

Anomeryx Endpoint Prototype for Behaviour-Based Risk Scoring and Malware Classification

Nurain Farhana Asilah Binti Sharudin, Muhammad Bin Amran*

Faculty of Artificial Intelligence and Cyber Security Universiti Teknikal Malaysia Melaka (UTeM),
Melaka, Malaysia

*Corresponding Author

DOI: <https://doi.org/10.47772/IJRISS.2026.100900176>

Received: 20 September 2026; Accepted: 25 September 2026; Published: 05 October 2026

ABSTRACT

Anomeryx is a local Windows endpoint prototype that integrates malware classification, behavioural risk scoring and persistent alert logging. The study evaluates classifier performance on API-call data, the functional response of the scoring mechanism and resource utilisation in a controlled virtual machine. API-call sequences from MalBehavD-V1 are converted into frequency features, selected using particle swarm optimisation (PSO) and classified by a multilayer perceptron (MLP). A weighted scoring layer combines classifier confidence with normalised file entropy, file-change rate, registry activity and process anomaly to assign five severity levels. Evaluation of 514 benign and malicious samples produced 92.22% accuracy, 97.38% precision, 86.77% recall and 91.77% F1-score. Controlled ransomware-like simulation exercised the scoring and logging functions, including a Critical scenario with a score of 91. Observed CPU utilisation was approximately 3% during live monitoring and 51% during training. The findings demonstrate functional integration within the prototype's scope. Classification results remain preliminary because independence between feature selection and the test partition is not established. General malware labels, simulated behavioural indicators and testing on one virtual machine also limit ransomware-specific and deployment claims. Further validation involves independent test partitions, ransomware-labelled data, comparative scoring experiments and repeated measurements with live endpoint telemetry.

Keywords: endpoint security; malware classification; ransomware; behavioural risk scoring; multilayer perceptron.

INTRODUCTION

Background and motivation

Timely recognition of suspicious endpoint activity can help limit damage from ransomware. Behavioural detection examines actions performed during execution instead of relying only on known signatures. API calls, file operations and other runtime signals may provide evidence of malicious activity, but their meaning depends on the workload and observation context [1]–[4].

Classification and alert prioritisation serve different purposes. A classifier assigns a label or score to an input sample; a monitoring application also needs to present that output and the associated activity in a way a user can review. Anomeryx links the classifier with a behavioural score and persistent alert history. The usefulness of that integration requires separate validation from the classifier's predictive accuracy.

Study objectives and contribution

Anomeryx was developed to examine indicators relevant to ransomware, calculate a weighted severity score and integrate these functions into a local endpoint application. Its contribution is an engineering prototype combining PSO feature selection, MLP classification, an explicit scoring rule and persistent alert logging. The study reports classifier results and resource observations from a controlled Windows virtual machine.

The evaluation comprises three components: classification of benign and malicious API-call samples, functional testing of risk scoring and alert generation, and observation of resource utilisation across application states. The prototype's scope is local detection support and alert presentation on a single endpoint. Automated containment and enterprise deployment are outside the implemented scope.

Related Work

Prior research examines complementary aspects of ransomware detection. Masum et al. [1] investigate feature selection and several machine-learning classifiers, while Abbasi [2] studies automated behaviour-based analysis, detection and classification. Begovic et al. [3] organise encryption-related detection around API and system calls, input/output monitoring and file-system activity. Ferdous et al. [4] review AI-based ransomware detection. Together, these studies motivate the use of execution behaviour as evidence for classification and highlight the range of representations available to a detector.

Runtime monitoring varies across platforms and data sources. Brodzik et al. [5] examine machine learning in the Linux kernel, and Davidian et al. [6] investigate early detection with deep-learning models. Chew et al. [8] identify system-call patterns and implement a streaming finite-state approach for Android ransomware. Differences in operating environment, feature representation and evaluation data limit direct comparison with the Windows desktop prototype considered here.

Feature selection and file activity provide additional design directions. Gurukala and Verma [7] investigate PSO with ensemble-based models; Putrevu et al. [9] examine file-trap selection; and Arányi et al. [10] analyse server-side file-operation logs. Anomeryx combines these broader interests in behavioural analysis and feature selection with a local interface for severity presentation and alert history. Its contribution concerns the implemented system integration and preliminary evaluation; comparative improvements over existing methods have not been established.

METHODOLOGY

Data preparation and classification task

The machine-learning component uses MalBehavD-V1, a dataset of API calls from benign and malicious Windows executables [11]. Each input record contains a file hash, a class label and an API-call sequence. The sequences are converted into numerical API-call frequency features, with benign samples represented by 0 and malicious samples by 1. Preprocessing handles missing values and stores the resulting data in CSV format.

