KYBERNETIKA — VOLUME *62* (2026), NUMBER 3, PAGES 456–480

FROM API CALLS TO BEHAVIORAL GRAPHS: STATE-CLUSTERING AND MARKOV TRANSITION FEATURES FOR CLASSICAL AND QUANTUM MALWARE DETECTION

Aktham Youssef, Ivan Zelinka and Eslam Amer

This paper presents a malware-detection pipeline for Windows API-call sequences that converts variable-length traces into fixed-length behavioral features. API tokens are embedded with Word2Vec and clustered with K-means to form a shared behavioral state space, and each trace is summarized by first-order Markov transition frequencies (S^2 features). Experiments on 150,656 traces with a family-aware split (20% of families held out for testing) show that linear baselines trained on the full transition representation provide strong and scalable performance. Quantum models are evaluated under explicit computational limits by applying TruncatedSVD to low-dimensional subsets and comparing a fidelity-kernel QSVM and a variational quantum classifier against a matched classical SVC-RBF on identical stratified subsets. Beyond ROC/PR curves, security-oriented operating points are reported by selecting thresholds on validation to enforce high malware recall ($R_{\text{mal}} \geq 0.90$), highlighting the recall trade-off and the scaling cost of quantum kernel evaluation as qubit count increases.

Keywords: malware detection, API call sequences, Word2Vec, K-means, Markov transitions, quantum machine learning, QSVM, VQC

Classification: 81P68, 68M25, 68T05, 68T10

1. INTRODUCTION

Malware detection remains a practical challenge in cybersecurity. Malicious programs evolve quickly, and many samples are designed to bypass static signatures [19]. Dynamic behavioral analysis provides a complementary view: instead of relying on byte-level patterns, it focuses on how a program interacts with the operating system during execution [14]. One widely used behavioral signal is the sequence of Windows API calls, which can reveal patterns related to memory allocation, process injection, persistence, and network activity [2, 3].

API-call sequences are difficult learning objects. Their lengths vary widely, calls repeat frequently, and the same high-level behavior can be implemented through different low-level call patterns; moreover, traces can vary across runs and may be noisy under dynamic analysis [14]. These properties motivate representations that preserve local

DOI: 10.14736/kyb-2026-3-0456

sequential behavior while producing fixed-length features that standard machine learning models can handle efficiently [2].

A common strategy is to embed API tokens into a continuous space (e. g., Word2Vec) and then build fixed-length features from sequences [13]. However, raw tokens often lead to fragmented representations: semantically related APIs still appear as separate symbols, rare calls increase sparsity, and generalization to unseen variants becomes harder [2]. In operational settings, malware detection usually prioritizes reducing false negatives, which pushes decision thresholds toward high malware recall and can increase false positives; therefore, Accuracy alone can be misleading. Under class imbalance, PR curves are often more informative than ROC curves, while ROC remains useful as a threshold-sweeping summary [8, 16].

A key practical requirement is to obtain a representation that (i) scales to large datasets, (ii) reduces token-level sparsity, and (iii) retains behaviorally meaningful local structure. Markov-style transition summaries provide a principled way to encode local dynamics as fixed-length features, and have been shown effective in malware behavior modeling in other ecosystems (e. g., Markov-chain abstractions) [11]. While such transition features do not capture long-range dependencies explicitly, they offer an interpretable and scalable baseline that is suitable for large-scale benchmarking and controlled comparisons.

This paper introduces an end-to-end pipeline that converts API-call sequences into a compact behavioral alphabet and extracts first-order transition features for classification (Section 4). Word2Vec embeddings are trained on the training corpus, and K-means clustering with $K = 30$ groups related APIs into shared behavioral states. An additional `UNK` state handles out-of-vocabulary tokens, resulting in $S = 31$ discrete states. Each trace is mapped to a sequence of state indices and summarized using a fixed-length transition representation of dimension $S^2 = 961$. To reduce sensitivity to trace length, learning features are based on per-trace normalized transition frequencies.

To ensure a realistic generalization assessment, we use a family-aware split in which entire families are held out for testing, reducing leakage from closely related variants that can otherwise inflate performance [14]. This is especially important in malware detection, where near-duplicate samples and family-level similarities can cause overly optimistic results if splitting is performed at the sample level.

Quantum machine learning methods are also evaluated under controlled computational limits. Quantum kernels and variational classifiers can represent nonlinear decision boundaries using circuit-based feature maps and trainable ansätze [6, 9, 17]. At the same time, kernel evaluation requires pairwise similarities and scales quadratically with the number of training samples, which makes full-dataset experiments expensive. For this reason, a matched-subset protocol is used (Section 5). Transition features are reduced using TruncatedSVD to $D \in \{8, 12\}$ dimensions, scaled to $[0, \pi]$, and mapped to D-qubit feature maps. Quantum models (QSVM and VQC) are compared against a classical SVC-RBF baseline on identical stratified subsets. In addition to ROC and PR curves, results are reported at security-oriented operating points selected on validation to enforce a minimum malware recall requirement and to quantify the resulting trade-off in benign recall (Section 5.3).

Contributions. The main contributions are:

- A scalable representation for Windows API-call sequences that combines Word2Vec token embeddings with K-means clustering to form a shared behavioral alphabet, followed by first-order transition features suitable for large-scale learning.
- A family-aware and security-oriented evaluation protocol that complements ROC/PR analysis with validation-based operating-point selection under an explicit malware-recall constraint.
- A controlled comparison between classical baselines and quantum models (QSVM and VQC) under matched low-dimensional subset constraints, including an 8-qubit setting and a 12-qubit setting that highlights scaling limits.

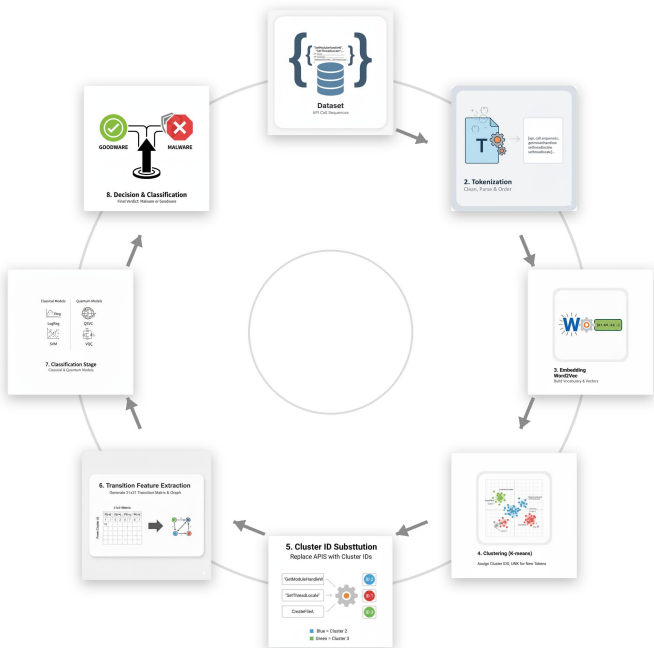


Fig. 1: End-to-end pipeline of the proposed approach.

Figure 1 summarizes the end-to-end workflow of the proposed pipeline. Starting from raw Windows API-call traces, each sample is parsed into an ordered sequence of API tokens while preserving repetitions. Word2Vec embeddings are then trained on the training corpus only, and K-means clustering (with $K = 30$ plus an `UNK` state) groups related APIs into a compact behavioral alphabet of $S = 31$ states. Each trace is mapped to a state sequence and summarized by first-order Markov transition frequency

features of fixed dimension $S^2 = 961$, which serve as input to the learning models. Classical baselines are evaluated on the full held-out test set using these transition features, while quantum models operate under matched subset constraints after TruncatedSVD reduction to $D \in \{8, 12\}$ and scaling to $[0, \pi]$ for circuit encoding.

The remainder of the paper is organized as follows. Section 2 summarizes the main notation used throughout the paper. Section 3 reviews related work and summarizes the QML concepts used in this study. Section 4 details dataset preprocessing, shared clustering, and transition-feature construction. Section 5 describes the experimental protocols, subset construction, and evaluation metrics (including the security-oriented operating-point rule). Section 6 reports the empirical results, and Section 7 concludes with limitations and future directions.

