VirusTotal	Supplementary information for suspicious URLs, when available.

4. METHODOLOGY

4.1 Random Forest classifier

Random Forest is a supervised machine-learning algorithm used to classify data based on patterns learned from labelled training samples. In the Quantum-Powered Cyber Attack Detection and Prevention System, Random Forest is used to analyse network traffic and URL features to identify legitimate activities and potential cyber threats. The algorithm combines multiple decision trees to produce a final classification, improving the reliability of predictions compared with relying on a single decision tree.

The Random Forest classification process begins with the input data, which may consist of network traffic records or website URLs. The extracted features are preprocessed through data cleaning, categorical encoding, and numerical normalization, as required by the corresponding input pipeline. The processed data are then supplied to the trained Random Forest model for classification (Fig. 1).

During training, the Random Forest algorithm constructs multiple decision trees using randomly selected training samples and subsets of features. Each decision tree learns relationships between the input features and their corresponding class labels. For example, when analysing network traffic, a tree may examine packet counts, flow duration, packet transmission rates, or failed login attempts to identify patterns associated with suspicious behaviour. When analysing URLs, the model may examine URL length, domain length, HTTPS usage, and other structural characteristics to distinguish legitimate URLs from potentially malicious ones.

When a new input is provided, it is evaluated by the decision trees in the trained forest. Each tree generates a class prediction, and the model combines these predictions to determine the final classification. For classification tasks, this decision is generally based on majority voting or the aggregation of class probabilities, depending on the model configuration. The output identifies the predicted category, such as legitimate activity or a supported cyberattack class.

After classification, the system uses the prediction in its threat-assessment stage to assign a risk score and severity category. SHAP TreeExplainer is used to explain the Random Forest prediction by estimating the contribution of individual features. These explanations help identify the attributes that influenced the model's decision and provide additional information for security analysis. The system can then generate an alert and select a configured response action according to the assessed threat level. The performance of the Random Forest classifier is evaluated using accuracy, precision, recall, F1-score, ROC-AUC where applicable, and inference latency. These metrics help assess its ability to identify cyber threats while limiting incorrect classifications and maintaining suitable response times. The resulting predictions and explanations can be displayed on the dashboard and stored in MongoDB for monitoring and subsequent review.