API-call frequency features summarise the occurrence of calls without preserving their complete temporal order. The dataset defines a benign-versus-malicious classification task. Its positive labels do not identify confirmed ransomware as a separate class, so the model metrics quantify general malware classification. Ransomware-like behaviour is evaluated separately through controlled simulation.

Feature selection and model training

PSO searches for a reduced feature subset. Each particle represents a candidate subset, evaluated with a Random Forest classifier using F1-score and three-fold cross-validation. A penalty discourages larger subsets. The selected indices and feature names are retained to maintain a consistent input order across training and inference.

The documented workflow places PSO feature selection before an 80/20 training–test split. The selected features are standardised using StandardScaler and classified by an MLP with three hidden layers of 128, 64 and 32 units. The model uses ReLU activation and the Adam optimiser. The model and scaler are stored as `mlp_model.pkl` and `scaler.pkl`, while `selected_features.npy` and `feature_names.json` preserve the selected feature indices and names.

The evaluation record does not establish that the eventual test samples were excluded from PSO feature selection. Since the Random Forest fitness calculation uses class labels, their inclusion would introduce information from

the test set into model development. The resulting classification metrics are therefore preliminary estimates with a potential source of optimistic bias. This limitation is considered in the interpretation of the results and the future evaluation design.

Behavioural risk score

The risk-scoring layer combines model confidence C with normalised entropy E , file-change rate F , registry-change count R and process anomaly P . Each input lies between 0 and 1, and the implemented weighted score S is:

$$S = 100(0.40C + 0.25E + 0.20F + 0.10R + 0.05P)$$

The weights sum to one, yielding a score between 0 and 100 when all inputs fall within the specified range. Classifier confidence contributes up to 40 points, entropy up to 25, file-change rate up to 20, registry activity up to 10 and process anomaly up to five. The weights are heuristic settings used to express the relative contribution of each input. The score is a severity index; empirical probability calibration and optimisation of the weights have not been established.

Table 1 presents the prototype's integer severity bands and corresponding responses. High and Critical outcomes generate alerts. The level names express the application's prioritisation policy: a Safe label denotes the lowest score band and does not constitute a guarantee that the endpoint is free of malicious activity.

Table 1. Prototype severity bands and responses

Level	Reported score band	Response
Safe	0–19	No alert
Low	20–39	Continue monitoring
Medium	40–64	User review
High	65–84	Generate alert
Critical	85–100	Generate and prioritise alert

Application architecture and implementation

Anomeryx is a local Python desktop application. PyQt5 provides the interface, Pandas and NumPy support numerical processing, and Scikit-learn provides preprocessing, classification and evaluation. Joblib serialises the trained artefacts, psutil collects CPU and memory measurements, and SQLite stores alerts. The main functions are Live Monitor, Scan File, Alert Log and Train Model.

Training produces the feature schema, selected subset, scaler and MLP used during subsequent inference. Classification depends on an API-call feature vector that follows the stored schema and feature order. The scoring layer combines the classifier output, when the required inputs are available, with the behavioural indicators. High and Critical scores are written to the alert database with the timestamp, severity level, score, confidence and activity details.

The application is self-contained and does not require a remote database or application server for its principal functions. However, some file-system and registry indicators are calculated or simulated rather than collected through comprehensive operating-system telemetry. The current implementation therefore demonstrates component integration without establishing continuous end-to-end detection from real endpoint events.

EVALUATION AND RESULTS

Environment and functional checks

Testing was conducted in a Windows 10 x64 virtual machine with two virtual processors, approximately 4.3 GB RAM and a 60 GB NVMe virtual disk. Functional checks covered startup, model validation, monitoring, file scanning, score calculation, alert generation and retrieval, dataset preview, model training and controlled ransomware-like simulation. The tested functions operated within the defined prototype scope.

Classification performance

The confusion matrix contains 223 true positives, 251 true negatives, six false positives and 34 false negatives. The evaluation therefore covers 514 samples: 257 malicious and 257 benign. Table 2 uses malicious as the positive class, and Table 3 presents the metrics calculated from these counts.

