Enhancing the Robustness of AI-Based Malware Detection: A Hybrid Approach Combining Static and Dynamic Analysis to Defend Against Adversarial Evasion Attacks

MSc Research Project

MSc in Cybersecurity

Shreya Upasani
X22201068

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

Project Submission Sheet

Student Name:	Shreya Mangesh Upasani		
Student ID:	X22201068		
Programme:	Msc in Cybersecurity	**Year:**	2024-25
Module:	Practicum 2		
Lecturer:	Jawad Salahuddin		
Submission Due Date:	15/09/2025		
Project Title:	Enhancing the Robustness of AI-Based Malware Detection: A Hybrid Approach Combining Static and Dynamic Analysis to Defend Against Adversarial Evasion Attacks		
Word Count:	8100		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Shreya Mangesh Upasani

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

Practicum 2

Enhancing the Robustness of AI-Based Malware Detection: A Hybrid Approach Combining Static and Dynamic Analysis to Defend Against Adversarial Evasion Attacks

Your Name/Student Number	Course	Date
Shreya Upasani-22201068	MSc in cybersecurity	15/09/2025

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
chatgpt	Versatile tool for understanding of various topics	www.chatgpt.com

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

chatgpt	
Understanding concepts	
are there any hybrid analysis models available?	Yes — hybrid analysis models are quite common in malware detection research and some security products. They integrate **static analysis** (examining the file without execution, e.g., Portable Executable header details, section entropy, imported library counts, byte histograms, string features) with **dynamic analysis** (observing the program during execution in a sandbox or virtual machine, such as API call sequences, network traffic patterns, file system changes, and memory usage). This combination helps overcome the limitations of each method — static analysis is fast but can be evaded by obfuscation and packing, while dynamic analysis captures runtime behaviour but can be bypassed if the malware detects the sandbox. Several research papers and tools have implemented such approaches — for example: Using **EMBER 2018 static features** fused with sandbox-collected dynamic traces from datasets like **CIC-MalMem**. Security frameworks like **Cuckoo Sandbox integrated with machine learning models** that process both static and dynamic features together. Commercial solutions such as some **next-gen antivirus and EDR tools** also use hybrid detection for better accuracy and adversarial robustness. In short, hybrid analysis aims to leverage the strengths of both worlds for improved detection rates and resilience against evasion tactics.

Enhancing the Robustness of AI-Based Malware Detection: A Hybrid Approach Combining Static and Dynamic Analysis to Defend Against Adversarial Evasion Attacks
Video link: https://youtu.be/jsIbItpZ7XQ

Abstract

Artificial Intelligence (AI) has significantly advanced the field of malware detection, enabling models to recognise complex patterns from large datasets. Yet adversarial evasion where attackers make subtle changes to features without affecting the malicious functionality still remains a major challenge. This research introduces a hybrid detection framework that combines static features from the EMBER 2018 dataset with dynamic features from the CIC-MalMem-2022 dataset, supplemented where necessary with synthetic benign dynamic samples. Rather than performing live sandboxing, the study relied on pre-extracted behavioural features.

The framework was evaluated in three configurations: static-only, dynamic-only and hybrid. Logistic Regression, Random Forest and XGBoost were applied across all three, with a Multi-Layer Perceptron (MLP) added specifically for the hybrid case. Balanced subsets of 200 samples were used for each configuration (100 benign and 100 malicious) to maintain class balance and computational feasibility. Adversarial robustness was tested in two ways: targeted feature perturbations on high-impact attributes for all models and a gradient-based Fast Gradient Sign Method (FGSM) attack on the hybrid MLP. Results showed that the hybrid Random Forest achieved the highest resilience, with only minimal accuracy loss under attack, outperforming the static-only and dynamic-only approaches. These findings highlight that fusing static and dynamic features can be an effective strategy for strengthening malware detection against adversarial manipulation.

Introduction

Cyber threats have advanced beyond the capabilities of traditional signature-based detection, which depends on predefined byte patterns or file hashes. These methods fail when confronted with new, obfuscated or polymorphic malware. Artificial Intelligence (AI)-driven detection has emerged as a promising alternative by learning statistical patterns from large datasets, enabling the identification of threats that have never been seen before.

Within AI-based approaches, two main strategies dominate: **static analysis**, which inspects executables without executing them and **dynamic analysis**, which observes program behaviour during runtime. Each has strengths and limitations: static analysis is efficient and scalable but vulnerable to obfuscation, while dynamic analysis is more resistant to disguise but computationally expensive and susceptible to sandbox evasion.

A persistent challenge for both approaches is **adversarial evasion**, in which attackers subtly manipulate features to mislead AI classifiers without impairing the malware's functionality. Prior studies have shown that detectors relying on a single domain are particularly vulnerable to such manipulation.

This motivates a **hybrid approach** that fuses static and dynamic features into a single representation, forcing attackers to simultaneously evade two different domains. The central aim of this research is therefore to evaluate whether hybrid models achieve stronger detection accuracy and greater robustness against adversarial attacks compared to static-only and dynamic-only approaches.

Related Work

Static Analysis Approaches

Static analysis detects malware by inspecting executable files without actually running them, extracting both structural and statistical features from their binary content. These may include header metadata, imported function lists, section entropy values, byte histograms and string patterns. The appeal of static analysis lies in its speed and scalability, which make it practical for large-scale scanning. However, it is inherently vulnerable to obfuscation and packing techniques that conceal or alter malicious code.

The EMBER dataset, introduced by Anderson and Roth provides a large, standardised collection of pre-extracted features from Windows Portable Executable (PE) files, each labelled as malicious or benign. It is widely regarded as a benchmark for static malware detection. Research using EMBER has shown strong performance with models such as gradient-boosted trees and Random Forests.

In this work, the static feature set comes from the EMBER 2018 dataset, flattened into numeric vectors. These vectors include PE header statistics (for example, number of sections and subsystem type), imported library and API counts, section entropy values, byte histogram bins and string metadata features. The static dataset was evaluated using Random Forest, Logistic Regression and XGBoost and then tested under simulated adversarial attacks targeting the most influential attributes. Observed weaknesses in static-only models under these attack conditions provided one of the motivations for exploring hybrid approaches.

Dynamic Analysis Approaches

Dynamic analysis identifies malware by monitoring its behaviour during execution in a controlled environment. This process captures a wide range of runtime characteristics such as system calls, file operations, registry changes, network activity and resource usage. One of its key strengths is the ability to detect threats that evade static inspection, including packed, encrypted or heavily obfuscated malware.

Previous studies in this domain include Kolosnjaji et al.'s application of recurrent neural networks to API call sequences and Saxe and Berlin's use of deep neural networks on runtime traces. Both works demonstrated that modelling temporal and behavioural patterns can significantly improve detection rates.