2. PRELIMINARY NOTATIONS

An API-call trace is represented as an ordered sequence $\mathbf{x} = (a_1, a_2, \ldots, a_L)$, where a_t is the API token at position t and L is the sequence length. The indicator function $\mathbb{I}[\cdot]$ equals 1 when its argument is true and 0 otherwise.

After Word2Vec training on the training corpus, each token a is mapped to an embedding $\mathbf{v}(a) \in \mathbb{R}^d$. The embedding space is partitioned into K clusters using K-means, producing a shared set of discrete states. Each token is mapped to a cluster index by $c(a) \in \{0, \ldots, K-1\}$, while out-of-vocabulary tokens are mapped to a dedicated `UNK` state. With `UNK`, the total number of states is $S = K + 1$.

A trace is then mapped to a state sequence $\mathbf{s} = (s_1, \ldots, s_L)$ with $s_t \in \{0, \ldots, S-1\}$. First-order transition counts are defined as

$$T_{ij}(\mathbf{s}) = \sum_{t=1}^{L-1} \mathbb{I}[s_t = i \wedge s_{t+1} = j], \qquad i, j \in \{0, \ldots, S-1\}.$$

For learning, transition counts are converted to per-trace transition frequencies by normalizing with the number of within-trace transitions:

$$F_{ij}(\mathbf{s}) = \frac{T_{ij}(\mathbf{s})}{\max(L-1, 1)}.$$

When $L > 1$, $\sum_{i,j} F_{ij}(\mathbf{s}) = 1$; when $L \leq 1$, the feature vector becomes the all-zero vector. The feature representation is the vectorization $\boldsymbol{\phi}(\mathbf{s}) = \mathrm{vec}(F(\mathbf{s})) \in \mathbb{R}^{S^2}$. In the implementation, the matrix entry (i, j) is stored at a single index $\mathrm{idx} = i \cdot S + j$.

For quantum experiments, $\boldsymbol{\phi}(\mathbf{s})$ is reduced to D dimensions using TruncatedSVD, producing $\mathbf{x} \in \mathbb{R}^D$, and each component is scaled to $[0, \pi]$ before encoding into a D-qubit feature map. Given a quantum feature map $U_\Phi(\cdot)$, the embedded state is $|\Phi(\tilde{\mathbf{x}})\rangle = U_\Phi(\tilde{\mathbf{x}})|0\rangle^{\otimes D}$, and the QSVM kernel is defined by the state fidelity $k_Q(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = |\langle \Phi(\tilde{\mathbf{x}}) \,|\, \Phi(\tilde{\mathbf{x}}') \rangle|^2$.

3. STATE OF THE ART

3.1. Behavioral malware analysis via API call sequences

Behavior-based malware analysis commonly relies on dynamic execution traces, where sequences of operating-system API calls serve as compact behavioral signatures [1]. Compared to purely static features, API-call sequences can better reflect intent (e. g., process injection, persistence, credential access) and tend to be more resilient to packing and superficial binary transformations. At the same time, API traces are noisy and may vary across executions; adversarial evasion can also perturb call order or inject benign-looking calls. As a result, effective learning methods must cope with variability, context dependence, and class imbalance.

Deep sequential modeling is a representative direction. Çatak et al. show that recurrent architectures can learn discriminative temporal patterns directly from Windows API-call sequences with limited feature engineering [20]. A complementary line emphasizes contextual modeling of local neighborhoods and call order. Amer et al. propose dynamic detection based on contextual understanding of API-call sequences to improve detection and prediction in Windows settings [2]. Public datasets derived from sandbox execution support reproducible benchmarking; for example, Mal-API-2019 provides labeled Windows API-call sequences for experimentation [20].

Beyond direct sequence classification, several recurring research themes are important for positioning: (i) representation learning for APIs (tokens and subsequences), (ii) robustness to perturbations and adversarial manipulation, (iii) generalization to previously unseen families, and (iv) interpretability of behavioral motifs that drive a decision.

3.2. From engineered features to representation learning in malware traces

Earlier work explored engineered sequence statistics and hybrid objectives that combine classification with auxiliary behavioral summarization. Wang and Yiu propose a multi-task seq2seq-style model for API-call sequences, using an auxiliary decoder to predict file-access patterns alongside malware labels, improving interpretability while maintaining competitive performance [10]. More recent approaches strengthen the representation layer using embeddings and attention mechanisms. Huang et al. learn malware representations from execution sequences using embedding components and attention to highlight informative events [7].

Markov-chain abstractions have also proven effective in malware analysis. MAMADroid models sequences through Markov transitions and demonstrates that behavioral abstraction can improve robustness and interpretability [11]. Although Android and Windows differ in API ecosystems, the underlying methodological takeaways generalize: (a) abstraction can reduce fragmentation, (b) transition structure can encode meaningful behavior, and (c) evaluation should reflect realistic execution variance.

3.3. Quantum machine learning foundations relevant to sequence-derived features

Quantum machine learning (QML) in the NISQ era is typically implemented as a hybrid pipeline. Classical preprocessing first produces a feature vector (e. g., from bytes, op-

codes, or API-call traces), which is then encoded into a parameterized quantum circuit (PQC) used either as (i) a variational classifier or (ii) a kernel feature map. Havlíček et al. provide a seminal example of quantum feature maps that can induce complex decision boundaries [9]. From a learning-theoretic perspective, many supervised QML models can be interpreted through kernels; Schuld clarifies this connection and discusses when quantum models reduce to, match, or generalize classical kernels [17].

Variational quantum algorithms (VQAs) are a practical foundation for classifier design. Cerezo et al. review variational circuit families, training procedures, and applications, which is directly relevant when a PQC is used as a trainable classifier component [6]. In malware classification, this typically leads to two integration strategies: (1) variational classifiers operating on trace-derived feature vectors, and (2) kernel methods comparing samples via circuit-defined similarity.

3.4. Trainability and scalability challenges in NISQ QML

Despite their promise, NISQ QML models face well-known limitations. McClean et al. describe *barren plateaus*, where gradients vanish exponentially in circuit depth or qubit count, making variational optimization difficult [12]. In the kernel setting, Thanasilp et al. discuss *exponential concentration*, where kernel values can become uninformative under measurement constraints, potentially requiring large sampling budgets to estimate similarities reliably [18].

These challenges motivate careful design choices for cybersecurity-oriented QML: shallow or structured ansätze, problem-informed encodings, and clear reporting of measurement budgets and noise assumptions. When inputs are high dimensional—as is common for sequence-derived representations—encoding strategies and circuit expressivity must be balanced against trainability and sampling cost.

3.5. Existing QML studies in malware and cybersecurity contexts

QML studies for malware classification exist but remain less common than classical deep learning approaches. Barrué and Quertier compare quantum machine learning models to classical baselines for malware classification and discuss optimization issues and research directions [4]. Across related studies, recurring practical constraints include limited qubit counts (necessitating feature reduction) and sensitivity to noise and shot budgets.

In broader cybersecurity contexts, Bellante et al. evaluate QML potential via a case study on intrusion detection systems (IDS) and propose criteria for assessing when quantum advantage might plausibly emerge under realistic workloads [5]. While IDS and malware classification differ, the emphasis on workload realism and constraint-aware evaluation transfers directly to malware settings.

3.6. Implementation ecosystem and reproducibility

Reproducible QML experimentation depends heavily on tooling and documentation. Qiskit Machine Learning provides an open-source workflow for kernels, variational models, and integration with classical pipelines [15]. In malware research, practical reproducibility usually requires: (i) releasing feature-extraction and preprocessing scripts, (ii)

documenting trace normalization choices and any truncation policies, (iii) specifying encoding maps and qubit counts precisely, and (iv) reporting measurement/noise settings when experiments move beyond statevector simulation.

Summary. API-sequence malware detection is widely explored using deep sequential models and representation learning [2, 7, 10, 20], while behavioral abstraction via transitions can improve robustness and interpretability [11]. QML offers two common routes—variational classifiers and quantum kernels [6, 9, 17] — but must contend with trainability and sampling barriers in the NISQ regime [12, 18]. Existing QML work in malware and cybersecurity highlights constraint-aware protocols and careful evaluation [4, 5].