Table 2. Confusion matrix for the reported malware classification test

Actual class	Predicted benign	Predicted malicious	Total
Benign	251	6	257
Malicious	34	223	257
Total	285	229	514

Table 3. Metrics derived from the reported confusion counts

Metric	Calculation	Result
Accuracy	$(223 + 251) / 514$	92.22%
Precision	$223 / (223 + 6)$	97.38%
Recall	$223 / (223 + 34)$	86.77%
F1-score	$2 \times 223 / (2 \times 223 + 6 + 34)$	91.77%
Specificity	$251 / (251 + 6)$	97.67%
False-positive rate	$6 / 257$	2.33%
False-negative rate	$34 / 257$	13.23%

The classifier identified 223 of the 257 malicious samples and missed 34, yielding 86.77% recall and a 13.23% false-negative rate. Six of the 257 benign samples were classified as malicious, giving a false-positive rate of 2.33%. The high precision of 97.38% reflects performance under the balanced class distribution used in this evaluation; it may differ under deployment conditions. These findings quantify the existing general-malware test and remain subject to the potential feature-selection bias described in Section 3.2.

Risk scoring and simulated behaviour

Functional tests exercised the Safe, Low, Medium, High and Critical categories. One scenario produced a score of 91 and was classified as Critical. High and Critical outcomes generated records in the Alert Log, demonstrating the connection between severity assignment and persistent alerting. A ransomware-like simulator restricted to test files in a controlled directory increased indicators such as entropy and file-change rate,

producing a corresponding score response. This confirms functional responsiveness to the supplied behaviours. Detection sensitivity against real ransomware, false-alert frequency during benign activity, alert latency and files affected before alerting were not measured.

Resource utilisation

Table 4 reports the recorded CPU and memory percentages for each application state. With Anomeryx closed, utilisation was approximately 3% CPU and 63% memory. During live monitoring, the corresponding values were 3% and 65%. The highest recorded CPU value, 51%, occurred during model training.

Table 4. Resource-utilisation observations in the test virtual machine

Application state	CPU	Memory
Anomeryx closed	3%	63%
Open without monitoring	4%	65%
Live monitoring	3%	65%
Ransomware-like simulation	18%	67%
Model training	51%	67%

Live monitoring showed the same approximate CPU percentage as the closed-application observation and a two-percentage-point increase in memory utilisation. Simulation and training coincided with higher resource utilisation. These differences describe the recorded application states. Without repeated matched measurements and separation of process activity from system activity, they do not provide controlled estimates of Anomeryx overhead.

The observations suggest modest resource use during the tested monitoring state and a greater computational demand during training. Reuse of saved model artefacts allows training to occur separately from routine monitoring. However, the single virtual machine and incomplete live telemetry limit generalisation to physical endpoints and sustained operational workloads.

DISCUSSION AND LIMITATIONS

Interpretation of the integrated score

The weighted rule makes the numerical contribution of each indicator explicit and gives the interface a consistent basis for presenting severity. Several inputs nevertheless have benign explanations: compressed or encrypted files may exhibit high entropy, while backup and synchronisation can generate intensive file activity. Combining indicators provides context, but its predictive benefit has not been quantified through ablation experiments. Dependence between classifier evidence and behavioural inputs may also cause the same underlying activity to influence more than one score component.

The weighting scheme also constrains alert behaviour. Model confidence alone contributes at most 40 points, while the behavioural terms together contribute at most 60. Neither reaches the High threshold of 65 when the other contributes zero. In particular, substituting zero for an unavailable classifier output would prevent the behavioural terms alone from generating a High alert. This property follows directly from the scoring equation and demonstrates the sensitivity of alerting to input availability. The present evaluation does not establish the effect of missing inputs or classifier-confidence calibration on alert reliability.

Validity and reproducibility

The classification results are limited by both task definition and evaluation design. MalBehavD-V1 provides general malware labels, and the evaluation does not quantify performance on confirmed ransomware families. Test independence is uncertain because feature selection is documented before the split. The absence of an MLP comparison without PSO also prevents attribution of any performance benefit to feature selection. In addition, incomplete reporting of the dataset snapshot, duplicate handling, preprocessing rules and optimisation settings limits exact reproduction of the experiment.

The scoring and runtime results have separate limitations. The risk weights and thresholds have not been calibrated against independent labelled endpoint events, and some file-system and registry indicators are simulated. Normalisation windows, fractional-score boundary handling and missing-input behaviour are not fully characterised by the current tests. Resource measurements come from one virtual machine without a documented sampling duration or repetition count and do not isolate process overhead. The evidence consequently supports functional prototype behaviour within the test environment, with limited conclusions about predictive reliability and deployment performance.

Future work

A further classification evaluation will require an independent test partition established before data-dependent preprocessing, feature selection or model tuning. Grouping related executables and holding out complete ransomware families would permit assessment of generalisation beyond closely related samples. Training-only cross-validation would support PSO and hyperparameter selection, followed by a final evaluation on the untouched test partition. A versioned dataset, explicit inclusion rules, saved partitions and complete model settings would improve reproducibility.