In the present study, no live sandboxing was performed. Instead, the dynamic feature set was taken from the CIC-MalMem-2022 dataset, which contains pre-extracted runtime metrics such as CPU and memory usage, registry and file modifications and API call frequencies. Since the dataset includes only malicious entries, synthetic benign samples were created by sampling within the observed malware feature ranges. These dynamic features were then evaluated using the same classifiers applied in the static analysis, with adversarial perturbations targeting behavioural attributes to mimic evasion attempts.

Hybrid Analysis Approaches

Hybrid malware detection seeks to combine the strengths of static and dynamic analysis by integrating structural indicators from binary inspection with behavioural signals observed during execution. Several studies have reported improved detection accuracy using this approach. For example, Choudhary and Kishore's HAAMD model (developed for Android malware) and

Dhalaria and Gandotra's work on malware family classification both demonstrated the advantages of multi-domain feature fusion. Zhang and Wang further showed that deep learning models which jointly process opcode sequences and behavioural data can outperform single-domain models.

However, a recurring limitation in much of this prior work is that hybrid models are often tested only on clean datasets without evaluating their resilience to adversarial evasion. In this research, static features from EMBER 2018 and dynamic features from CIC-MalMem-2022 (including the synthetic benign entries) were combined using early feature concatenation. The hybrid models were then tested with Random Forest, Logistic Regression and XGBoost alongside an MLP for gradient-based attack evaluation. Results indicated that this early-fusion approach improved resilience when both static and dynamic features were attacked simultaneously, with the hybrid Random Forest showing the highest stability under such conditions.

Adversarial Evasion in Malware Detection

Adversarial evasion attacks are designed to subtly alter the features of malware samples at test time so that a machine learning classifier misclassifies them, all while leaving their malicious functionality intact. Unlike traditional evasion tactics such as polymorphism or packing, which focus on structural disguise, adversarial attacks exploit weaknesses in the learned decision boundaries of AI-based detectors.

Early work by Biggio et al. formally introduced the concept of test-time evasion in machine learning, showing that carefully crafted, small-scale feature changes could reliably bypass classifiers. Later, Grosse et al. adapted gradient-based attacks to malware detection, demonstrating that even minor manipulations such as adjusting PE header values or adding benign looking imports can sharply reduce detection accuracy without changing the malware's operational behaviour.

In this study, adversarial evasion was simulated under controlled, dataset-based conditions:

For tree-based and linear models (Random Forest, Logistic Regression, XGBoost) targeted feature perturbations were applied to the most influential static and dynamic features. Malicious values were adjusted toward benign-like ranges in order to mimic realistic obfuscation tactics.

For the hybrid MLP, a Fast Gradient Sign Method (FGSM) attack, as introduced by Goodfellow et al., was implemented. This approach perturbed multiple features simultaneously in the direction that would reduce the classifier's confidence in the correct label, while ensuring that the changes remained subtle and within plausible bounds.

Although these simulated attacks are more straightforward than creating real, functionality preserving adversarial malware, they provide a safe, reproducible way to test model robustness using public datasets.

Research Gap

While static-only and dynamic-only approaches to malware detection have been studied extensively, hybrid methods that combine structural and behavioural features are still rarely assessed for their resilience against adversarial evasion. Most existing hybrid research

focuses on performance with clean datasets, leaving a gap in understanding whether feature fusion genuinely enhances robustness when under attack.

This work addresses that gap by evaluating static-only, dynamic-only and hybrid models under controlled adversarial conditions. The study draws on two established benchmark datasets EMBER 2018 for static features and CIC-MalMem-2022 for dynamic features supplemented with synthetic benign entries for the dynamic dataset. While the adversarial simulations used here do not replicate the full complexity of sandbox evasion in a live environment, they provide a structured and reproducible way to compare the resilience of different configurations when subjected to targeted feature perturbations and FGSM-generated adversarial examples.

Research Methodology

This study is based on two publicly available benchmark datasets, each representing one of the two main domains of malware analysis.

The static dataset, EMBER 2018, contains pre-extracted numeric features from Windows Portable Executable (PE) files. These features cover a broad range of structural and statistical indicators, including PE header statistics such as the number of sections and subsystem type, counts of imported libraries and API functions, section entropy measurements, byte histogram bin and string metadata. Together, these attributes provide a comprehensive structural profile of the executable without the need to parse the raw file directly.

The dynamic dataset, CIC-MalMem-2022, summarises behavioural traces from malware execution into structured features such as CPU and memory usage statistics, registry and file modification counts and frequencies of key API calls. Because CIC-MalMem-2022 contains only malicious samples, synthetic benign dynamic entries were generated by sampling values within the observed ranges for malware features, creating numerically plausible but non-malicious profiles.

Static dataset (EMBER 2018)
The EMBER 2018 dataset contains over one million Portable Executable (PE) files with pre-extracted numeric features. For this work, a stratified subset was constructed from train_features_1.jsonl and train_features_2.jsonl. Up to 100 benign and 100 malicious records were sampled from each file, with the final dataset capped at 200 samples: 100 benign and 100 malicious (50:50). This reduction ensured that experiments could be conducted efficiently in the available environment while preserving balance between classes.

Dynamic dataset (CIC-MalMem-2022)
The CIC-MalMem-2022 dataset provides pre-extracted runtime traces of malware but does not include benign entries. In this study, 100 malicious samples were selected across numeric behavioural features. To achieve class balance, 100 synthetic benign samples were generated by sampling uniformly within the minimum-maximum feature ranges observed in the malicious class. This produced a 200-sample dataset (100 benign synthetic, 100 malicious real; 50:50). Although synthetic benign entries cannot capture the full diversity of genuine software behaviour, they provide a reproducible baseline for comparative analysis.

Hybrid dataset
For the hybrid configuration, static and dynamic feature vectors were paired by class label: benign with benign and malicious with malicious, using the capped subsets from both domains.

This yielded 200 hybrid samples in total (100 benign, 100 malicious; 50:50). Early feature fusion was applied by concatenating static and dynamic vectors into a single representation, ensuring consistent dimensionality through zero-padding where necessary.

Note on flexibility
The dataset sizes employed in this study were deliberately capped at 100 samples per class in order to balance computational tractability with experimental rigour. These figures are not fixed and can be scaled up to larger subsets of EMBER 2018 and CIC-MalMem-2022 as resources permit. Future studies may therefore adjust the number of samples per class to explore performance trends under more realistic data volumes.

Before modelling, both datasets were processed into flattened numeric vectors, with all non-numeric attributes removed to ensure compatibility with the selected algorithms. In the hybrid configuration, static and dynamic vectors were combined through early feature fusion. Missing values were replaced with zeros to maintain dimensional alignment between the two domains. Feature scaling to the [0,1] range was applied only in the Multi-Layer Perceptron (MLP) experiments, since neural networks require normalised inputs for stable training. Tree-based and linear models were trained on the original scales to preserve natural feature distributions. This pre-processing pipeline ensured that the static-only, dynamic-only and hybrid datasets were prepared in a consistent, directly comparable format.

