**COS70008 – Technology Innovation Project and Research**

*Developing a Web Based System for predicting and analysing Malicious Attacks using a Hybrid Machine Learning Model*

**Assignment – 2**

**Research Report and Project Brief**

**Student Name: Arun Ragavendhar Arunachalam Palaniyappan**

**Student ID: 104837257**

**Date: 30 / 03 / 2025**

Contents

[**1. Literature Review** 3](#_Toc193918671)

[**2. Methodology** 4](#_Toc193918672)

[**Phase 1: Dataset Preprocessing** 4](#_Toc193918673)

[**Phase 2: Model Training** 4](#_Toc193918674)

[**Phase 3: Hybrid Pipeline Integration** 5](#_Toc193918675)

[**Phase 4: Web-Based System Deployment** 5](#_Toc193918676)

[**3. Background of the Project** 6](#_Toc193918677)

[**4. Project Goals and Objectives** 6](#_Toc193918678)

[**5. Desired Outcomes and Benefits** 6](#_Toc193918679)

[**6. Learning issue** 6](#_Toc193918680)

[**7. Project Scope and Exclusions** 6](#_Toc193918681)

[**8. Project Deliverables** 6](#_Toc193918682)

[**9. Project Management Plan** 7](#_Toc193918683)

[**9.1 Timeline** 7](#_Toc193918684)

[**9.2 Milestones** 7](#_Toc193918685)

[**9.3 Team Roles** 7](#_Toc193918686)

[**10. Conclusion** 7](#_Toc193918687)

[**11. Appendix** 8](#_Toc193918688)

[**11.1 Abbreviations** 8](#_Toc193918689)

[**11.1 List of Figures and Tables** 8](#_Toc193918690)

[**12. References** 8](#_Toc193918691)

**Word Count (excluding Table of Contents, References, etc.): 1776**

# **1. Literature Review**

The initial literature review explored traditional malware detection methods, machine learning classifiers, anomaly detection models, and publicly available datasets. It helped outline the main challenges in detecting both known and unknown malware threats. One key finding was that hybrid AI models—combining supervised classification for known threats with unsupervised anomaly detection for unknown or zero-day attacks—can offer a more complete detection strategy. This review builds on that by comparing model performance, structure, and real-world test results to identify suitable algorithms and datasets for the project.

Alazab et al. (2022) tested Random Forest and other classifiers on the Microsoft Malware Classification Challenge dataset. Random Forest achieved 97.15% accuracy and an F1-score of 0.963, helped by its ensemble-based decision trees and use of Gini impurity for feature splits. On the other hand, Dastbaz et al. (2021) reported that Support Vector Machines (SVMs) dropped below 90% accuracy as the number of classes increased, showing limitations in scaling. Yang et al. (2023) tested XGBoost on the EMBER dataset and reached 98.4% accuracy, but the model needed heavy tuning and was sensitive to class imbalance. These issues, along with long training times, make it less practical for systems that need faster or lighter performance.

In unsupervised learning, Autoencoders were identified as a useful option for anomaly detection. These models compress input data through an encoder and rebuild it through a decoder. Anomalies are found by measuring the difference between input and output. Abubakar et al. (2022) showed that this method could detect zero-day malware with 92.6% accuracy and a false positive rate under 3%. Other unsupervised models were also reviewed. Huda et al. (2021) tested Isolation Forest, which reached 85% accuracy but showed unstable results when dealing with complex malware patterns. Rashid et al. (2023) noted that One-Class SVM caused frequent false positives, making it less reliable for practical use.

To overcome the limits of single-model systems, some researchers developed hybrid architectures. Patel et al. (2023) proposed a pipeline that used an Autoencoder to score anomalies and then passed the data to a Random Forest classifier. This hybrid model recorded a 95.3% F1-score and reduced false negatives by 22% compared to using either model alone. Chen et al. (2022) used a similar setup in a cloud environment and showed that the layered structure helped with faster learning and better detection of unknown threats.