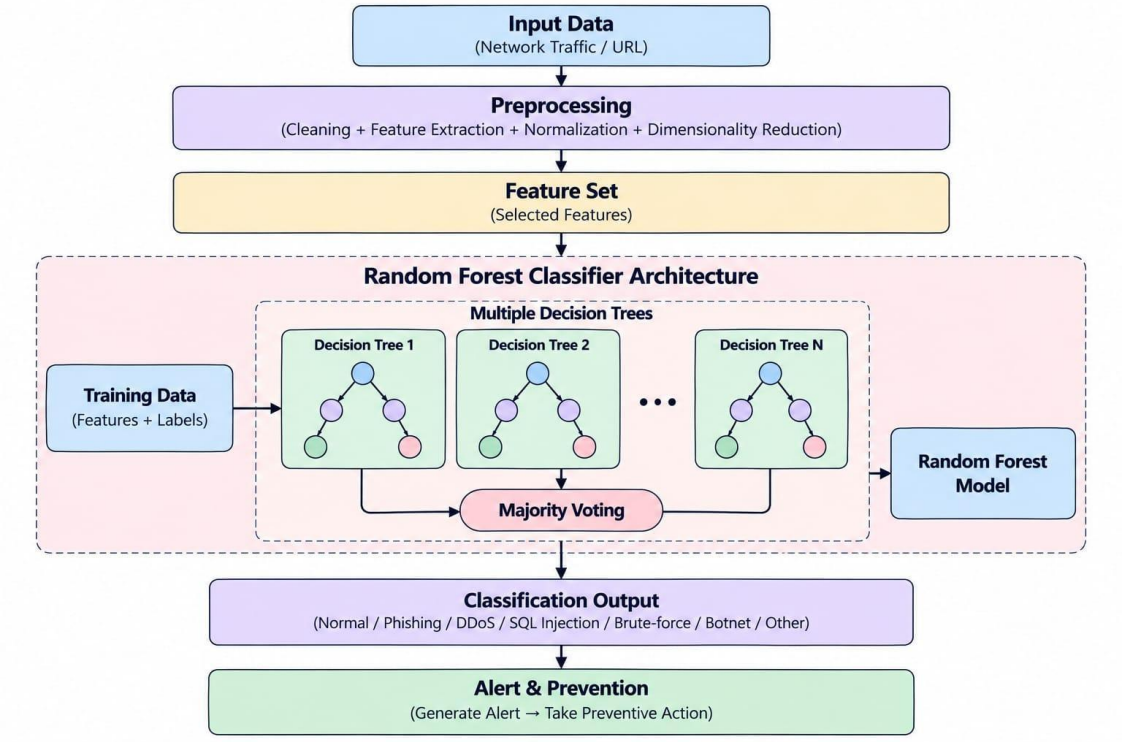


Fig. 1. Random Forest classifier architecture

4.2 Quantum Support Vector Classifier (QSVC)

The Quantum Support Vector Classifier (QSVC) is a quantum machine-learning model that combines the principles of Support Vector Machines (SVMs) with quantum computing techniques to classify data. In the Quantum-Powered Cyber Attack Detection and Prevention System, QSVC is used to analyse network traffic and URL features to identify potentially malicious activities. The model uses quantum feature encoding and a quantum kernel to represent and compare input samples, enabling the support vector classification algorithm to identify patterns associated with different classes (Fig. 2).The QSVC classification process begins with data preprocessing. The input features are cleaned, encoded, and normalized according to the requirements of the corresponding data pipeline. Principal Component Analysis (PCA) is then applied to reduce the feature representation to four principal components for quantum processing. Dimensionality reduction is important because processing a large number of input features can increase the computational requirements of quantum circuits. The reduced feature vectors are subsequently passed to the quantum feature-encoding stage.

In this system, the ZZFeatureMap provided by the Qiskit framework is used to encode the reduced classical features into quantum states. The feature map represents the input data through quantum circuit operations and incorporates relationships between features. This encoding transforms the classical feature vectors into quantum representations that can be used to calculate similarities between samples. The quality of this representation influences how

effectively the classifier can distinguish between different classes.After feature encoding, the quantum kernel calculates similarity values between pairs of encoded samples. These values form a kernel matrix that represents the relationships between the training samples in the quantum feature space. The support vector classification algorithm uses this matrix to identify a decision boundary that separates the target classes. During training, the classifier learns the relevant decision boundary from labelled samples. During prediction, a new input sample is encoded using the same feature map, and its kernel similarities with the training samples are used to determine the predicted class.

In the cyberattack detection system, QSVC can be applied to the supported network traffic and URL classification pipelines. Depending on the input dataset and its class labels, the model predicts whether an activity is legitimate or belongs to a supported attack category. Its predictions are subsequently used by the system's threat-assessment component to generate a risk score and assign a severity level. The resulting information can be presented on the dashboard and used to support the generation of security alerts.To improve interpretability, SHAP KernelExplainer is used to estimate the contributions of the reduced input features to the QSVC predictions. Since the quantum model operates on four PCA components, the resulting explanations initially correspond to those components rather than directly to the original input features. In the implemented explanation pipeline, the absolute PCA loadings are used to project these contributions back towards the original feature space. This provides an approximate indication of the original features associated with the prediction; it should not be interpreted as an exact decomposition of each feature's contribution.

The performance of QSVC is evaluated alongside Random Forest, XGBoost, and VQC using classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC where applicable. Inference latency is also measured to examine the computational cost of generating predictions. The evaluation helps determine how QSVC performs under the tested conditions and identifies its practical limitations.