Positioning. Prior work on API-sequence malware detection spans engineered sequence statistics (e. g., n-grams/HMM/Markov abstractions) and deep sequential models, alongside representation-learning approaches such as Word2Vec-style token embeddings and behavioral abstractions. On the quantum side, QML-for-security studies often focus on quantum kernels (QSVM) and variational classifiers (VQC) under tight qubit and sampling constraints, where scalability and trainability remain key concerns. In contrast to evaluation practices that rely mainly on accuracy, performance is reported here at explicit security-oriented operating points: thresholds are selected on validation to enforce high malware recall ($R_{\text{mal}} \geq \tau$, with $\tau = 0.90$) and the resulting trade-off in benign recall (R_{good}) is quantified, reflecting deployment-oriented requirements.

Table 1 provides a simplified comparison of representative related work and summarizes how the proposed pipeline differs in representation and evaluation protocol.

4. METHODOLOGY

4.1. Dataset and preprocessing

Each sample is provided as a plain-text file containing a Windows API-call trace written as a brace-delimited list of string tokens (e. g., {`‘‘GetModuleHandleW",` `‘‘SetThreadLocale",...`}). Although the braces resemble a set notation, we parse the file and treat the content as an ordered sequence in the same order as recorded in the trace, preserving repetitions, because our feature extraction depends on adjacent calls. The dataset contains $N = 150{,}656$ samples (109,017 malware and 41,639 goodware) and is processed in 16 chunks for scalable I/O. A sample is denoted by $\mathbf{x} = (a_1, a_2, \ldots, a_L)$, where a_t is the API token at position t and L is the sequence length.

Family-aware split. To reduce leakage from closely related variants, a family-aware split is used, where 20% of families are held out for testing (test size = 0.2). This yields 121,256 training samples (88,345 malware; 32,911 goodware) and 29,400 test samples (20,672 malware; 8,728 goodware). Family identifiers are derived from the dataset folder structure, and the split is performed at the family level within each class to preserve class balance.

Work	**Domain**	**Representation**	**Model**	**Key emphasis / note**
[20]	Windows API	Raw API sequences	RNN-based deep model	End-to-end sequential modeling of API-call traces.
[2]	Windows API	Contextual sequence modeling	ML/DL contextual approach	Emphasizes contextual neighborhoods and call-order information.
[10]	API sequences	Sequence + auxiliary behavior	Multi-task seq2seq	Adds auxiliary outputs to support behavioral interpretation.
[7]	Execution sequences	Learned embeddings + attention	Attention-based model	Learns representations and highlights informative events in sequences.
[11]	Android calls	Markov-chain abstraction	Classical ML on Markov features	Demonstrates abstraction + transitions; platform differs but methodology is transferable.
[4]	Malware (QML)	Reduced classical features	QML (QSVM/VQC variants)	QML feasibility under limited qubits; relies on feature reduction and constrained regimes.
[5]	IDS (QML)	Tabular / engineered features	QML evaluation methodology	Focus on realism and constraint-aware evaluation; different task domain.
This work	Windows API	Word2Vec→KMeans states + Markov transitions ($S^2 = 961$); SVD for QML	LR/Linear SVM (full) + QSVM/VQC (matched subsets)	Family-aware split and security-oriented operating points ($R_{\text{mal}} \geq 0.90$) with matched quantum–classical subsets.

Tab. 1: Simplified comparison of representative related work and the positioning of this paper.

Sequence-length characteristics. API sequences show substantial length variability, motivating a fixed-length representation. Figure 2 reports the length distribution and highlights a long tail (max $L = 3029$), which makes sequence-length–dependent models less convenient for large-scale benchmarking.

4.2. Shared clustering of API Tokens

The goal is to map heterogeneous API tokens into a shared discrete alphabet of behavioral states. Dense API embeddings are learned using Word2Vec trained on the training

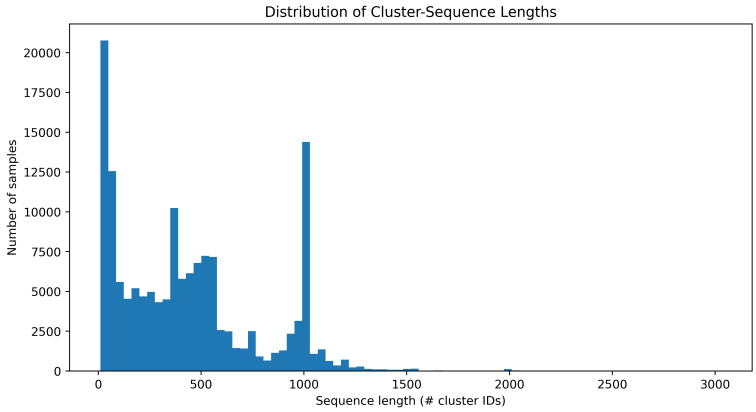


Fig. 2: Distribution of API-call sequence lengths across all samples. The long tail (max $L = 3029$) motivates a fixed-length transition-based representation.

corpus only. Each API token a is represented by a vector $\mathbf{v}(a) \in \mathbb{R}^d$ with embedding dimension $d = 128$. The resulting vocabulary contains 8,798 unique API tokens. In the implementation, Skip-gram Word2Vec is used with window size 6, `min_count`=2, and 10 training epochs (fixed seed for reproducibility).

K-means clustering. The learned embedding vectors are clustered using K-means with $K = 30$ to obtain centroids $\{\boldsymbol{\mu}_k\}_{k=0}^{K-1}$ and an assignment function:

$$c(a) = \arg \min_{k \in \{0,\ldots,K-1\}} \|\mathbf{v}(a) - \boldsymbol{\mu}_k\|_2^2. \tag{1}$$

Before clustering, embedding vectors are ℓ_2-normalized so that Euclidean K-means better reflects cosine-based similarity in the embedding space.

Choice of $K = 30$ (expressivity vs. robustness). The number of clusters K controls the granularity of the shared behavioral alphabet. A smaller K yields a coarser abstraction that may merge distinct behaviors, whereas a larger K increases expressivity but can fragment the space and become sensitive to rare APIs and clustering noise. In this study, $K = 30$ is used as a practical trade-off that preserves behavioral diversity while keeping the state space compact. With an additional `UNK` state, this yields $S = K + 1 = 31$ states and a fixed transition feature dimension of $S^2 = 961$, which remains computationally manageable for full-data classical baselines and provides a stable input for subsequent dimensionality reduction in the quantum experiments. A systematic ablation over multiple K values is left for future work.

Sensitivity to the number of clusters K. The choice of K controls the granularity of the shared behavioral state space and therefore affects both feature sparsity and model capacity. A smaller K produces a coarser abstraction that can merge distinct behaviors,

whereas a larger K increases expressivity but may fragment the representation and amplify noise from rare APIs. In this study, $K = 30$ was selected as a practical trade-off that keeps the state space compact ($S = 31$ with `UNK`) while retaining sufficient behavioral diversity for large-scale training and for subsequent SVD-based reduction in the quantum experiments. A systematic sensitivity study over multiple K values is beyond the scope of this revision, as it requires re-running the embedding, clustering, and downstream evaluation pipeline for each K under the same family-aware split and operating-point protocol.

Cluster-sequence representation. Each API sequence $\mathbf{x} = (a_1, \ldots, a_L)$ is converted to a cluster sequence $\mathbf{s} = (s_1, \ldots, s_L)$ by:

$$s_t = c(a_t), \qquad t = 1, \ldots, L. \tag{2}$$

Tokens not present in the Word2Vec vocabulary are mapped to a dedicated unknown state `UNK`. In the implementation, `UNK` is assigned index 30, yielding $S = K + 1 = 31$ discrete states.

4.3. First-order Markov transition frequency features

A first-order Markov *feature representation* is used to summarize local dynamics of the clustered state sequence $\mathbf{s}$. Here, "Markov" refers to using *adjacent* state pairs (s_t, s_{t+1}) (first-order dependence) to build fixed-length features; we do *not* assume that traces are generated by an exact Markov process, nor do we require a row-stochastic transition matrix for learning. Instead, the learning representation is a *globally normalized transition-frequency matrix* that captures how often each state-to-state transition occurs within a trace.