Dataset quality also plays a major role in model performance. Among available options, CIC-MalMem2022 from the Canadian Institute for Cybersecurity was noted for including both static and dynamic memory features from real malware samples. This gives it an advantage over datasets like EMBER, which only includes static PE file data, or ADFA-LD, which lacks variety and scale. CIC-MalMem2022 supports both anomaly detection and multi-class classification tasks, making it a strong match for hybrid model training.

Overall, the review supports the use of a combined Autoencoder and Random Forest model for building a hybrid detection system. CIC-MalMem2022 is identified as a suitable dataset for this approach. This model is one of five designs proposed for development and client review.

# **2. Methodology**

This project adopts a methodology based on Creswell’s (2014) mixed-method research framework. It integrates both quantitative and qualitative elements and is divided into four main phases:

## **Phase 1: Dataset Preprocessing**

The selected dataset is CIC-MalMem2022, which contains 58,596 labelled samples with 55 numerical features from both static and dynamic memory analysis. The data represents behaviours of both benign and malicious programs. Preprocessing steps include handling missing values, removing inconsistencies, and standardising features using Scikit-learn’s StandardScaler. Labels such as 'Trojan', 'Backdoor', 'Ransomware', and 'Benign' are encoded into numerical format using LabelEncoder to prepare the data for machine learning. (Canadian Institute for Cybersecurity, 2022).

## **Phase 2: Model Training**

***Autoencoder (Anomaly Detection)***

An Autoencoder is to be built using TensorFlow/Keras. It includes an input layer (55 features), two hidden encoding layers, a bottleneck layer, and a decoder. The model is to be trained only on benign data. Anomalies will be detected using the reconstruction error:

***L = ||x − x̂||² = ||x − g(f(x))||²’***

*Where:*

* *x is the original input*
* *f(x) is the encoded version*
* *g(f(x)) or x̂ is the reconstructed output*
* *L is the reconstruction error (squared distance)*

A threshold T will be set using the mean (μ) and standard deviation (σ) of reconstruction errors on validation *data:*

***T = μ + 2σ***

If the reconstruction error L is greater than T, the sample is flagged as anomalous.

***Random Forest (Classification)***

A Random Forest model will be trained on the full labelled dataset using Scikit-learn, with 100 decision trees and entropy as the splitting criterion. 10-fold stratified cross-validation will be used to validate the model. The algorithm selects features by minimising Gini impurity:

***G = 1 − Σ(pi²)***

*Where:*

* *C is the number of classes*
* *pi is the proportion of samples from class i at a node*

Lower Gini values indicate better splits. The model improves classification by selecting splits that reduce impurity across child nodes. Predictions that exceed the anomaly threshold are logged for further review (Alazab et al. 2022).

## **Phase 3: Hybrid Pipeline Integration**

The hybrid pipeline uses a sequential setup. Incoming data is first passed through the Autoencoder. If the reconstruction error is within the threshold, the data is forwarded to the Random Forest model for classification. Patel et al. (2023) showed that this sequence can reduce false negatives by up to 30%, helping the system detect more threats that would otherwise be missed.

## **Phase 4: Web-Based System Deployment**

The trained models will be integrated into a Flask backend. RESTful APIs can handle prediction requests, login/authentication, history access, and report downloads. The frontend can be built using React.js. Axios can be used to send and receive data between the frontend and backend. All predictions, inputs, and logs is to be stored in a MySQL database. The system will include role-based access to protect user data. According to Chen et al. (2022), similar setups achieved prediction response times below 500ms, which is acceptable for use in academic and enterprise environments.

# **3. Background of the Project**

Technology is evolving, and so are cyber threats. Traditional malware detection methods, which mostly rely on fixed patterns or signatures, are becoming less effective. These systems often struggle to detect new, unknown, or modified types of malwares and fail to accurately predict or analyse a cyber physical system’s behaviour (CPS). This creates a gap between the way threats behave and how they are detected.