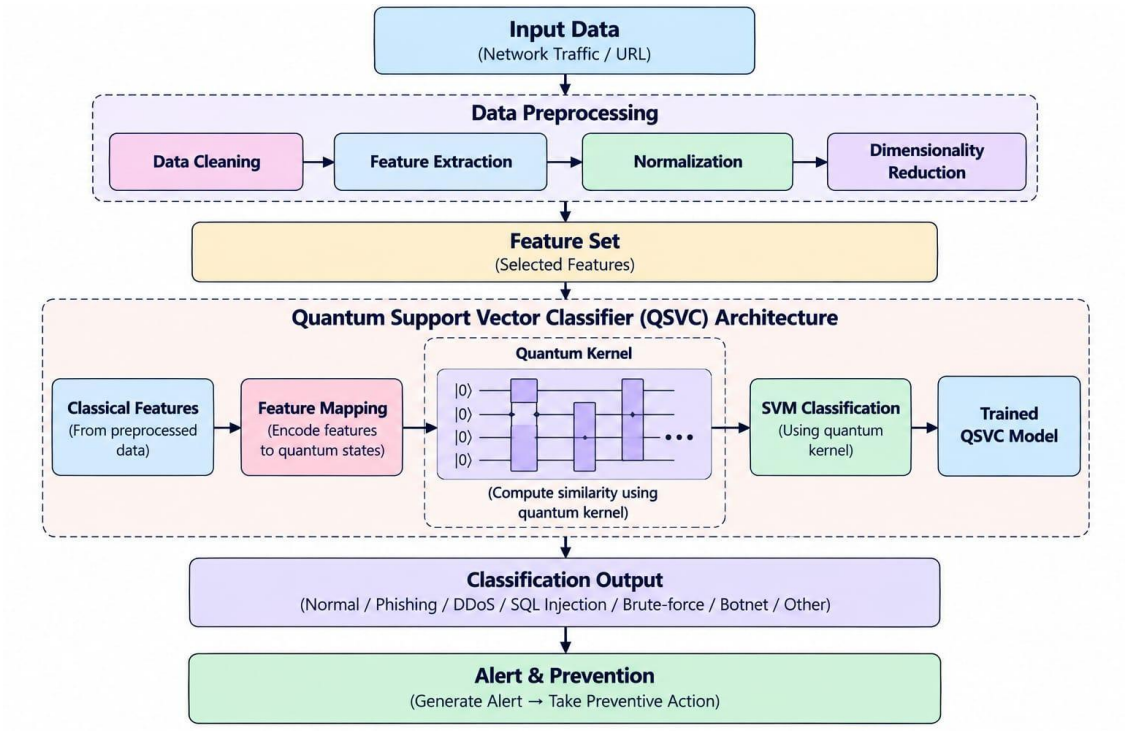


Fig. 2. Quantum Support Vector Classifier architecture

4.3 XGBoost classifier

XGBoost (Extreme Gradient Boosting) is a supervised machine-learning algorithm that uses an ensemble of decision trees to perform classification. It is based on the gradient boosting technique, in which multiple decision trees are constructed sequentially to improve the predictive performance of the overall model. Unlike Random Forest, which builds multiple trees that contribute to an aggregated prediction, XGBoost builds each successive tree to reduce the errors of the existing ensemble. This process is guided by a loss function that measures the difference between the predicted outputs and the actual class labels (Fig. 3).

In the Quantum-Powered Cyber Attack Detection and Prevention System, XGBoost is used to analyse network traffic and URL features to identify potentially malicious activities. The process begins with the extraction and preprocessing of input features. For network traffic analysis, these features include flow duration, packet counts, flow byte rate, packet rate, TCP flag counts, payload entropy, failed login attempts, and protocol information. For URL analysis, the features include URL length, domain length, the number of dots and hyphens, HTTPS usage, domain age, and top-level domain indicators. The prepared feature vectors are supplied to the trained XGBoost classifier for prediction.During training, the algorithm initially constructs a prediction based on the available training data. It then calculates the errors associated with the current predictions and builds additional decision trees to

improve the model's performance. Each successive tree contributes to the existing ensemble by correcting patterns of errors identified during the previous stage. The final classification is obtained by combining the contributions of the individual trees. Through this iterative learning process, XGBoost can identify complex relationships between input features and the corresponding class labels.

XGBoost optimizes a regularized objective function that combines a differentiable loss term with a penalty on model complexity (Eq. 1 and Eq. 2). The loss measures the difference between the predicted and actual class labels, using a logistic loss for binary decisions or a softmax loss when multiple attack categories are considered. The regularization term penalizes the number of leaves and the magnitude of the leaf weights, which limits overfitting and improves generalization to unseen traffic.

$$Obj = \Sigma_i\, l(y_i, \hat{y}_i) + \Sigma_k\, \Omega(f_k) \qquad (1)$$

$$\Omega(f) = \gamma T + \tfrac{1}{2}\, \lambda\, \|w\|^2 \qquad (2)$$

Here, y_i is the true label and $\hat{y}_i$ the predicted output for sample i, f_k is the k-th tree, T is the number of leaves in a tree, w is the vector of leaf weights, and γ and λ are regularization coefficients. At each boosting iteration, the objective is approximated using the first-order gradient and second-order derivative (Hessian) of the loss. These statistics are used to evaluate candidate splits through a gain criterion and to compute the optimal weight of each leaf, so that a new tree is added only when it reduces the regularized objective.The behaviour of the model is controlled by a small set of hyperparameters: the number of boosting rounds, the learning rate (shrinkage), the maximum tree depth, the minimum child weight, the row and column subsampling ratios, and the regularization coefficients. Shrinkage scales the contribution of each new tree so that no single tree dominates the ensemble, while row and column subsampling introduce randomness that reduces correlation between trees. These settings balance model capacity against overfitting and are tuned on the training data.