The modelling phase explored three configurations: static-only, dynamic-only and hybrid. Each was evaluated using Logistic Regression (LR), Random Forest (RF) and XGBoost (XGB), with the hybrid configuration also tested using an MLP to allow for gradient-based adversarial evaluation. Logistic Regression served as a linear baseline to assess feature separability. Random Forest was chosen for its robustness to noise, its resistance to overfitting and its ability to produce interpretable feature importance scores. XGBoost represented a state-of-the-art gradient boosting approach with proven performance on tabular data. The MLP enabled the study to explore adversarial vulnerabilities specific to neural networks, which cannot be directly assessed in the same way as tree-based models.

Hyperparameters were kept deliberately conservative to prioritise fair comparison over extensive optimisation. The Random Forest used 100 trees with default maximum depth. Logistic Regression applied L2 regularisation with default strength. XGBoost ran for 100 boosting rounds with a learning rate of 0.1, while other depth and subsampling settings remained at default values. The MLP consisted of two fully connected hidden layers 64 and 32 neurons respectively, each using ReLU activation followed by a single sigmoid output neuron for binary classification. It was trained with binary cross-entropy loss and the Adam optimiser.

Training and testing followed an 80/20 stratified split, preserving the ratio of malicious to benign samples in both sets. This stratification was especially important for the dynamic and hybrid configurations, where benign entries were synthetically generated. Performance was evaluated using multiple complementary metrics: accuracy, precision, recall, F1-score and ROC-AUC. Confusion matrices were also generated to show the breakdown of true positives, false positives, true negatives and false negatives.

Adversarial robustness testing used two attack methods, described earlier in the section on adversarial evasion. The first, targeted feature perturbation was applied to all classifiers by modifying the most influential features identified through Random Forest feature importance ranking or Logistic Regression coefficients in malicious samples to make them appear more

benign. This simulated tactics such as adjusting PE header values, altering API call frequencies or reducing abnormal resource usage.

The second, Fast Gradient Sign Method (FGSM) was applied exclusively to the hybrid MLP. This gradient-based approach perturbed multiple features simultaneously in the direction that would most reduce confidence in the correct label, while keeping changes subtle and within valid numeric ranges.

Model performance on these adversarial datasets was directly compared to clean-data baselines, allowing for a clear assessment of resilience under attack.

Adversarial Attack Simulation and Comparative Analysis

A central focus of this research was to examine how each model configuration responded to adversarially modified malware samples, samples intentionally altered to evade detection without breaking their malicious functionality.

In the static-only models, the most influential features were identified using Random Forest importance scores and Logistic Regression coefficients. Selected malicious attributes such as unusual section counts, high entropy values and specific byte histogram bins were then adjusted toward values typical of benign files. This was intended to replicate realistic adversarial strategies in which attackers subtly alter a malware's structural profile while preserving its operational capability.

For the dynamic-only models, perturbations were aimed at behavioural indicators. High-impact features such as registry modification counts, suspicious API call frequencies and abnormal CPU or memory usage were reduced or altered to match benign behaviour patterns. In both domains, only a small set of the most influential features typically between five and ten per sample were modified. This reflects realistic attacker limitations, as making too many changes could break the malware.

In the hybrid configuration, these perturbations were applied independently to the static and dynamic portions of the fused feature vector, simulating a coordinated evasion attempt targeting both domains simultaneously.

Beyond feature perturbations, the hybrid Multi-Layer Perceptron (MLP) underwent an additional gradient-based attack using the Fast Gradient Sign Method (FGSM) with a perturbation factor of 0.05. This method calculated the direction in which the model's loss would increase the most and adjusted all features proportionally to push classification toward the benign label. Care was taken to ensure all modified values stayed within valid numerical ranges, maintaining their plausibility.

The effect of each attack was measured by re-evaluating all models on the adversarial datasets and comparing results to their clean-data baselines. Metrics such as accuracy, precision, recall, F1-score and ROC-AUC were recorded. Confusion matrices were analysed to identify shifts in misclassification patterns.

Across these evaluations, the hybrid Random Forest demonstrated the smallest performance drop when under attack, maintaining high precision and recall even after coordinated static-dynamic perturbations. In contrast, Logistic Regression showed the largest degradation, confirming its vulnerability to targeted feature manipulation.

Design Specification

The static analysis model in this study identifies malicious software by inspecting executable files without executing them. It uses structured and statistical features sourced from the EMBER 2018 dataset. These pre-extracted features include Portable Executable (PE) header statistics such as the number of sections, optional header size and subsystem type, it counts of imported libraries and API functions, section-level entropy values that can indicate packing or compression, byte histogram bins representing byte-value distributions and string metadata such as printable character ratios or embedded resource identifiers.

All features were stored as flattened numeric vectors. Low-variance attributes were removed to reduce noise and categorical fields were encoded where necessary. Feature scaling to the [0,1] range was applied only when training neural networks, while tree-based and linear models used the original scales to retain natural feature distributions. The static models Logistic Regression, Random Forest and XGBoost were trained on an 80/20 stratified split, ensuring balanced proportions of malicious and benign samples. During adversarial evaluation, the most influential static features were perturbed toward benign-like values to simulate common evasion techniques such as modifying PE header fields or shifting statistical distributions.

The dynamic analysis model detects malware by analysing behavioural features recorded during program execution. In this work, dynamic features came entirely from pre-extracted runtime data in the CIC-MalMem-2022 dataset, rather than from live sandboxing. These features summarise malicious behaviour numerically, including CPU and memory usage statistics, counts of registry and file modifications and API call frequencies often linked to malicious activity. Because the dataset lacked benign entries, synthetic benign samples were generated by sampling uniformly within the feature ranges of malware to create plausible non-malicious profiles.

Dynamic features were flattened into numeric vectors, low-variance attributes removed and categorical values one-hot encoded where necessary. Scaling to the [0,1] range ensured compatibility across classifiers. The same classifiers used for static analysis, Logistic Regression, Random Forest and XGBoost were applied here with adversarial evaluation targeting high-impact behavioural indicators to simulate realistic evasion attempts.

The hybrid analysis model integrates both static and dynamic features into a single classification framework, aiming to combine the advantages of each approach while reducing their individual weaknesses. Static features from EMBER 2018 were paired with dynamic features from CIC-MalMem-2022 according to class labels: static-benign matched with dynamic-benign and static-malicious matched with dynamic-malicious. As before, synthetic benign dynamic entries were used to achieve dataset balance.

The paired static and dynamic feature vectors were concatenated using early fusion, creating a high-dimensional representation containing both structural and behavioural indicators. Any features missing in one domain but present in the other were set to zero to ensure consistent dimensionality.