This project looks at how artificial intelligence can help bridge that gap, through the learnings from the literature reviews. The focus is on using machine learning to build a web-based system with a hybrid model to analyse and detect different types of malicious attacks as well as to predict and analyse the behaviour of CPS. Instead of relying on a single method, the project explores how combining different AI models can lead to more flexible and accurate detection.

The goal is to offer multiple unique solutions by designing and testing different combinations of hybrid machine learning models. The challenge lies in identifying useful data and building open-ended applications that apply these models in practical ways. This approach supports better analysis of threat behaviours and helps improve the ability to respond to evolving attacks.

# **4. Project Goals and Objectives**

The goal of this project is to build a web-based system that can detect and analyse different types of malicious attacks using hybrid machine learning models. It also aims to support behaviour analysis by identifying unusual patterns in cyber-physical systems.

To achieve this, the project sets the following objectives:

* **Dataset Identification and Analysis:** Evaluation and preprocessing publicly available datasets relevant to various threat types. These include CIC-DDoS2019, MalwareBazaar, CIC-MalMem2022, Drebin Lite, and phishing datasets from UCI and Kaggle, selected for their diversity in static, dynamic, and behavioural features.
* **Hybrid Model Development:** Designing and training multiple hybrid machine learning pipelines combining supervised and unsupervised techniques. The 5 proposed designs:
  + Random Forest + PCA
  + CNN-LSTM
  + Autoencoder + Random Forest
  + TF-IDF + Logistic Regression
  + DistilBERT + Random Forest or LSTM
* **Cyber-Physical Behaviour Modelling:** Applying ML techniques to detect abnormal patterns in system behaviour, combining classification for known threats with anomaly detection for zero-day or unknown threats.
* **System Design and Prototyping:** Creation of five Figma prototypes representing different malware use cases and system designs:
  + **Design 1 (Klaus):** DDoS – Random Forest + PCA (CIC-DDoS2019)
  + **Design 2 (Dream):** Static/Dynamic Malware – CNN-LSTM (MalwareBazaar + Drebin Lite)
  + **Design 3 (Arun):** Malware Classification + Anomaly Detection – Autoencoder + Random Forest (CIC-MalMem2022)
  + **Design 4 (Vatana):** SMS Phishing – LSTM or DistilBERT (TBD)
  + **Design 5 (Ted):** URL Phishing – TF-IDF + Logistic Regression

From these, **three designs will be selected** for full implementation. One will be developed to **Minimum Viable Product (MVP)** level with end-to-end functionality including user interaction, detection, history tracking, and administrative control.

* **Development and Implementation:** Developing 3 three tier, full-stack, client-server web application having:
  + Frontend: React.js
  + Backend: Flask (Python)
  + Database: MySQL, PostgreSQL, or SQLite
  + ML Middleware: TensorFlow/Keras, Scikit-learn, etc.

# **5. Desired Outcomes and Benefits**

This project is expected to deliver the following outcomes:

* **Five Figma Design Prototypes**, each representing a unique malware detection use case with distinct datasets, models, and features.
* **Three Full-Stack Implementations**, built from the selected prototypes, integrating trained hybrid machine learning models with a functioning web-based system.
* **One MVP (Minimum Viable Product)**, featuring:
  + Real-time detection through the hybrid ML pipeline
  + Role-based access (user/admin)
  + File/hash/manual input options
  + Prediction results with classification and anomaly scores
  + History tracking and downloadable reports (CSV)
  + Admin tools for log monitoring, user management, and basic analytics

The benefits of this project include:

* **Practical application of machine learning** to real-world cybersecurity problems, especially malware detection and behaviour analysis.
* **Experience with full-stack AI system deployment**, covering dataset handling, model training, API development, and secure integration.
* **Development of usable tools** for individuals or organisations to support early detection and analysis of malicious activity.
* **Foundation for future research or product development**, by demonstrating working AI-based detection pipelines integrated into scalable web applications.