XGBoost is well suited to the structured, tabular data used in this work. It handles features of different scales and types, such as packet counts, byte rates, entropy values, and binary URL indicators, without requiring complex transformations. Its sparsity-aware split finding manages missing values, and class weighting can be applied when benign and attack samples are imbalanced. Prediction requires only a traversal of the trained trees, which makes inference fast and explains the low latency observed for this model in Section 5. In addition, tree-based feature importance and the SHAP TreeExplainer provide transparent explanations of which features contributed to each detection, supporting the explainability objective of the system.

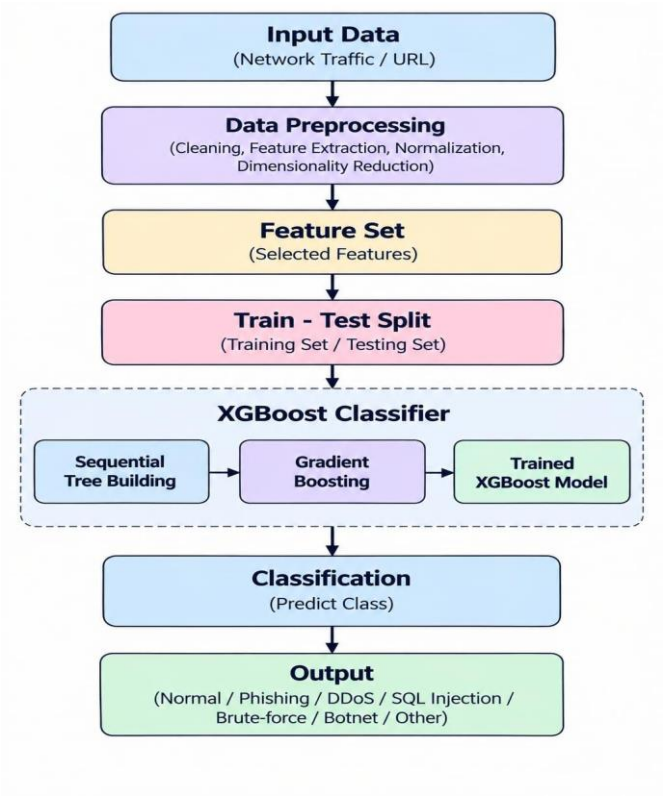


Fig. 3. XGBoost classifier architecture

4.4 Variational Quantum Classifier (VQC)

The Variational Quantum Classifier (VQC) is a quantum machine-learning model that uses a parameterized quantum circuit to learn classification patterns.

The input data are preprocessed and reduced to four principal components before being encoded into the quantum circuit. A parameterized variational circuit then performs trainable quantum operations on the encoded representation (Fig. 4).

During training, the circuit parameters are optimized according to the classification objective. The resulting measurement outputs are used to determine the predicted class. The trained VQC model can then classify new network traffic or URL feature vectors within the supported input pipeline.

The VQC follows a hybrid quantum-classical workflow. After standardization, the feature vectors are reduced to four principal components using PCA, so that each component can be assigned to one qubit of a four-qubit circuit. This reduction is necessary because the number of features that can be encoded is limited by the number of available qubits, although it may discard part of the information contained in the original feature set.