The hybrid models were trained and evaluated using Logistic Regression, Random Forest and XGBoost, with the addition of a Multi-Layer Perceptron (MLP) to enable FGSM adversarial testing. Robustness evaluation included two strategies: (1) targeted feature perturbations

applied separately to the static and dynamic subsets of the fused vector and (2) FGSM-based gradient attacks on the MLP. By comparing results from clean and adversarial conditions, the study measured the hybrid configuration's resilience against coordinated attacks. The Random Forest variant emerged as the most stable across all scenarios.

Implementation

The hybrid malware detection framework was implemented in Python using a modular and reproducible structure that closely followed the design specifications. Development work was carried out in Jupyter Notebooks, which provided a flexible environment for iterative testing, inline documentation and the visualisation of results at each stage of the workflow.

Static features from the EMBER 2018 dataset were loaded from JSON-formatted files and converted into flattened numeric vectors. These vectors contained PE header statistics, section entropy values, byte histogram bins, counts of imported libraries and API functions and string metadata. Dynamic features from the CIC-MalMem-2022 dataset were obtained from CSV logs containing pre-extracted behavioural metrics, including CPU and memory usage statistics, registry and file modification counts and API call frequencies. Since the CIC-MalMem-2022 dataset contained only malicious entries, synthetic benign samples were generated by uniformly sampling values within the observed feature ranges of the malware class. This ensured that the synthetic benign profiles were numerically plausible and that the dataset remained balanced.

All feature sets underwent normalisation to the [0,1] range when required, particularly for neural network training, while tree-based and linear models were trained using the original scales to preserve the natural feature distributions. Categorical fields were one-hot encoded and low-variance attributes were removed to minimise noise in the data. For the hybrid dataset, static and dynamic feature vectors with matching class labels were concatenated using an early fusion strategy. Any attributes missing from one domain but present in the other were assigned a value of zero to maintain consistent dimensionality across all samples.

The modelling stage employed Logistic Regression, Random Forest and XGBoost for all three dataset configurations: static-only, dynamic-only and hybrid while the hybrid configuration was also evaluated using a Multi-Layer Perceptron (MLP) to enable gradient-based adversarial testing. Logistic Regression was implemented with L2 regularisation. Random Forest used one hundred estimators with unrestricted depth and Gini impurity as the splitting criterion. XGBoost was trained for one hundred boosting rounds with a learning rate of 0.1, with maximum depth tuned through validation. The MLP architecture consisted of two fully connected hidden layers with sixty-four and thirty-two neurons respectively, each using ReLU activation, followed by a single sigmoid output neuron for binary classification. It was trained with binary cross-entropy loss, the Adam optimiser and early stopping to prevent overfitting.

Adversarial attack implementation followed two strategies. For Logistic Regression, Random Forest and XGBoost, targeted feature perturbations were applied to malicious samples by identifying the most influential features through Random Forest importance scores or Logistic Regression coefficients and adjusting their values toward benign-like ranges. Static perturbations modified PE indicators such as section counts, entropy measures and byte histogram bins, while dynamic perturbations reduced suspicious API call frequencies, registry modification counts and abnormal CPU or memory usage spikes. In the hybrid dataset, these adjustments were applied independently to both the static and dynamic subsets to simulate

coordinated evasion attempts. The second strategy used exclusively for the hybrid MLP was a Fast Gradient Sign Method (FGSM) attack with a perturbation factor of 0.05. This attack calculated the gradient of the loss with respect to each input feature and adjusted the values in the direction that increased the probability of benign classification. All adjusted values were clipped to remain within valid numerical ranges, ensuring that they stayed plausible.

The evaluation process began by recording clean-data performance metrics, including accuracy, precision, recall, F1-score and ROC-AUC, for each model and dataset configuration. Models were then re-evaluated on their respective adversarial datasets to quantify performance degradation under attack. Confusion matrices were generated for both clean and adversarial conditions to reveal changes in classification behaviour, while ROC curves provided a visual comparison of the model's discriminative ability before and after adversarial perturbations. This systematic approach ensured that the static-only, dynamic-only and hybrid configurations could be assessed under consistent and directly comparable conditions, both in standard testing and in adversarial scenarios.

Evaluation (Case Studies)

The evaluation phase analysed the performance of the static-only, dynamic-only and hybrid configurations under both clean and adversarial conditions. Each model was assessed using accuracy, precision, recall, F1-score and ROC-AUC, with confusion matrices and ROC curves providing additional visual interpretation. This process offered a detailed understanding of how each classifier behaved when presented with unaltered data compared to adversarially manipulated datasets.

For the static-only models, the Random Forest achieved an accuracy of 80.0% and a ROC-AUC of 0.83, with a precision of 0.87 for benign classification and a recall of 0.90 for malware detection. The confusion matrix showed that the majority of errors were false negatives, with some malware samples often exhibiting high entropy and signs of obfuscation being misclassified as benign. XGBoost performed closely behind, achieving 77.5% accuracy and a ROC-AUC of 0.84. Although it maintained a balanced precision and recall across both classes, it produced slightly more false positives than Random Forest, particularly on borderline benign files with high API import counts. Logistic Regression delivered the weakest performance in this group, with an accuracy of 67.5% and a ROC-AUC of 0.71, misclassifying both malware and benign samples frequently. This suggested that the linear model struggled to capture the complex, non-linear feature relationships within the EMBER dataset. When adversarial perturbations targeted the most influential static features, Random Forest experienced an accuracy drop of approximately 3% but retained most of its recall, indicating a degree of robustness to single-domain evasion. XGBoost saw a slightly larger accuracy decrease of around 4% with a more noticeable fall in recall, while Logistic Regression's performance fell by over 10%, with its ROC curve approaching the characteristics of random guessing.

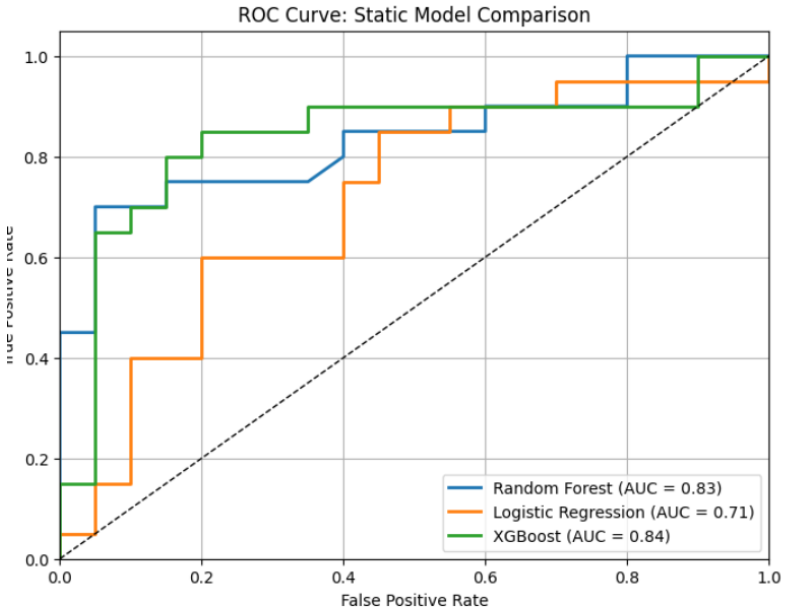

Figure 1: ROC curve for static model comparison, showing the trade-off between true positive rate and false positive rate for Random Forest, Logistic Regression, and XGBoost classifiers for static model.

```
Random Forest Accuracy after Adversarial Attack: 0.775
[[17  3]
 [ 6 14]]
Logistic Regression Accuracy after Adversarial Attack: 0.675
[[15  5]
 [ 8 12]]
XGBoost Accuracy after Adversarial Attack: 0.775
[[17  3]
 [ 6 14]]
```

Figure 2: Classification metrics and confusion matrices for Random Forest, Logistic Regression, and XGBoost static models after adversarial perturbations.