Transition counts. For a given sample, transition counts between consecutive states are computed as:

$$T_{ij}(\mathbf{s}) = \sum_{t=1}^{L-1} \mathbb{I}[s_t = i \ \wedge \ s_{t+1} = j], \qquad i, j \in \{0, \ldots, S-1\}. \tag{3}$$

Normalized transition-frequency features (used for learning). To reduce sensitivity to sequence length, counts are normalized by the total number of within-trace transitions:

$$F_{ij}(\mathbf{s}) = \begin{cases} \frac{T_{ij}(\mathbf{s})}{L-1}, & L > 1, \\ 0, & L \leq 1, \end{cases} \tag{4}$$

so that for non-degenerate traces ($L > 1$) the features satisfy $\sum_{i,j} F_{ij}(\mathbf{s}) = 1$. A fixed-length feature vector is constructed by flattening the $S \times S$ matrix:

$$\boldsymbol{\phi}(\mathbf{s}) = \mathrm{vec}(F(\mathbf{s})) \in \mathbb{R}^{S^2}. \tag{5}$$

With $S = 31$, the feature dimension is $S^2 = 961$.

Limitation: local vs. long-range dependencies. The proposed representation is based on first-order (adjacent) state pairs and therefore captures only local transition statistics. Dependencies that span multiple steps in the trace (e. g., motifs that require longer context) are not explicitly modeled. Potential extensions include higher-order transition features (e. g., n-step or kth order transitions) or sequence models such as RNNs/Transformers applied to the token or state sequence to capture longer-range structure; these directions are left for future work.

Why fewer than $S = 31$ states may appear in plots. Although the global state space is fixed to $S = 31$ (clusters $0 \dots 29$ plus `UNK` at index 30), a single trace typically visits only a subset of states. Therefore, cropped visualizations that show only active states may display fewer nodes, while the underlying feature representation remains 31×31.

Sparsity of the transition-frequency features. Although the feature space has dimension $S^2 = 961$, each sample activates only a subset of transitions. Figure 3 reports the distribution of the number of nonzero entries (nnz) in $\boldsymbol{\phi}(\mathbf{s})$.

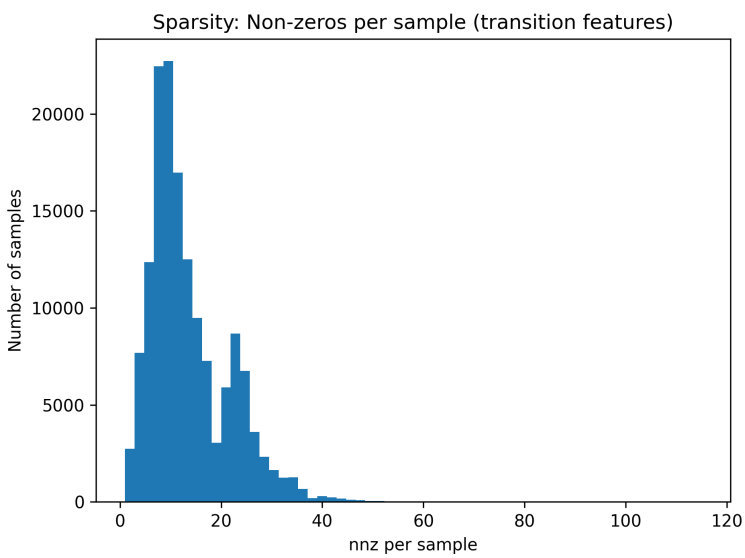


Fig. 3: Distribution of the number of nonzero transition entries (nnz) per sample in the $S^2 = 961$-dimensional transition-frequency representation $\boldsymbol{\phi}(\mathbf{s})$.

Row-normalized transitions (visualization only). For heatmap visualization, transition counts can also be normalized row-wise:

$$P_{ij}(\mathbf{s}) = \frac{T_{ij}(\mathbf{s})}{\sum_{j'} T_{ij'}(\mathbf{s}) + \epsilon}, \tag{6}$$

where $\epsilon > 0$ avoids division by zero for states that do not appear as a source in a given trace. This row-normalized form is used only for visualization; learning models operate on the globally normalized transition-frequency features in Eqs. 4–5.

Illustration of transition structure. Figure 4 shows a cropped view of transition structure for representative goodware and malware samples using Eq. 6.

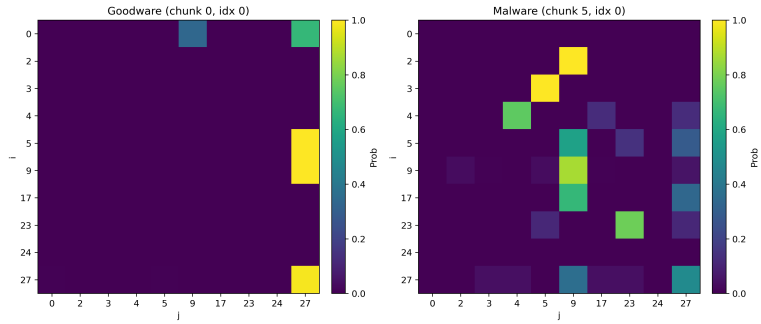


Fig. 4: Illustrative transition structure (cropped to the most active states) for representative goodware vs. malware samples. Color intensity reflects row-normalized transition strength based on Eq. 6.

Graph view of transitions (simplified Markov graphs). Transitions can also be visualized as a directed graph (Figure 5). Nodes correspond to visited states and directed edges indicate nonzero transitions. For readability, the visualization keeps only dominant outgoing transitions per node and removes isolated nodes after filtering, so the displayed graph may contain fewer nodes than $S = 31$.

4.4. Learning models

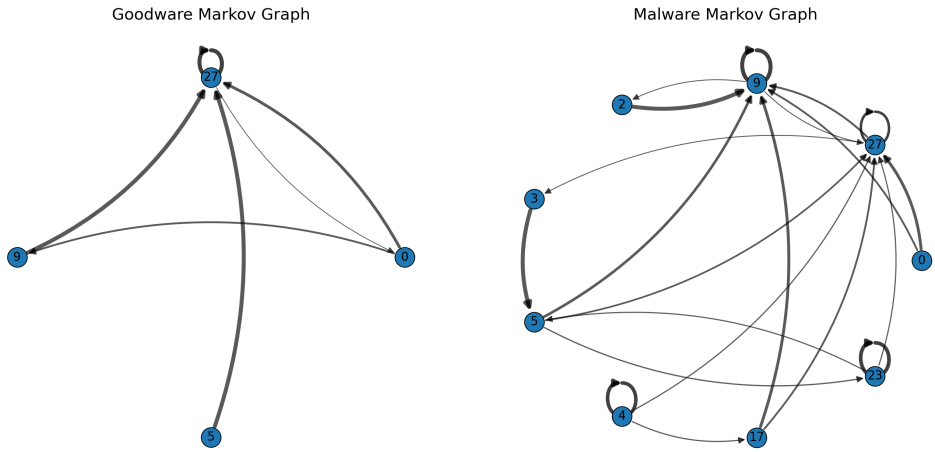
Both classical and quantum learning methods are evaluated. All models learn from the fixed-length transition representation $\boldsymbol{\phi}(\mathbf{s}) \in \mathbb{R}^{961}$ (Eq. 5). Because quantum kernel estimation scales quadratically with the number of training points, dimensionality reduction and subset-based evaluation are used for quantum experiments.

4.4.1. Dimensionality reduction for quantum evaluation

Let $\mathbf{z} = \boldsymbol{\phi}(\mathbf{s}) \in \mathbb{R}^{961}$ be the transition feature vector. TruncatedSVD is applied to obtain a low-dimensional embedding:

$$\mathbf{x} = \mathbf{W}^{\top}\mathbf{z}, \qquad \mathbf{W} \in \mathbb{R}^{961\times D}, \quad D \in \{8, 12\}, \tag{7}$$

where $\mathbf{W}$ contains the top-D right-singular vectors estimated from the training set. The resulting $\mathbf{x} \in \mathbb{R}^{D}$ is scaled to a range suitable for quantum feature maps using MinMax scaling to $[0, \pi]$. Scaling parameters are fit on the training split and reused for validation/test.