The classical features are first mapped into a quantum state by a quantum feature map, which applies rotation gates whose angles depend on the input values, optionally followed by entangling gates. The encoded state is then transformed by a parameterized variational circuit (ansatz). The ansatz consists of repeated layers of trainable single-qubit rotations and entangling gates, which create correlations between the qubits and allow the circuit to represent non-linear decision boundaries. The state prepared by the complete circuit is given in Eq. 3.

$$|\psi(x, \theta)\rangle = U(\theta)\, U_{\varphi}(x)\, |0\rangle^{\otimes 4} \qquad (3)$$

Here, x is the reduced input vector, $U_{\varphi}(x)$ is the feature map, $U(\theta)$ is the variational circuit, and θ is the set of trainable parameters. The qubits are then measured in the computational basis. The measurement outcomes are repeated over a number of shots to estimate the probability of each bit string, and a mapping function assigns these probabilities to the output classes.Training is performed by a classical optimizer that updates θ to minimize a cost function, such as the cross-entropy loss between the predicted class probabilities and the true labels (Eq. 4). In each iteration, the quantum circuit is evaluated for the training samples, the loss is computed classically, and the optimizer adjusts the parameters until the loss converges or a maximum number of iterations is reached. Gradient-free optimizers such as COBYLA and SPSA are commonly used for this purpose.

$$L(\theta) = -(1/N) \sum_i \log p_{yi}(x_i, \theta) \qquad (4)$$

Several characteristics of the VQC help explain its results in this study. Training requires many repeated circuit evaluations, and each prediction requires multiple measurement shots, which increases inference latency compared with the classical models. The four-qubit limit restricts the input dimensionality, and the optimization of the circuit parameters can be affected by local minima and flat regions of the cost landscape. These factors may contribute to the lower accuracy and higher latency of the VQC relative to the classical classifiers, while the model still demonstrates that quantum circuits can be trained for cyberattack detection.

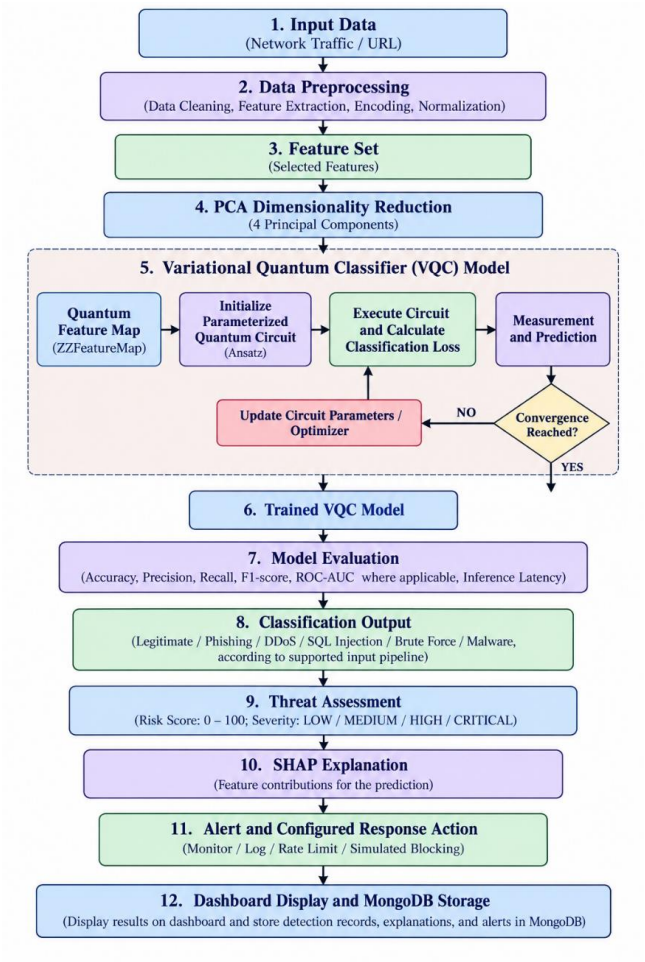


Fig. 4. Variational Quantum Classifier (VQC) architecture

4.5 Comparison of classical and quantum models

Table 3 compares the two classical and two quantum models on their technique, feature handling, explainability, measured performance and main trade-offs.