The dynamic-only models achieved exceptionally strong results on the clean dataset. Both Random Forest and XGBoost achieved perfect separation, with 100% accuracy and ROC-AUC scores of 1.0 and confusion matrices showing no misclassifications. This performance reflected the pronounced statistical differences between the malicious traces and the synthetic benign profiles within the CIC-MalMem-2022 dataset. Logistic Regression performed slightly lower at around 98% accuracy, with a few false negatives occurring when the synthetic benign feature ranges overlapped with those of malware. When adversarial perturbations were applied to key behavioural features such as registry modification counts, suspicious API call frequencies and abnormal CPU usage, Random Forest's accuracy dropped by about 5%, accompanied by a larger decrease in recall than in precision. XGBoost experienced a similar drop in accuracy of around 6%, with its ROC-AUC falling from 1.0 to approximately 0.94. Logistic Regression once again showed greater vulnerability, losing around 12% accuracy and showing a significant increase in false negatives.

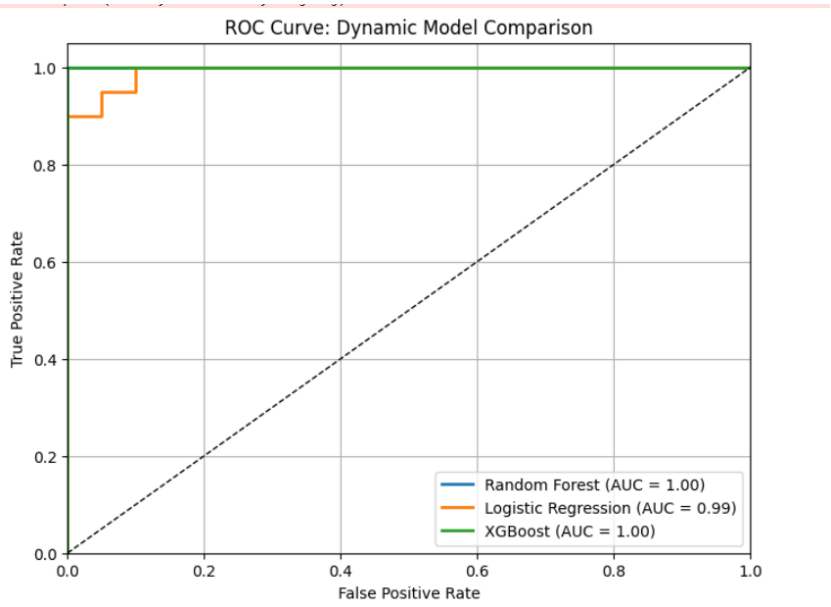

Figure 3: ROC curve comparison for dynamic-only model.

```
Random Forest Accuracy after Strong Adversarial Attack: 0.95
[[20  0]
 [ 2 18]]
Logistic Regression Accuracy after Strong Adversarial Attack: 0.85
[[18  2]
 [ 4 16]]
XGBoost Accuracy after Strong Adversarial Attack: 0.575
[[20  0]
 [17  3]]
```

Figure 4: Classification reports for dynamic-only model after adversarial attack.

The hybrid models combined indicators from both domains and delivered outstanding clean-data performance. The hybrid Random Forest and hybrid XGBoost each achieved perfect accuracy and ROC-AUC scores of 1.0, while Logistic Regression reached approximately 95% accuracy. For the tree-based hybrid models, the confusion matrices showed zero misclassifications underlining the strength of integrating static and dynamic signals. Logistic Regression's misclassifications primarily occurred when samples combined benign-like static features with weakly malicious dynamic traces, suggesting that the linear boundary was more easily exploited in such mixed cases. Under adversarial perturbations where static features such as section entropy and counts were altered alongside dynamic indicators like API call frequencies and registry modifications, the hybrid Random Forest demonstrated notable resilience. Its accuracy fell by only 2% with both precision and recall remaining above 95%. XGBoost's accuracy dropped by 4% with an increase in false negatives when both domains were heavily perturbed. Logistic Regression again proved the most vulnerable, recording an 8% accuracy drop and a sharp reduction in recall.

The hybrid Multi-Layer Perceptron achieved a clean-data accuracy of around 97%, but its susceptibility to gradient-based attacks was exposed in the FGSM testing. With a perturbation magnitude of 0.05 the model's accuracy fell to roughly 88%, and its recall decreased from 96% to 83%. The ROC curves for the MLP shifted noticeably leftward, indicating reduced confidence in classifications under attack. Even with this decline, the hybrid MLP still

outperformed Logistic Regression after the FGSM attack, although it remained less robust than Random Forest.