(a) Goodware simplified transition graph. (b) Malware simplified transition graph.

Fig. 5: Simplified directed graphs derived from per-sample transitions. Nodes are state indices and edges indicate dominant transitions after filtering for visualization.

4.4.2. Classical Baselines on Transition Features

As full-data baselines, Logistic Regression and Linear SVM are trained on $\boldsymbol{\phi}(\mathbf{s})$. For a linear classifier with weights $\mathbf{w}$ and bias b, the decision function is:

$$f(\boldsymbol{\phi}) = \mathbf{w}^\top \boldsymbol{\phi} + b, \tag{8}$$

and the predicted label is $\hat{y} = \mathbb{I}[f(\boldsymbol{\phi}) \geq t]$, where t is selected using the operating-point protocol in Section 5.3.

4.4.3. Kernel SVM and Quantum Kernel SVM (QSVM)

Kernel SVMs learn a decision function of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \, k(\mathbf{x}_i, \mathbf{x}) + b, \tag{9}$$

where $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ are training points, α_i are learned coefficients, and $k(\cdot, \cdot)$ is a positive semi-definite kernel.

Classical RBF kernel. For classical subset comparisons, the Gaussian RBF kernel is used:

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\!\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2\right). \tag{10}$$

Quantum fidelity kernel. In QSVM, a kernel is defined using a quantum feature map circuit $U_\Phi(\cdot)$. Given an input $\tilde{\mathbf{x}} \in [0,\pi]^D$, a quantum state is prepared. Here, $\mathbf{x} \in \mathbb{R}^D$ denotes the reduced feature vector after TruncatedSVD, and $\tilde{\mathbf{x}} \in [0,\pi]^D$ is the same vector after MinMax scaling for quantum encoding. $U_\Phi(\tilde{\mathbf{x}})$ is the data-dependent quantum feature map acting on D qubits, and $|0\rangle^{\otimes D}$ is the D-qubit all-zero initial state.

$$|\Phi(\tilde{\mathbf{x}})\rangle = U_\Phi(\tilde{\mathbf{x}})\,|0\rangle^{\otimes D}. \tag{11}$$

The kernel is defined as the state fidelity:

$$k_Q(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \left|\langle \Phi(\tilde{\mathbf{x}}) \,|\, \Phi(\tilde{\mathbf{x}}')\rangle\right|^2. \tag{12}$$

The feature map uses `ZZFeatureMap` with one repetition (reps=1) and linear entanglement, matching the D-qubit input dimension. Gram matrices are computed and supplied to an SVM using a precomputed kernel.

QSVM implementation details. QSVM experiments use a statevector simulator, which provides exact fidelity values without finite-shot sampling noise. Reported QSVM results therefore reflect deterministic kernel values for the chosen feature map and subsets.

4.4.4. Variational Quantum Classifier (VQC)

VQC optimizes a parameterized quantum circuit to separate the classes in the embedded space. Given an encoded state $|\Phi(\tilde{\mathbf{x}})\rangle$, a trainable ansatz $U(\boldsymbol{\theta})$ is applied. In VQC, $U(\boldsymbol{\theta})$ denotes a trainable variational ansatz (a parameterized quantum circuit) applied after the fixed feature-map state $|\Phi(\tilde{\mathbf{x}})\rangle$ to produce the model state $|\psi(\tilde{\mathbf{x}};\boldsymbol{\theta})\rangle$ used for classification:

$$|\psi(\tilde{\mathbf{x}};\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|\Phi(\tilde{\mathbf{x}})\rangle. \tag{13}$$

A measurement produces a real-valued score that is used for classification:

$$g(\tilde{\mathbf{x}};\boldsymbol{\theta}) = \langle\psi(\tilde{\mathbf{x}};\boldsymbol{\theta})|\,\hat{M}\,|\psi(\tilde{\mathbf{x}};\boldsymbol{\theta})\rangle. \tag{14}$$

In the implementation, training minimizes cross-entropy loss using the COBYLA optimizer with a fixed iteration budget (maxiter=80). VQC results are reported for the 8-dimensional setting with train $n = 300$, validation $n = 80$, and test $n = 150$.

Reproducibility. Classical models use scikit-learn and quantum models use Qiskit Machine Learning. Subset construction is stratified with a fixed random seed (seed=1337), and the same subset indices are reused for matched QSVM vs. classical SVC-RBF comparisons.

5. EXPERIMENTAL SETUP

5.1. Data splits and evaluation subsets

A family-aware split is used as described in Section 4, with 20% of families held out for testing. This produces 121,256 training samples and 29,400 test samples. Full-data

classical baselines are trained on the complete training split and evaluated on the full held-out test set.

Quantum-kernel methods require pairwise fidelity estimation and therefore scale quadratically with the number of training points. To enable controlled quantum–classical comparisons under identical data constraints, QSVM and a matched classical SVC-RBF baseline are evaluated on stratified subsets after dimensionality reduction.

Practical scalability of quantum models. QSVM kernel evaluation requires building Gram matrices and therefore entails $O(n^2)$ kernel computations in the number of training samples, in addition to the $O(mn)$ cost for evaluating a test set of size m against n training points. This quadratic scaling quickly becomes the dominant bottleneck as the subset size grows. Moreover, increasing the feature-map dimension (and thus the number of qubits) further increases the per-kernel evaluation cost, which limits the feasible regimes for kernel-based quantum learning on large cybersecurity datasets. For this reason, quantum results are reported under a matched-subset protocol: QSVM is compared against a classical SVC-RBF baseline on identical stratified subsets and on the same reduced features. The 12-dimensional setting is included primarily to illustrate scaling and stability constraints, whereas threshold-free ROC/PR comparisons are emphasized in the 8-dimensional setting where they remain feasible.

Data leakage prevention. To reduce leakage from closely related variants, all samples sharing the same family identifier are assigned to a single split (train or test). Dimensionality reduction (TruncatedSVD) and feature scaling (MinMax to $[0, \pi]$) are fitted on the training split only and then reused for validation/test. No hyperparameter tuning is performed beyond the fixed settings in Table 2; only the operating threshold $t^\star$ is selected on the validation split (Eq. 18). The held-out test set is used once for final reporting.

8-dimensional (8-qubit) matched subset. After TruncatedSVD to $D = 8$ dimensions (Eq. 7) and scaling to $[0, \pi]$, stratified subsets are constructed using a fixed seed (seed=1337): train $n = 200$, validation $n = 80$, and test $n = 150$. The same subset indices are reused for QSVM and classical SVC-RBF to ensure a matched comparison.

12-dimensional (12-qubit) subset. A higher-dimensional setting is also evaluated by reducing to $D = 12$ dimensions and constructing a larger stratified subset: train $n = 600$, validation $n = 200$, and test $n = 400$. Due to the higher cost of 12-qubit kernel computation, this setting is reported mainly through operating-point results rather than full ROC/PR curves.

VQC subset. Variational training is typically more stable with more training samples. VQC is trained in the $D = 8$ setting using train $n = 300$, validation $n = 80$, and test $n = 150$, while using the same evaluation protocol described below.

5.2. Models compared

The evaluated models are: (i) Logistic Regression and Linear SVM on the full Markov transition representation of dimension $S^2 = 31^2 = 961$, (ii) Classical SVC-RBF on reduced features ($D \in \{8, 12\}$), (iii) QSVM using a fidelity-based quantum kernel with `ZZFeatureMap` (reps=1), and (iv) VQC trained with a parameterized circuit and the COBYLA optimizer (maxiter fixed). For subset experiments, the classical SVC-RBF uses the same reduced features and the same subset indices as QSVM.

Hyperparameter specification. Table 2 summarizes key hyperparameters. Unless stated otherwise, standard library defaults are used (scikit-learn for classical models and Qiskit Machine Learning for quantum models). All hyperparameters are fixed before test evaluation, and operating thresholds are selected on validation only according to Eq. 18.