Comparisons with an MLP without PSO, regularised logistic regression and Random Forest would help isolate the contribution of feature selection and classifier choice. Common data partitions and comparable tuning budgets would support interpretable comparisons. Confusion matrices, precision, recall, specificity, F1-score and uncertainty intervals would characterise classification errors, while analysis by ransomware family and benign workload would reveal conditions associated with missed detections and false positives.

Further score validation could compare model-only, behaviour-only and combined approaches on labelled endpoint activity, including benign backup, compression, installation and synchronisation workloads. Calibration of weights and thresholds on development data, followed by evaluation on held-out sessions at comparable false-alert rates, would assess the value of the combined score. Removal of individual indicators would quantify their contribution. Alert delay, false alerts per endpoint-day and files affected before an alert would connect score quality to operational outcomes.

Runtime validation would extend the prototype to documented Windows event collection and verify consistency between training and live feature representations. Repeated matched workload sessions across multiple hardware configurations would quantify variability in process CPU use, resident memory, event processing and detection latency. These measurements would provide a stronger basis for assessing the lightweight design objective and the costs of complete live monitoring.

CONCLUSION

Anomeryx integrates API-call-based malware classification, behavioural severity scoring and persistent alert logging in a local Windows prototype. Its preliminary evaluation on 514 benign and malicious samples yielded 92.22% accuracy, 97.38% precision, 86.77% recall and 91.77% F1-score. Controlled ransomware-like simulation demonstrated functional scoring and alert generation, while resource observations showed greater CPU utilisation during training than during the tested monitoring state.

The principal contribution is the integration of classification and explicit risk prioritisation within a self-contained endpoint application. Potential feature-selection leakage, general malware labels, heuristic score

settings and a single virtual-machine environment constrain the findings. Independent classification tests, ransomware-specific evaluation, scoring ablations and repeated measurements with live telemetry are the next steps towards establishing predictive reliability and deployment suitability.

Ethical Approval

Ethical approval was not required because this study did not involve human participants or animals.

Conflict Of Interest

The authors declare no conflict of interest.

Data Availability

The MalBehavD-V1 source dataset used in this study is publicly available from the repository cited in reference 11.

REFERENCES

1. Masum, M., Faruk, M. J. H., Shahriar, H., Qian, K., Lo, D., & Adnan, M. I. (2022). Ransomware classification and detection with machine learning algorithms. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 316–322). IEEE. <https://doi.org/10.1109/CCWC54503.2022.9720869>
2. Abbasi, M. S. (2023). Automating behavior-based ransomware analysis, detection, and classification using machine learning [Doctoral thesis, Victoria University of Wellington]. <https://openaccess.wgtn.ac.nz/articles/thesis/22180858>
3. Begovic, K., Al-Ali, A., & Malluhi, Q. (2023). Cryptographic ransomware encryption detection: Survey. arXiv. <https://doi.org/10.48550/arXiv.2306.12008>
4. Ferdous, J., et al. (2024). AI-based ransomware detection: A comprehensive review. IEEE Access, 12, 136666–136695. <https://doi.org/10.1109/ACCESS.2024.3461965>
5. Brodzik, A., et al. (2024). Ransomware detection using machine learning in the Linux kernel. arXiv. <https://doi.org/10.48550/arXiv.2409.06452>
6. Davidian, M., Kiperberg, M., & Vanetik, N. (2024). Early ransomware detection with deep learning models. Future Internet, 16(8), Article 291. <https://doi.org/10.3390/fi16080291>
7. Gurukala, N. K. Y., & Verma, D. K. (2024). Feature selection using particle swarm optimization and ensemble-based machine learning models for ransomware detection. SN Computer Science, 5. <https://doi.org/10.1007/s42979-024-03454-4>
8. Chew, C. J. W., Kumar, V., Patros, P., & Malik, R. (2024). Real-time system call-based ransomware detection. International Journal of Information Security, 23, 1839–1858. <https://doi.org/10.1007/s10207-024-00819-x>
9. Putrevu, M. A., et al. (2024). A comprehensive analysis of machine learning based file trap selection methods to detect crypto ransomware. arXiv. <https://doi.org/10.48550/arXiv.2409.11428>
10. Arányi, G., Miseta, T., & Szücs, V. (2026). Ransomware detection based on server-side file operation logs using machine learning. Journal on Information Security, 2026, Article 8. <https://doi.org/10.1186/s13635-026-00229-7>
11. mpasco. (n.d.). MalBehavD-V1: A dataset of API calls extracted from malware and benign executable files in Windows [Data set]. GitHub. Retrieved September 22, 2026, from <https://github.com/mpasco/MalbehavD-V1>