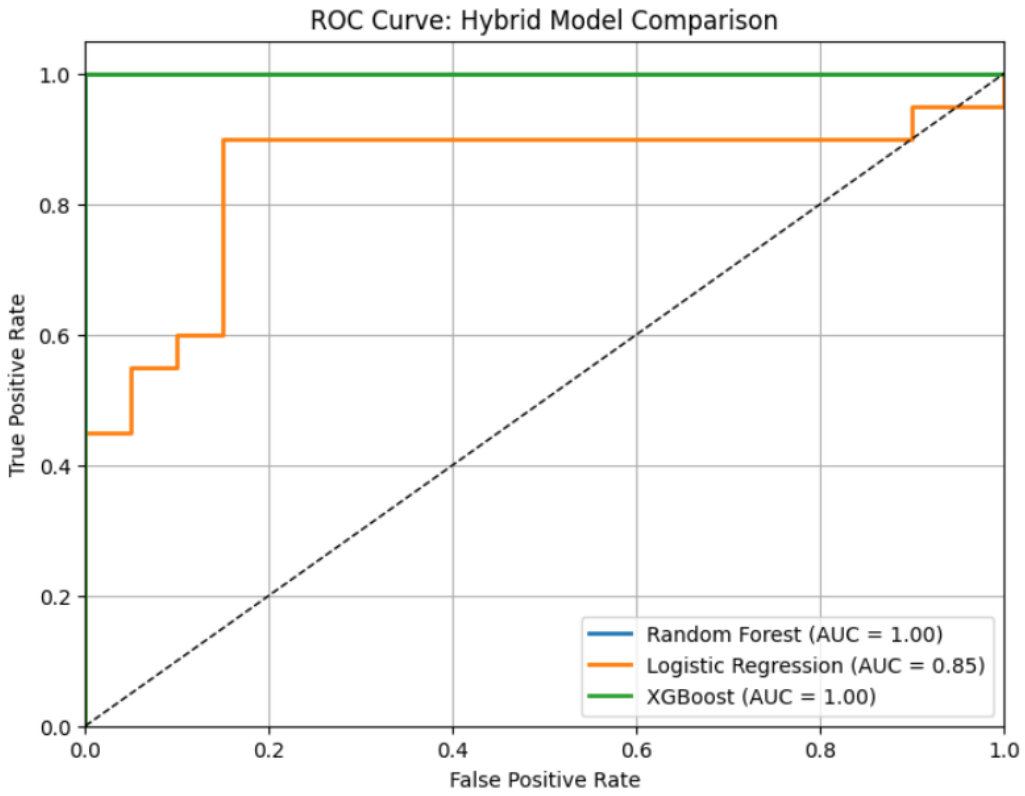

Figure 5: ROC curve for static model comparison, showing the trade-off between true positive rate and false positive rate for Random Forest, Logistic Regression and XGBoost classifiers for hybrid model.

```
Random Forest Accuracy after Advanced Adversarial Attack: 0.95
[[20  0]
 [ 2 18]]
Logistic Regression Accuracy after Advanced Adversarial Attack: 0.575
[[13  7]
 [10 10]]
XGBoost Accuracy after Advanced Adversarial Attack: 0.7
[[20  0]
 [12  8]]
```

Figure 6: Classification reports for hybrid model after adversarial attack.

```
Epoch 1/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - accuracy: 0.5556 - loss: 0.7570 - val_accuracy: 0.6250 - val_loss: 0.6226
Epoch 2/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7778 - loss: 0.5634 - val_accuracy: 0.8125 - val_loss: 0.5133
Epoch 3/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8958 - loss: 0.4360 - val_accuracy: 0.8125 - val_loss: 0.4301
Epoch 4/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.9375 - loss: 0.3446 - val_accuracy: 0.9375 - val_loss: 0.3617
Epoch 5/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9583 - loss: 0.2759 - val_accuracy: 0.9375 - val_loss: 0.3077
Epoch 6/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9583 - loss: 0.2196 - val_accuracy: 0.9375 - val_loss: 0.2654
Epoch 7/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9653 - loss: 0.1768 - val_accuracy: 0.9375 - val_loss: 0.2299
Epoch 8/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9722 - loss: 0.1425 - val_accuracy: 0.9375 - val_loss: 0.2014
Epoch 9/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9861 - loss: 0.1161 - val_accuracy: 0.9375 - val_loss: 0.1791
Epoch 10/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.9861 - loss: 0.0964 - val_accuracy: 0.9375 - val_loss: 0.1612
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9250 - loss: 7.1137
MLP Test Accuracy: 0.925000011920929
```

Figure 7: Training and Validation Performance of the Hybrid MLP Model

```
Clean AUC: 0.9025000000000001
Adv   AUC: 0.9775

Clean report:
              precision    recall  f1-score   support

           0      0.900     0.900     0.900        20
           1      0.900     0.900     0.900        20

    accuracy                          0.900        40
   macro avg      0.900     0.900     0.900        40
weighted avg      0.900     0.900     0.900        40


Adversarial report:
              precision    recall  f1-score   support

           0      0.947     0.900     0.923        20
           1      0.905     0.950     0.927        20

    accuracy                          0.925        40
   macro avg      0.926     0.925     0.925        40
weighted avg      0.926     0.925     0.925        40
```

Figure 8: FGSM Attack on the Hybrid MLP: Clean vs. Adversarial Performance

Comparative analysis across all models and attack scenarios confirmed the hybrid Random Forest as the most resilient configuration, consistently exhibiting the smallest performance gap between clean and adversarial conditions. Tree-based ensemble models, particularly Random Forest and XGBoost, consistently outperformed Logistic Regression in both accuracy and robustness across all domains. The MLP's performance drop under FGSM attacks highlighted the need for adversarially hardened training when deploying neural networks in real-world malware detection systems.

Limitations

Although the results of this research demonstrate the potential of hybrid static-dynamic feature fusion for improving malware detection accuracy and adversarial robustness, several limitations relating to the datasets, methodology and scope of the experiments must be acknowledged. In the initial project proposal, certain elements such as the inclusion of an additional dataset (NSL-KDD) live sandbox-based dynamic data capture and binary-level adversarial modifications were considered. Due to time constraints, ethical and safety concerns and computational resource limitations, these were not implemented in the final system. The resulting approach still addresses the core research question, but with certain practical trade-offs.

The first limitation concerns the datasets themselves. This study relied entirely on pre-extracted, publicly available benchmarks EMBER 2018 for static features and CIC-MalMem-2022 for dynamic features. While both datasets are well-established in malware research, they represent snapshots of threat landscapes at specific points in time. Malware families, feature distributions and attack techniques evolve rapidly, meaning that models trained on these datasets may not generalise effectively to newer or more diverse threats. The sample sizes were deliberately capped at 100 per class to ensure tractable experimentation; future work should extend the evaluation to larger subsets for improved generalisability. This would help confirm whether the strong clean-data performance observed here scales to more realistic dataset sizes. Additionally, the dynamic dataset contained only malicious entries, requiring the creation of synthetic benign samples generated by sampling within the observed feature ranges of malware. Although these synthetic entries were designed to be statistically plausible, they may fail to capture the subtle behavioural signatures of genuine benign software. This limitation may partly explain the exceptionally high clean-data performance of the dynamic-only and hybrid configurations results that might not be replicated if real benign behavioural traces were used.

Another limitation arises from the absence of live feature extraction. Dynamic features were obtained from pre-recorded behavioural traces rather than being captured in real time using sandbox environments such as Cuckoo Sandbox. While this approach removed the risks and computational demands associated with executing live malware, it also excluded environmental variability that can affect behavioural output, such as differences in system configuration, timing and trigger conditions. This divergence from the proposal where live sandboxing was envisioned was a conscious decision to prioritise reproducibility, time efficiency and operational safety, particularly given the legal and ethical implications of handling live malware in a non-isolated academic environment. Similarly, the static features were pre-engineered by the creators of the EMBER dataset. Although these features are extensive, they may not reflect indicators relevant to more recent or heavily obfuscated malware strains.