Model	**Key hyperparameters**	**Value / Note**
Logistic Regression (full)	C, penalty, solver, max_iter	$C = 1.0$, L2, `lbfgs`, max_iter=2000
Linear SVM (full)	C, loss, max_iter	$C = 1.0$, `squared_hinge`, max_iter=1000
SVC-RBF (subset)	C, γ	$C = 1.0$, $\gamma =$ `scale`
QSVM (precomputed-kernel SVC)	C	$C = 1.0$
Quantum feature map (QSVM/VQC)	map, reps, entanglement	`ZZFeatureMap`, reps=1, entanglement=`linear`
VQC	ansatz, optimizer	`RealAmplitudes` (reps reported per experiment), COBYLA (maxiter=80)

Tab. 2: Key hyperparameters used in the experiments.

5.3. Evaluation metrics

Both threshold-free and thresholded metrics are reported.

Threshold-free metrics. Where feasible, ROC and Precision–Recall (PR) curves are reported on test sets, together with ROC-AUC and average precision (AP). ROC summarizes the trade-off between true positive rate (TPR) and false positive rate (FPR) across thresholds, while PR highlights precision–recall behavior under class imbalance.

Malware is treated as the positive class. Given true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (\text{malware recall } R_{\mathrm{mal}}), \tag{15}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \quad (\text{goodware recall } R_{\mathrm{good}}), \tag{16}$$

$$\mathrm{FPR} = 1 - \mathrm{TNR}, \qquad \mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{17}$$

Security-oriented operating points (threshold selection). In deployment-oriented malware detection, false negatives are typically more costly than false positives. In addition to ROC/PR, an operating threshold $t^\star$ is selected on validation by enforcing a minimum malware recall constraint:

$$t^\star = \arg\max_{t \in \mathcal{T}} R_{\mathrm{good}}(t) \quad \text{subject to} \quad R_{\mathrm{mal}}(t) \geq \tau, \tag{18}$$

where $\tau = 0.90$ and $\mathcal{T}$ is a finite threshold grid scanned over validation scores. In the implementation, $\mathcal{T}$ is constructed from score quantiles (199 thresholds from the 0.01 to 0.99 quantiles). The selected threshold $t^\star$ is then applied once to the corresponding test set, and accuracy, R_{mal}, and R_{good} are reported.

5.4. Implementation notes

Subset construction is stratified and uses fixed random seeds for reproducibility. For QSVM, Gram matrices are computed in batches and supplied to an SVM with a precomputed kernel. Quantum components (QSVM/VQC) are evaluated using Qiskit Machine Learning on a statevector backend. VQC uses the COBYLA optimizer with a fixed iteration budget (maxiter=80). All experiments use the same shared clustering and transition-feature pipeline described in Section 4.

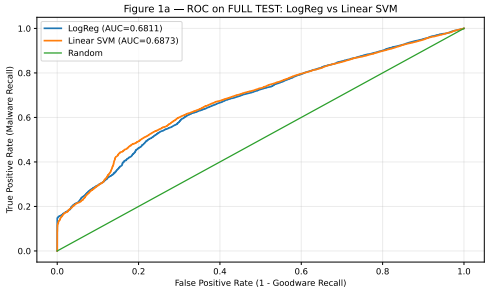
6. RESULTS

6.1. Classical baselines on the full test set

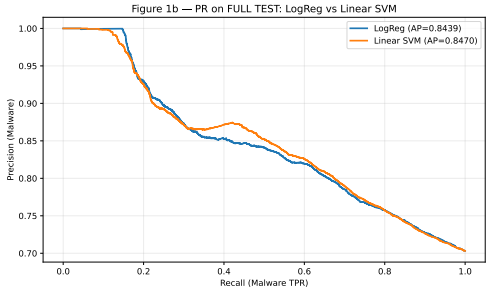
Logistic Regression and Linear SVM are evaluated on the full held-out test set of 29,400 samples (20,672 malware and 8,728 goodware) using the $S^2 = 31^2 = 961$-dimensional Markov transition representation. For each trace, transition counts are normalized by the number of within-trace transitions (Eq. 4), yielding per-trace transition frequencies that reduce sensitivity to sequence length. Threshold-free performance is summarized using ROC and Precision–Recall (PR) curves (Figure 6). PR curves are particularly informative under class imbalance, while ROC curves summarize the TPR–FPR tradeoff across decision thresholds.

6.2. Quantum vs. classical on a matched low-dimensional subset (8 qubits)

Quantum-kernel evaluation requires pairwise fidelity computation and therefore scales as $O(n^2)$ in the number of training samples. As a result, quantum models are evaluated under a controlled subset protocol rather than on the full dataset. Transition features are reduced to $D = 8$ dimensions using TruncatedSVD and mapped to an 8-qubit feature map for QSVM and VQC. Stratified subsets are constructed with a fixed seed (seed=1337): train $n = 200$, validation $n = 80$, and test $n = 150$. Figure 7 compares QSVM against a matched classical SVC-RBF baseline trained on the same reduced features and identical subset indices.

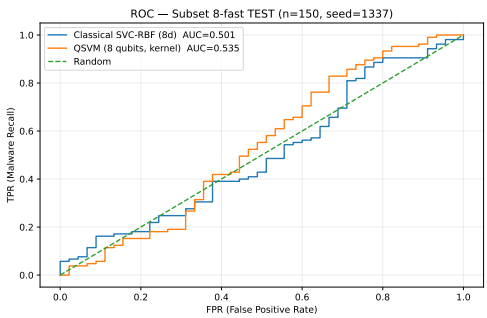


(a) ROC on full test set.

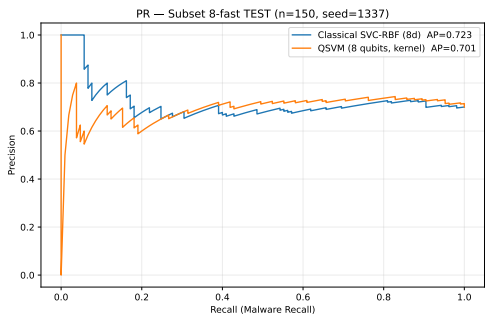


(b) PR on full test set.

Fig. 6: Full-data classical baselines using Markov transition features. ROC summarizes TPR–FPR behavior across thresholds, whereas PR highlights precision–recall performance under class imbalance.



(a) ROC on subset test ($n = 150$).



(b) PR on subset test ($n = 150$).

Fig. 7: Quantum vs. classical comparison on the matched 8-dimensional subset (seed=1337). Both models are evaluated on the same subset test set ($n = 150$).

Operating-point protocol. In addition to ROC/PR curves, security-oriented operating points are reported. A validation threshold $t^\star$ is selected to satisfy $R_{\text{mal}} \geq 0.90$, and is then applied once to the test split. Table 3 reports the validation and test metrics at this operating point.

Model	Split	$t^\star$	Acc.	R_{mal} (TPR)	R_{good} (TNR)
QSVM (8 qubits)	VAL	0.6379	0.7000	0.9138	0.1364
QSVM (8 qubits)	TEST	0.6379	0.6933	0.9048	0.2000
Classical SVC-RBF (8d)	VAL	0.8498	0.7125	0.9138	0.1818
Classical SVC-RBF (8d)	TEST	0.8498	0.6867	0.8857	0.2222

Tab. 3: Operating-point comparison on the matched 8-qubit subset (seed=1337). Threshold $t^\star$ is selected on validation to satisfy $R_{\text{mal}} \geq 0.90$, then evaluated on test.

Interpreting low accuracy and low R_{good} under a high-recall constraint. Enforcing a high malware-recall constraint shifts the decision threshold toward aggressive malware prediction. This increases false positives, reduces benign recall (R_{good}), and can lower accuracy. Such behavior reflects an intentional security trade-off rather than a modeling error; accordingly, performance is interpreted through ROC/PR curves (Figure 7) and through the recall trade-off analysis (Figure 9), not accuracy alone.