Table 3. Comparative performance analysis of classical machine learning and quantum machine learning models

Feature	Random Forest	XGBoost	QSVC	VQC
Model type	Classical ML	Classical ML	Quantum ML	Quantum ML
Full form	Random Forest Classifier	Extreme Gradient Boosting	Quantum Support Vector Classifier	Variational Quantum Classifier
Core technique	Multiple decision trees	Gradient boosting	Quantum kernel and SVM	Parameterized quantum circuit
Feature processing	Preprocessing	Preprocessing	PCA (4 components)	PCA (4 components)
Feature encoding	Not required	Not required	ZZFeatureMap	Quantum feature map

Feature	Random Forest	XGBoost	QSVC	VQC
Training method	Ensemble learning	Sequential tree optimization	Quantum kernel learning	Circuit parameter optimization
Prediction method	Majority voting / probability aggregation	Combined tree predictions	Quantum kernel similarity	Quantum circuit measurements
Explainability	SHAP TreeExplainer	SHAP TreeExplainer	SHAP KernelExplainer	SHAP KernelExplainer
Accuracy	89.25%	90.50%	82.50%	78.00%
F1-score	0.8510	0.8651	0.7980	0.7420
ROC-AUC	0.9240	0.9336	0.8410	0.7950
Inference latency	0.0450 ms	0.0062 ms	18.20 ms	24.50 ms
Main advantage	Robust classification	High predictive performance	Quantum kernel-based classification	Trainable quantum circuit
Main limitation	Depends on training data	Requires hyperparameter tuning	Quantum kernel computational cost	Circuit training and simulation cost

5. RESULTS

5.1 Experimental results

The performance of the Quantum-Powered Cyber Attack Detection and Prevention System was evaluated using four classification models: Random Forest, XGBoost, Quantum Support Vector Classifier (QSVC), and Variational Quantum Classifier (VQC). The evaluation focused on classification accuracy, F1-score, ROC-AUC, and inference latency (Table 4). These metrics were used to compare the predictive performance and computational responsiveness of classical and quantum machine-learning approaches.

The experimental results indicate that XGBoost achieved the highest accuracy of 90.50%, an F1-score of 0.8651, and a ROC-AUC of 0.9336. Random Forest achieved an accuracy of 89.25%, an F1-score of 0.8510, and a ROC-AUC of 0.9240. These results demonstrate that both classical models provided strong classification performance in the evaluated environment, with XGBoost achieving the best overall results among the four models.

Among the quantum machine-learning models, QSVC achieved an accuracy of 82.50%, an F1-score of 0.7980, and a ROC-AUC of 0.8410. VQC achieved an accuracy of 78.00%, an F1-score of 0.7420, and a ROC-AUC of 0.7950. Although both quantum models demonstrated the ability to classify the evaluated data, their measured performance was lower than that of the classical classifiers in these experiments.

Table 4. Performance evaluation of classical and quantum machine learning models

Model	Accuracy (%)	F1-score	ROC-AUC	Inference latency (ms)
Random Forest	89.25	0.8510	0.9240	0.0450
XGBoost	90.50	0.8651	0.9336	0.0062
QSVC	82.50	0.7980	0.8410	18.20

Model	Accuracy (%)	F1-score	ROC-AUC	Inference latency (ms)
VQC	78.00	0.7420	0.7950	24.50

5.2 Comparison of classification performance

The comparison shows that XGBoost outperformed the other evaluated models in accuracy, F1-score, and ROC-AUC. Its accuracy exceeded that of Random Forest by 1.25 percentage points, QSVC by 8.00 percentage points, and VQC by 12.50 percentage points. Random Forest also demonstrated competitive performance, with an accuracy of 89.25%, indicating that classical ensemble-based methods remain effective for the tested cyberattack detection tasks (Table 5).

The quantum classifiers produced lower accuracy and F1-scores than the classical models. QSVC performed better than VQC across all three reported classification metrics. The difference between the two quantum models suggests that the choice of quantum classification method can influence predictive performance, even when both models operate within a similar preprocessing framework.These findings indicate that quantum machine learning can be incorporated into a cyberattack detection framework and evaluated alongside established classical methods. However, the results do not demonstrate a predictive-performance advantage for the quantum classifiers under the tested conditions. Further experiments involving larger datasets, different feature representations, alternative circuit configurations, and noisy quantum simulations would be necessary to determine how their performance changes under different conditions.

Table 5. Comparative analysis of classical and quantum machine learning models for cyber attack detection

Comparison parameter	Random Forest	XGBoost	QSVC	VQC
Classification approach	Bagging-based ensemble	Boosting-based ensemble	Quantum kernel-based classification	Variational quantum circuit
Accuracy (%)	89.25	90.50	82.50	78.00
F1-score	0.8510	0.8651	0.7980	0.7420
ROC-AUC	0.9240	0.9336	0.8410	0.7950
Relative performance in tested setup	High	Highest	Moderate	Lowest among evaluated models
Quantum computation required	No	No	Yes, for quantum kernel evaluation	Yes, for circuit execution and training

5.3 Inference latency analysis

Inference latency was measured to examine the computational responsiveness of each classifier. XGBoost recorded the lowest reported latency of 0.0062 ms, followed by Random Forest at 0.0450 ms. QSVC recorded a latency of 18.20 ms, while VQC recorded 24.50 ms.