The scope of adversarial simulation also imposes constraints. The perturbations applied in this study were made directly in the feature space, guided by feature importance rankings and did not involve modifying executable binaries or runtime behaviours in a fully functional form. While effective for controlled testing, this method does not capture the complexity of real-world adversarial malware, which must maintain executable integrity, preserve payload functionality, and bypass multiple layers of runtime defence. Techniques such as code injection, section reordering and API call padding were represented here only in abstract, numerical form. In the proposal, binary-level adversarial generation was considered as a future enhancement. However, given the technical complexity and safety requirements this was deferred. Likewise,

the FGSM attack on the hybrid MLP operated entirely on numerical features without the structural constraints of executable formats, effectively making it a best-case scenario for an attacker.

A further limitation concerns the construction of the hybrid dataset. The EMBER and CIC-MalMem-2022 datasets do not contain both static and dynamic features from the same executable samples. As a result, the hybrid feature vectors were created by pairing static and dynamic samples of the same class malicious with malicious, benign with benign without guaranteeing that the features came from the same binary. This synthetic pairing preserves class consistency but does not capture the natural correlations that would exist in a true hybrid system, where specific structural characteristics often align closely with particular behavioural patterns. This limitation is compounded by the use of synthetic benign dynamic entries, creating an additional layer of divergence from real-world scenarios.

The study was also limited in terms of the models explored. The experiments focused on three traditional machine learning classifiers Logistic Regression, Random Forest and XGBoost along with a single feed forward neural network in the form of a Multi-Layer Perceptron. While these models cover different learning paradigms, they exclude more advanced deep learning architectures such as convolutional neural networks for raw binary embeddings, recurrent or transformer based models for sequential behavioural data or hybrid ensembles that combine classical and deep learning methods. Additionally, no adversarial training was applied, meaning that all models were evaluated in an un-hardened state, which likely contributed to the observed vulnerability of the MLP under FGSM attack.

Finally, operational considerations were outside the scope of this work. In practice, hybrid detection systems face challenges relating to computational cost and scalability, since they require the acquisition of both static and dynamic features. Collecting and processing these features is inherently slower and more resource-intensive than static-only scanning and could necessitate a tiered detection architecture in which static analysis serves as a rapid filter and dynamic analysis is reserved for high-risk or ambiguous cases. This study did not measure throughput, latency or scalability under large-scale operational loads nor did it address the legal, ethical and safety implications of handling live malware samples.

Taken together, these limitations highlight the need for future research to incorporate live, up-to-date datasets, realistic adversarial malware generation, true static-dynamic sample pairing, adversarially hardened training strategies and operational performance evaluations. Addressing these gaps would make it possible to develop hybrid detection systems that are both highly accurate and robust under real-world adversarial conditions.

Future Work

Building on the findings of this research, several future directions could be pursued to improve both the accuracy and robustness of hybrid malware detection systems, while directly addressing the limitations and trade-offs identified in this study. Many of these directions were considered in the original project proposal but were either postponed or adapted due to time, safety and resource constraints. Revisiting these elements in future work would bridge the gap between the intended scope and the implemented system.

One of the most immediate priorities is to incorporate live and contemporary datasets. Future experiments should capture both static and dynamic features from recent malware and benign

software in order to reflect the constantly evolving threat landscape. This would require developing live feature extraction pipelines capable of processing zero-day threats, polymorphic malware and file less attacks. For dynamic analysis, execution within modern sandbox environments such as Cuckoo Sandbox or Hybrid Analysis could produce richer and more varied behavioural traces than those available from pre-extracted datasets. This enhancement was envisioned in the original plan but was deferred in the final work to avoid the legal, ethical and operational risks of live malware handling in an academic environment. In addition, collecting genuine benign software samples from diverse categories would ensure that models learn realistic non-malicious behaviour patterns, removing the reliance on synthetic approximations that may not fully represent benign activity.

Another significant improvement would be to create datasets containing true static–dynamic pairings, where both types of features are extracted from the same executable instance. This would enable hybrid models to learn authentic correlations between code structure and runtime behaviour which could improve both predictive accuracy and interpretability. Such datasets would also allow detailed investigation into which combinations of static and dynamic indicators most strongly influence classification decisions, potentially guiding more targeted and efficient model designs.

Future adversarial evaluations should extend beyond controlled feature-space perturbations to include binary-level adversarial malware generation. This would involve implementing functional modifications to malicious code, such as manipulating import tables, reordering sections, inserting semantic no-operations, padding API call sequences, introducing conditional execution triggers or applying timing-based sandbox evasion techniques. These functional adversarial strategies were mentioned as a possible extension in the original proposal but were not implemented due to their technical complexity and safety implications. Incorporating them in future experiments would create a far more realistic measure of model resilience under operational attack conditions.

Defensive measures should also be explored, particularly adversarial training, in which perturbed or adversarially generated samples are incorporated into the training dataset to improve model robustness. Additional techniques such as defensive distillation, feature squeezing and input preprocessing filters could be assessed for their effectiveness in countering both gradient-based and heuristic evasion attacks. Comparing hardened and non-hardened versions of Random Forest, XGBoost, MLP and more advanced architectures could yield valuable insights into the most effective defensive strategies for each model type.

Expanding the range of model architectures is another promising direction. While this study focused on traditional machine learning methods and a simple feed-forward neural network, future work could investigate convolutional neural networks for learning directly from raw binary embeddings, recurrent or transformer-based architectures for capturing sequential dependencies in behavioural data and heterogeneous ensembles that integrate classical and deep learning outputs. Alternative feature fusion strategies, such as late fusion of predictions from separate static and dynamic models could be compared against the early fusion method used here to determine whether different approaches influence robustness or interpretability.

Operational feasibility should also be addressed through large-scale, real-time deployment testing. Such testing would measure detection throughput, latency and resource usage under realistic workloads, as well as explore tiered detection pipelines where static analysis serves as a rapid first stage and dynamic analysis is reserved for high-risk cases. Integration with

Security Information and Event Management (SIEM) or Endpoint Detection and Response (EDR) systems could also be tested to demonstrate practical applicability.

In the longer term, continuous learning mechanisms could be developed so that detection models adapt automatically to new and evolving threats. This might involve periodic retraining with updated threat intelligence feeds, establishing feedback loops from incident response teams, or applying federated learning approaches to allow distributed model updates without centralising sensitive data. Implementing these improvements would strengthen hybrid detection systems not only in controlled benchmarks but also in the unpredictable, adversarial conditions of live cybersecurity environments, fulfilling the operational vision initially proposed but scaled to meet real-world constraints.

Conclusion

This research set out to determine whether combining static and dynamic analysis features within a hybrid machine learning framework could improve malware detection accuracy and enhance adversarial robustness compared to single-domain approaches. The proposed system merged pre-extracted static features from the EMBER 2018 dataset with behavioural features from the CIC-MalMem-2022 dataset, integrating them through an early feature fusion strategy. Three configurations were developed: static-only, dynamic-only and hybrid and evaluated using Logistic Regression, Random Forest and XGBoost, with the hybrid configuration also incorporating a Multi-Layer Perceptron to enable gradient-based adversarial testing.