Practical implications of the high-recall constraint. Enforcing $R_{\text{mal}} \geq 0.90$ shifts the operating threshold toward more aggressive malware decisions, which typically lowers R_{good} by increasing false positives. In practice, this operating mode is appropriate when the cost of missed malware is high (e. g., protecting critical endpoints, sandbox triage, or first-stage filtering in a SOC pipeline). The resulting increase in benign alerts can be managed by introducing a second-stage verifier (e. g., a heavier behavioral model or sandbox re-execution), prioritizing alerts by score, or applying class-dependent handling (automatic quarantine for high-confidence malware, analyst review for borderline cases). Therefore, R_{good} at the selected operating point should be interpreted as the expected false-positive burden under a security-first policy, rather than as a stand-alone measure of model quality.

6.3. VQC on the 8-Qubit Subset

A Variational Quantum Classifier (VQC) is evaluated in the same 8-dimensional setting as an alternative to kernel-based learning. VQC learns a decision boundary by optimizing a parameterized quantum circuit. To improve optimization stability in the variational setting, VQC is trained with a larger subset: train $n = 300$, validation $n = 80$, and test $n = 150$, using the same security-oriented evaluation protocol.

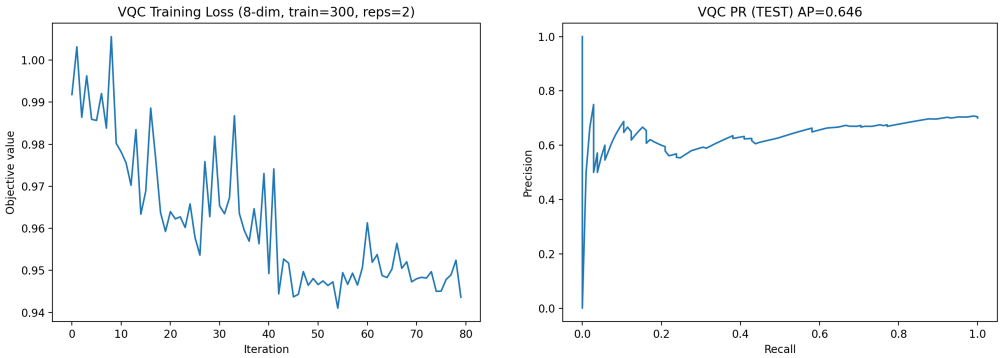
Figure 8 summarizes VQC behavior from two complementary views. The loss trajectory in Figure 8a illustrates the optimization dynamics under the chosen ansatz depth (reps=2). The PR curve in Figure 8b reports threshold-free performance on the subset test set.

VQC avoids explicit kernel-matrix construction, but its decisions remain subject to the same operating-point trade-offs. In particular, enforcing high malware recall at the selected operating point typically reduces benign recall. Therefore, the PR curve (Figure 8b) and the threshold trade-off analysis (Figure 9) provide a clearer interpretation than accuracy alone.

6.4. 12-Qubit QSVM operating-point comparison

A higher-dimensional setting is examined by reducing features to $D = 12$ dimensions and mapping them to a 12-qubit feature map. Because kernel computation becomes substantially more expensive in this regime, results are summarized using operating-point performance rather than full ROC/PR curves. A threshold $t^\star$ is selected on validation to satisfy $R_{\text{mal}} \geq 0.90$ and is then applied once to the test split (Table 4).

Note on the operating-point constraint. The recall constraint ($R_{\text{mal}} \geq 0.90$) is enforced on the validation split to select $t^\star$ (Eq. 18); test performance may fall below τ due to generalization variability, especially in the small-subset regime. Moreover, in



(a) Training loss (VQC, reps=2). (b) PR curve on subset test.

Fig. 8: VQC results on the 8-dimensional subset. Left: optimization dynamics. Right: threshold-free precision–recall performance on subset test.

Model	Split	$t^\star$	Acc.	R_{mal} (TPR)	R_{good} (TNR)
QSVM (12 qubits)	VAL	0.6149	0.7400	0.9041	0.2963
QSVM (12 qubits)	TEST	0.6149	0.6425	0.8185	0.2269
Classical SVC-RBF (12d)	VAL	0.9711	0.7350	0.9110	0.2593
Classical SVC-RBF (12d)	TEST	0.9711	0.6975	0.8861	0.2521

Tab. 4: Operating-point comparison in the 12-qubit setting. Threshold $t^\star$ is selected on validation to satisfy $R_{\text{mal}} \geq 0.90$, then evaluated on test.

this 12-qubit setting the matched classical SVC-RBF baseline outperforms QSVM on the test set at the same validation-selected operating point, reinforcing the scaling and stability challenges of quantum-kernel evaluation as the number of qubits increases.

Model	**Stage**	**Time (s)**
QSVM (12 qubits)	Kernel matrices (K_{train}+K_{val}+K_{test})	$\approx$ 43,200
Classical SVC-RBF (12d)	Train	0.019
Classical SVC-RBF (12d)	Inference (VAL+TEST)	0.018

Tab. 5: Runtime comparison in the 12-dimensional setting (train/val/test subsets: 600/200/400). QSVM time refers to Gram-matrix construction (K_{train}, K_{val}, K_{test}) on a statevector backend.

Table 5 reports the measured runtime under the same 12-dimensional subset protocol. Kernel-matrix construction for QSVM dominates the cost (hours on a statevector backend), whereas the matched classical SVC-RBF trains and scores in milliseconds, empirically illustrating the quadratic scaling burden of kernel-based quantum evalua-

tion.

Accordingly, the main threshold-free quantum–classical comparison is based on the 8-qubit subset experiment, where ROC/PR curves are feasible under identical data constraints. The 12-qubit experiment is interpreted primarily as a scaling/stability case study rather than a setting where quantum models are expected to match the classical baseline.

6.5. Security-oriented trade-off across thresholds

Figure 9 illustrates how malware recall and benign recall vary as the decision threshold changes on the subset experiment. At extreme thresholds, the classifier approaches predicting almost all samples as a single class, yielding recall close to 1 for one class and near 0 for the other. Intermediate thresholds trace the trade-off surface used for selecting security-oriented operating points.

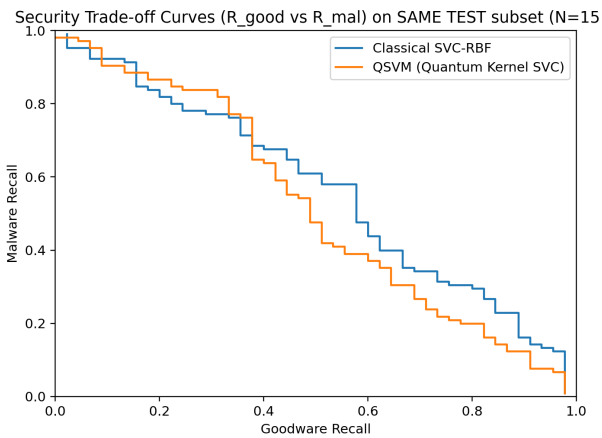


Fig. 9: Trade-off curve on the subset experiment, illustrating the relationship between malware recall and benign recall across decision thresholds.

6.6. Summary of key experiments

Figure 10 summarizes malware recall at the selected operating points for the main experiments. These results highlight controlled quantum–classical comparisons in the small-sample regime, while differences in benign recall reflect the cost of enforcing high malware recall.

6.7. Threats to validity and limitations

- **Subset variability (small-sample regime).** Quantum evaluations rely on matched low-dimensional subsets due to the $O(n^2)$ cost of kernel computation. Results in this regime can vary with the random seed and subset composition; conclusions are therefore limited to the studied subset protocol.

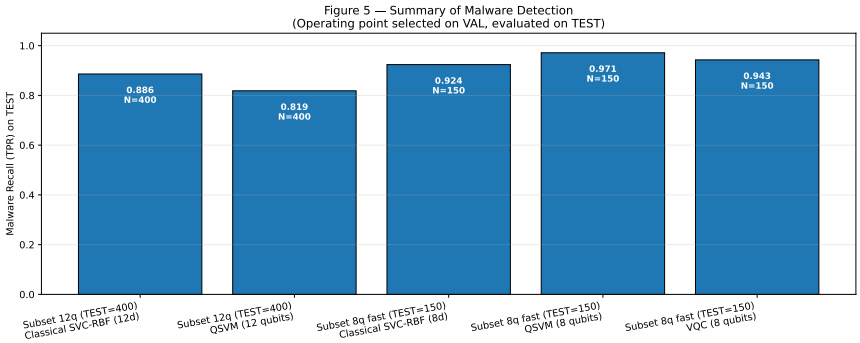


Fig. 10: Summary of malware recall at selected operating points for the main experiments.