The classical models exhibited substantially lower reported inference latency than the quantum models. XGBoost was the fastest model in the reported measurements, while VQC recorded the highest latency. The higher latency of the quantum classifiers may be associated with quantum feature encoding, kernel computation, circuit execution, and simulation overhead, depending on the implementation and measurement setup.

The latency results are particularly relevant to cybersecurity applications, where timely classification can support rapid alert generation and threat response. Nevertheless, the reported latency values should be interpreted in the context of the experimental environment. A fair comparison requires consistent timing procedures and clarification of whether preprocessing, quantum circuit execution, kernel construction, and other overheads are included in the measurements.

5.4 Explainability and threat assessment

The integration of SHAP provides feature-level explanations for the predictions generated by the classification models. TreeExplainer is used for Random

Forest and XGBoost, while KernelExplainer is used for QSVC and VQC. These explanations estimate how individual features contribute to the model outputs and help users investigate the characteristics associated with a predicted class.

The system also assigns a threat score ranging from 0 to 100 and classifies the severity as LOW, MEDIUM, HIGH, or CRITICAL according to the configured scoring rules. This additional assessment helps organize detection results by their assigned risk levels and supports the prioritization of suspicious activities. The combination of classification, feature explanations, and threat scoring provides more information than a class label alone.

The usefulness of the threat score depends on the scoring method and its validation. Therefore, the score should be interpreted as a system-generated risk assessment rather than a calibrated probability of malicious activity unless a separate calibration procedure has been performed.

5.5 Alert generation and system integration

The system integrates the classification models with a response workflow that generates alerts and supports configured actions, including monitoring, event logging, rate limiting, temporary blocking, and suspicious URL blocking. VirusTotal integration can provide supplementary threat-intelligence information for suspicious URLs when the service and required API access are available. MongoDB stores detection records, explanations, and security events, while the React dashboard displays the results for monitoring.

This integration connects the model predictions with the subsequent stages of threat assessment and response management. The Flask backend coordinates requests between the dashboard, detection models, and database components. As a result, the application combines detection, explainability, risk assessment, and monitoring within a single workflow. The prevention actions demonstrated in the application should be distinguished from actual operational blocking, which requires appropriate security-system integration and authorization.

6. DISCUSSION

The experimental results demonstrate that the effectiveness of cyberattack classification depends on the model and the conditions under which it is evaluated. XGBoost achieved the strongest reported predictive performance and the lowest reported inference latency, while Random Forest provided a competitive alternative. QSVC and VQC demonstrated the feasibility of incorporating quantum machine-learning classifiers into the system, but their results were lower than those of the classical models in the reported experiments.

The findings also highlight the importance of evaluating cybersecurity models using multiple metrics. Accuracy alone does not fully describe the quality of attack detection, particularly when datasets contain different proportions of legitimate and malicious samples. Precision, recall, F1-score, and ROC-AUC provide complementary information about classification behaviour, while inference latency helps assess responsiveness.The integration of SHAP, threat scoring, alert generation, VirusTotal, MongoDB, and dashboard monitoring extends the system beyond basic classification. These components provide explanatory information, organize events by assessed severity, and support the review of detection results. However, the reliability of the overall framework depends on the quality and representativeness of the input data, the validation of the scoring mechanism, and the consistency of the evaluation procedure.

Overall, the results support the use of classical machine-learning models as the strongest-performing classifiers in the current experimental setup, while the quantum models provide a basis for further investigation into quantum-assisted cyberattack detection. Future work should examine performance on larger and more diverse datasets, evaluate the effects of quantum noise, validate the models across independent test environments, and assess the practical requirements for deployment on quantum hardware. These investigations would provide a clearer understanding of the conditions under which quantum machine learning may offer practical benefits for cybersecurity.