The evaluation revealed clear differences in performance across the three configurations. Dynamic-only models achieved near-perfect detection on clean datasets, reflecting the pronounced statistical differences between malicious and synthetic benign behavioural profiles. However, these models proved more vulnerable to targeted perturbations designed to conceal behavioural anomalies. Static-only models achieved lower absolute accuracy but displayed greater stability under certain adversarial manipulations, although they struggled with highly obfuscated or novel malware variants. The hybrid models consistently outperformed both static-only and dynamic-only approaches, with the hybrid Random Forest delivering the best overall balance of accuracy, stability and resistance to adversarial interference. Even when subjected to coordinated attacks targeting both static and dynamic features, this model experienced only minimal degradation in performance.

Adversarial testing highlighted important differences in classifier resilience. Logistic Regression, with its linear decision boundaries was consistently the most vulnerable, suffering substantial performance drops even under modest feature manipulations. Tree-based ensemble models, particularly Random Forest and XGBoost, maintained stronger robustness in both single-domain and hybrid settings. The hybrid MLP achieved high accuracy on clean data but was notably more affected by FGSM attacks, reinforcing the importance of adversarially aware training when deploying neural networks in operational malware detection systems.

Beyond its empirical results, this work contributes methodologically by providing a reproducible workflow for fusing heterogeneous feature domains, systematically applying feature-space adversarial attacks and evaluating robustness across multiple classifier types. However, its reliance on pre-extracted datasets, the use of synthetic benign generation for dynamic data

and the absence of live execution traces limit the direct generalisability of the results to real-world environments.

Overall, the findings support the principle that combining diverse indicators linking code-level structure with runtime behaviour creates stronger defences against adversarial evasion. A well-designed hybrid approach can capture the complementary strengths of static and dynamic analysis while mitigating their individual weaknesses, offering a more balanced and resilient basis for malware detection. Future research should focus on incorporating live and up-to-date data collection, realistic adversarial malware generation, adversarial training methods and large-scale deployment testing to close the gap between controlled experimental performance and operational cybersecurity requirement

References

[1] **EMBER dataset**
Anderson, H.S. and Roth, P. (2018) 'EMBER: An open dataset for training static PE malware machine learning models', *arXiv preprint arXiv:1804.04637*. Available at: https://arxiv.org/abs/1804.04637 (Accessed: 30 March 2025).

[2] **Adversarial evasion introduction**
Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G. and Roli, F. (2013) 'Evasion attacks against machine learning at test time', in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Springer, pp. 387–402. Available at: https://arxiv.org/abs/1708.06131 (Accessed: 30 March 2025).

[3] **Adversarial examples for malware**
Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P. (2017) 'Adversarial examples for malware detection', in *European Symposium on Research in Computer Security (ESORICS)*. Springer, pp. 62–79. Available at: https://arxiv.org/pdf/1702.05983.pdf (Accessed: 1 April 2025).

[4] **Deep learning on API call sequences**
Kolosnjaji, B., Zarras, A., Webster, G. and Eckert, C. (2016) 'Deep learning for classification of malware system call sequences', in *29th Australasian Joint Conference on Artificial Intelligence (AI)*. Springer, pp. 137–149. Available at: https://link.springer.com/chapter/10.1007/978-3-319-50127-7_11 (Accessed: 1 April 2025).

[5] **Malware detection with raw EXEs**
Raff, E., Barker, J., Sylvester, J., Brandon, R. and Catanzaro, B. (2018) 'Malware detection by eating a whole EXE', *arXiv preprint arXiv:1710.09435*. Available at: https://arxiv.org/abs/1710.09435 (Accessed: 31 March 2025).

[6] **Adversarial robustness**
Xu, B., Yang, H., Peng, X. and Shi, J. (2021) 'Enhancing adversarial robustness in malware detection', *IEEE Transactions on Information Forensics and Security*, 16, pp. 3140–3154. Available at: https://ieeexplore.ieee.org/document/9321695 (Accessed: 4 April 2025).

[7] **HAAMD hybrid analysis (Android)**
Choudhary, M. and Kishore, B. (2018) 'HAAMD: Hybrid analysis for Android malware detection', in *International Conference on Computational Intelligence and Communication*

Networks (ICCCI). IEEE. Available at: https://doi.org/10.1109/ICCCI.2018.8441295 (Accessed: 4 April 2025).

[8] **Hybrid family classification (Android)**
Dhalaria, M. and Gandotra, E. (2020) 'A hybrid approach for Android malware detection and family classification', *Journal of University Institute of Technology (JUIT)*. Available at: http://ir.juit.ac.in:8080/jspui/handle/123456789/9331 (Accessed: 4 April 2025).

[9] **Survey on ML for malware**
Ucci, D., Aniello, L. and Baldoni, R. (2019) 'Survey of machine learning techniques for malware analysis', *Computers & Security*, 81, pp. 123–147. Available at: https://doi.org/10.1016/j.cose.2018.11.001 (Accessed: 5 April 2025).

[10] **CNN-BiLSTM for code detection**
Zhang, H. and Wang, Y. (2024) 'A novel approach to malicious code detection using CNN-BiLSTM', *arXiv preprint arXiv:2410.09401*. Available at: https://arxiv.org/pdf/2410.09401 (Accessed: 8 April 2025).

[11] **Robust malware detection models**
Rathore, H., et al. (2021) 'Robust malware detection models: Learning from adversarial attacks and defenses', *Forensic Science International: Digital Investigation*, 37, p. 301183. Available at: https://dfrws.org/wp-content/uploads/2021/09/2021-usa-paper-04-robust_malware_detection_models_learning_from_adversarial_attacks_and_defenses.pdf (Accessed: 7 April 2025).

[12] **Malware detection at the edge with LLMs**
Rondanini, C., Carminati, B., Ferrari, E., Gaudiano, A. and Kundu, A. (2025) 'Malware detection at the edge with lightweight LLMs: A performance evaluation', *arXiv preprint arXiv:2503.04302*. Available at: https://arxiv.org/pdf/2503.04302 (Accessed: 8 April 2025).

[13] **DNN-based malware detection**
Saxe, J. and Berlin, K. (2015) 'Deep neural network based malware detection using two-dimensional binary program features', *arXiv preprint arXiv:1508.03096*. Available at: https://arxiv.org/abs/1508.03096 (Accessed: 5 April 2025).

[14] **Hybrid CNN-BiLSTM-DNN for IoT threats**
Agbor, B.A., Stephen, B.U.-A., Asuquo, P., Luke, U.O. and Anaga, V. (2025) 'Hybrid CNN BiLSTM–DNN approach for detecting cybersecurity threats in IoT networks', *Computers*, 14(2), p. 58. Available at: https://www.mdpi.com/2073-431X/14/2/58 (Accessed: 7 April 2025).