- **Feature-order limitation (first-order transitions).** The transition representation captures only first-order local dynamics and does not model longer-range dependencies or motifs spanning distant calls. More expressive sequence models (e. g., higher-order transitions or deep sequence models) may capture additional discriminative structure.
- **Family-aware split definition and residual leakage risk.** The family-aware split reduces leakage by assigning samples with the same family identifier to a single split. Imperfect family labeling or aliasing across families could still introduce residual similarity across splits.
- **Quantum backend specificity.** QSVM and VQC are evaluated on a statevector simulator (exact amplitudes) rather than shot-based sampling or noisy hardware. Performance and feasibility can change under finite-shot estimation and hardware noise; results should be interpreted as simulator-level behavior.
- **Operating-point dependence.** The operating-point protocol enforces $R_{\text{mal}} \geq \tau$ (here $\tau = 0.90$), intentionally trading benign recall for malware detection. Different target recalls (e. g., $\tau = 0.95$) or cost models can lead to different thresholds and trade-offs.

Subset stability. Matched-subset quantum evaluations are necessarily small due to the $O(n^2)$ kernel cost. The subset experiments reported here use a single stratified seed (seed=1337). A multi-seed protocol (e. g., $N = 10$ seeds) reporting mean±std for R_{mal} and R_{good} at $\tau = 0.90$ is left for future work.

7. CONCLUSION

This paper introduced a malware-detection pipeline for Windows API-call sequences. The pipeline maps raw API tokens into shared behavioral states using Word2Vec embeddings and K-means clustering, then summarizes each trace with a fixed-length first-order

Markov transition frequency representation. On the full held-out test set, classical baselines trained on the 961-dimensional transition features provide a strong and scalable reference point. Quantum models were evaluated under explicit computational limits using low-dimensional subsets after TruncatedSVD, including a fidelity-kernel QSVM and a Variational Quantum Classifier (VQC).

Two practical observations emerge from the results. First, threshold-free ROC/PR curves are useful but do not fully reflect deployment needs. Operating points that enforce high malware recall naturally reduce benign recall and can lower accuracy; this reflects a security-driven trade-off rather than model failure. Second, the matched-subset protocol enables a controlled quantum–classical comparison under identical data and feature constraints. In particular, in the 8-qubit setting (reduced to $D = 8$), QSVM/VQC are compared against a matched classical SVC-RBF baseline using the same stratified subset indices and the same operating-point selection protocol, yielding a fair like-for-like comparison in this small-sample regime.

In contrast, in the 12-qubit setting (reduced to $D = 12$), the matched classical SVC-RBF baseline achieves stronger test performance than QSVM at the validation-constrained operating point, while the computational burden of quantum kernel evaluation increases substantially. This higher-qubit regime therefore primarily highlights practical scalability and stability limits of quantum-kernel methods on real cybersecurity traces, rather than suggesting a consistent advantage.

Several limitations remain. The transition representation captures only first-order local dynamics and does not model longer-range dependencies. In addition, subset-based quantum evaluation reduces statistical power, and the reported quantum results are simulator-level because circuits are evaluated with statevector simulation rather than shot-based estimation or noisy hardware.

Future work can extend this study by adding higher-order or context-aware transition features, exploring alternative quantum feature maps and ansatz designs, improving calibration and threshold selection under explicit cost models, and evaluating across multiple random seeds and additional datasets to better quantify stability and generalization across families and collection conditions.

(Received February 10, 2026)

REFERENCES

[1] E. Amer, S. El-Sappagh, T. Abuhamad, B. Ali Saleh Al-Rimy, and A. Mohasseb: Graphshield: Advanced dynamic graph-based malware detection using graph neural networks. Expert Systems Appl. (2025).

[2] E. Amer and I. Zelinka: A dynamic windows malware detection and prediction method based on contextual understanding of api call sequence. Computers Security *92* (2020), 101760. DOI:10.1016/j.cose.2020.101760

[3] E. Amer, I. Zelinka, and S. El-Sappagh: A multi-perspective malware detection approach through behavioral fusion of api call sequence. Computers Security *110* (2021), 102449. DOI:10.1016/j.cose.2021.102449

[4] G. Barrué and T. Quertier: Quantum machine learning for malware classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Springer 2023, pp. 245–260.

[5] A. Bellante, T. Fioravanti, M. Carminati, S- Zanero, and A. Luongo: Evaluating the potential of quantum machine learning in cybersecurity: A case-study on pca-based intrusion detection systems. Computers Security *154* (2025), 104341. DOI:10.1016/j.cose.2025.104341

[6] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles: Variational quantum algorithms. Nature Rev. Phys. *3* (2021), 625–644. DOI:10.1038/s42254-021-00348-9

[7] L- Cui, J. Cui, Y. Ji, Z. Hao, L. Li, and Z. Ding: Api2vec: Learning representations of api sequences for malware detection. In: Proc. 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis 2023, pp. 261–273.

[8] T. Fawcett: An introduction to roc analysis. Pattern Recogn. Lett. *27* (2006), 8, 861–874. DOI:10.1016/j.patrec.2005.10.010

[9] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta: Supervised learning with quantum-enhanced feature spaces. Nature *567* (2019), 7747, 209–212. DOI:10.1038/s41586-019-0980-2

[10] I. Kwon and E. G. Im: Extracting the representative api call patterns of malware families using recurrent neural network. In: Proc. International Conference on Research in Adaptive and Convergent Systems 2017, pp. 202–207.

[11] E. Mariconti, L. Onwuzurike, P. Andriotis, E.De Cristofaro, G. Ross, and G. Stringhini: MAMADroid: Detecting android malware by building markov chains of behavioral models. In: Proc. Network and Distributed System Security Symposium (NDSS). Internet Society 2017.

[12] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven: Barren plateaus in quantum neural network training landscapes. Nature Commun. *9* (2018), 4812.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean: Efficient estimation of word representations in vector space. arXiv preprint, *arXiv:1301.3781,* 2013.

[14] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach: Dynamic malware analysis in the modern era—a state of the art survey. ACM Comput. Surveys, 2019.

[15] M. Emre Sahin, Edoardo Altamura, Oscar Wallis, Stephen P. Wood, Anton Dekusar, Declan A. Millar, Takashi Imamichi, Atsushi Matsuo, and Stefano Mensa: Qiskit machine learning: an open-source library for quantum machine learning tasks at scale on quantum hardware and classical simulators. arXiv preprint, *arXiv:2505.17756,* 2025.

[16] T. Saito and M. Rehmsmeier: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE *10* (2015), 3, e0118432. DOI:10.1371/journal.pone.0118432

[17] M. Schuld: Supervised quantum machine learning models are kernel methods. arXiv preprint, *arXiv:2101.11020,* 2021.

[18] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes: Exponential concentration in quantum kernel methods. Nature Commun. *15* (2024), 1.

[19] I. Zelinka and E. Amer: An ensemble-based malware detection model using minimum feature set. In: Mendel *25* (2019), pp. 1–10. DOI:10.13164/mendel.2019.2.001

[20] F. O. Catak, A. F. Yazi, O. Elezaj, and J. Ahmed: Deep learning based sequential model for malware analysis using windows exe api calls. PeerJ Computer Sci. (2020), 6, e285. DOI:10.7717/peerj-cs.285

Aktham Youssef, Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB–Technical University of Ostrava, 708 00 Ostrava. Czech Republic.

e-mail: aktham.youssef.st@vsb.cz

Ivan Zelinka, Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB–Technical University of Ostrava, 708 00 Ostrava and Quantum Computing Lab, IT4Innovations National Supercomputing Center, VŠB–Technical University of Ostrava, 708 00 Ostrava. Czech Republic-
e-mail: ivan.zelinka@vsb.cz

Eslam Amer, School of Computing, University of Portsmouth, Portsmouth. United Kingdom.
e-mail: eslam.amer@port.ac.uk