7. CONCLUSION AND FUTURE SCOPE

The Quantum-Powered Cyber Attack Detection and Prevention System integrates classical machine learning, quantum machine learning, explainable artificial intelligence, and threat-response mechanisms to identify and analyse potential cybersecurity threats. The system employs Random Forest, XGBoost, Quantum Support Vector Classifier (QSVC), and Variational Quantum Classifier (VQC) to classify network traffic and suspicious URLs. SHAP provides explanations for model predictions, while the threat-assessment mechanism assigns risk scores and severity levels to support security monitoring and response. The integration of Flask, React, MongoDB, and VirusTotal further supports detection processing, dashboard monitoring, record storage, and supplementary URL threat intelligence.

The experimental results demonstrate that XGBoost achieved the highest accuracy of 90.50%, an F1-score of 0.8651, and a ROC-AUC of 0.9336, followed by Random Forest with an accuracy of 89.25%. QSVC and VQC achieved accuracies of 82.50% and 78.00%, respectively. These results indicate that the classical models performed better than the evaluated quantum models under the tested conditions, while the quantum models provide a basis for further investigation of quantum machine learning in cybersecurity. By combining attack classification, explainability, threat scoring, alert generation, and configurable response actions, the system provides an integrated approach to cyberattack detection and prevention.

Future Scope:

Future improvements may include the following developments to improve detection reliability and practical applicability:

- Evaluation on larger and more diverse datasets.
- Enhanced quantum circuit designs.
- Testing under noisy quantum conditions.
- Validation in realistic network environments.

REFERENCES

[1] Danasabe, A., Afaq, M., El-Maleh, A. H., Khan, A., El-Ferik, S., & Kaleem, Z. (2026). QATNet: A lightweight quantum-classical tabular network for low-latency intrusion detection. *IEEE Transactions on Quantum Engineering, 7*, 3102015.

[2] Hdaib, M., Rajasegarar, S., & Pan, L. (2024). Quantum deep learning-based anomaly detection for enhanced network security. *Quantum Machine Intelligence, 6*(1), 26.

[3] Kukliansky, A., Orescanin, M., Bollmann, C., & Huffmire, T. (2024). Network anomaly detection using quantum neural networks on noisy quantum computers. *IEEE Transactions on Quantum Engineering, 5*, 3100611.

[4] Kalinin, M., & Krundyshev, V. (2022). Security intrusion detection using quantum machine learning techniques. *Journal of Computer Virology and Hacking Techniques, 19*(1), 125–136.

[5] Cirillo, F., Esposito, C., & Seo, J. T. (2026). Benchmarking quantum machine learning methods for intrusion detection on noisy quantum computers. *Quantum Machine Intelligence, 8*(1), 27.

[6] Adenusi, D. A., Oladimeji, O. O., Oyekola, T. A., & Olagunju, K. S. (2025). Data-driven network intrusion detection using optimized machine learning algorithms.

[7] Mohammad, A., & Mohammad, H. (2026). DEHOA-optimized deep intrusion detection with quantum neural network extensions for IoT network security.

[8] Vinotha, K., & Eswaran, P. (2026). Quantum-inspired hyperparameter optimization for enhanced deep learning-based intrusion detection in wireless sensor networks.

[9] Amara, M., Amara, M., Mnasri, S., & Val, T. (2025). QCNN-ID: A quantum-classical hybrid model for IoT intrusion detection.

[10] Kadry, H., Farouk, A., Zanaty, E. A., & Reyad, O. (2023). Intrusion detection model using optimized quantum neural network and elliptical curve cryptography for data security.

[11] Ahmed, S., Shihab, I. F., & Khokhar, A. (2025). Quantum-driven zero trust architecture with dynamic anomaly detection in 7G technology: A neural network approach.

[12] Mahesha, A., Aa, N., Shivvanii, S., Ca, V., & Manimaran, S. (2026). An explainable machine learning and deep learning framework for malicious URL detection.

[13] Dirchaoui, M., Abarda, A., & El Ouardighi, A. (2026). Detecting fraudulent URLs on Arabic social platforms using an explainable machine learning approach. *Procedia Computer Science, 280*, 1306–1311.

[14] Doshi, T., Patel, V., Shah, N., Swain, D., Swain, D., & Acharya, B. (2025). PhishHunter-XLD: An ensemble approach integrating machine learning and deep learning for phishing URL classification.

[15] Prasad, Y. B., Dondeti, V., & Dondeti, R. (2026). Phishing URL detection in real-time: Machine learning and deep learning approaches on a custom threat intelligence dataset. *Procedia Computer Science, 283*, 4859–4879.