# **6. Learning issue**

The learning issue in this project is related to developing and integrating a hybrid machine learning model into a functional three-tier web-based malware detection system. For the assigned individual part of the project (Design 3), the task is to implement a system that uses an Autoencoder for anomaly detection and a Random Forest classifier for known malware classification.

To begin with, it is necessary to understand how to handle the dataset. This includes preprocessing steps such as removing null values, scaling numerical features, encoding labels, and preparing separate data subsets for training the models. The Autoencoder is to be trained on benign data to learn normal patterns, while Random Forest is to be trained on all labelled samples to support multi-class classification. A threshold is proposed to be set based on the reconstruction error for detecting anomalies.

In terms of design, a feature-based prototype is to be created using Figma. This design is planned to include user-side features such as login, file/manual input, prediction results, history tracking, and downloadable reports. Admin-side features are proposed to include detection logs, user management, and basic system analytics. Each feature is to be mapped to backend logic and supported by database operations.

The system is proposed to be implemented using a three-tier architecture: React.js for the frontend, Flask for the backend, and MySQL for the database. It is to be learnt how to send prediction inputs from the frontend to the backend using API calls (Axios), process them through the trained models, and return classification results and anomaly scores. The results are to be stored with timestamps and user metadata. Other functions such as CSV report generation, secure session handling, and role-based access are also planned to be implemented.

The overall learning requirement is to understand how to build and connect all parts of the system—data pipeline, model logic, user interface, API communication, and database storage—into a single working application. This includes learning how hybrid models can be used in real-time detection and how such a system can be designed to support both users and administrators in analysing malicious threats.

# **7. Project Scope and Exclusions**

The scope of this project includes the design of five hybrid malware detection systems. Out of these, three are selected for full implementation. One of the three is developed into a Minimum Viable Product (MVP) with complete end-to-end functionality. The MVP includes user login, file or manual input, real-time prediction using a hybrid model (Autoencoder + Random Forest), result display, prediction history, CSV report download, and admin features such as user management, detection logs, and system analytics. The system is built using a client-server architecture with React.js, Flask, and MySQL.

Some features were explored during the planning phase but excluded from the current version due to time and delivery constraints. These include dynamic sandboxing, transformer-based models (such as BERT and DistilBERT), multi-level admin roles, real-time alerts, and PDF report generation. These features required more time, model tuning, or infrastructure than the project allowed.

The scope focuses on delivering the core functionality and ensuring the system can run end-to-end. The current version is structured to serve as a base for future additions and improvements if required.

# **8. Project Deliverables**

The project will deliver key outputs across the design, development, and delivery phases:

* **Five Figma Design Prototypes (by Week 6):** Visual concepts representing five hybrid malware detection systems. Each prototype addresses a different attack type, dataset, and model combination. Feedback is gathered and incorporated before finalisation.
* **Three Full-Stack Detection Systems (by Week 9):** Based on selected designs, three systems will be implemented with complete frontend, backend, and integrated machine learning models.
* **One Minimum Viable Product (by Week 12):** A fully functional MVP including login, input (file/manual), real-time hybrid detection, prediction history, CSV report generation, and admin tools.
* **Final Presentation and Demo (Week 12):** Group demonstration covering system features, MVP walkthrough, and outcomes.
* **Individual Submissions:** Each member will submit a research report, innovation concept, and individual contribution report.

# **9. Project Management Plan**

This project follows a Waterfall-based management approach, with fixed timelines and defined deliverables. Each phase is completed before the next one begins. This helps the team stay on track and work through tasks in order. The approach is chosen to match the academic setting, fixed semester timeline, and the need to deliver prototypes, system builds, and reports in a structured way.

## **9.1 Timeline**

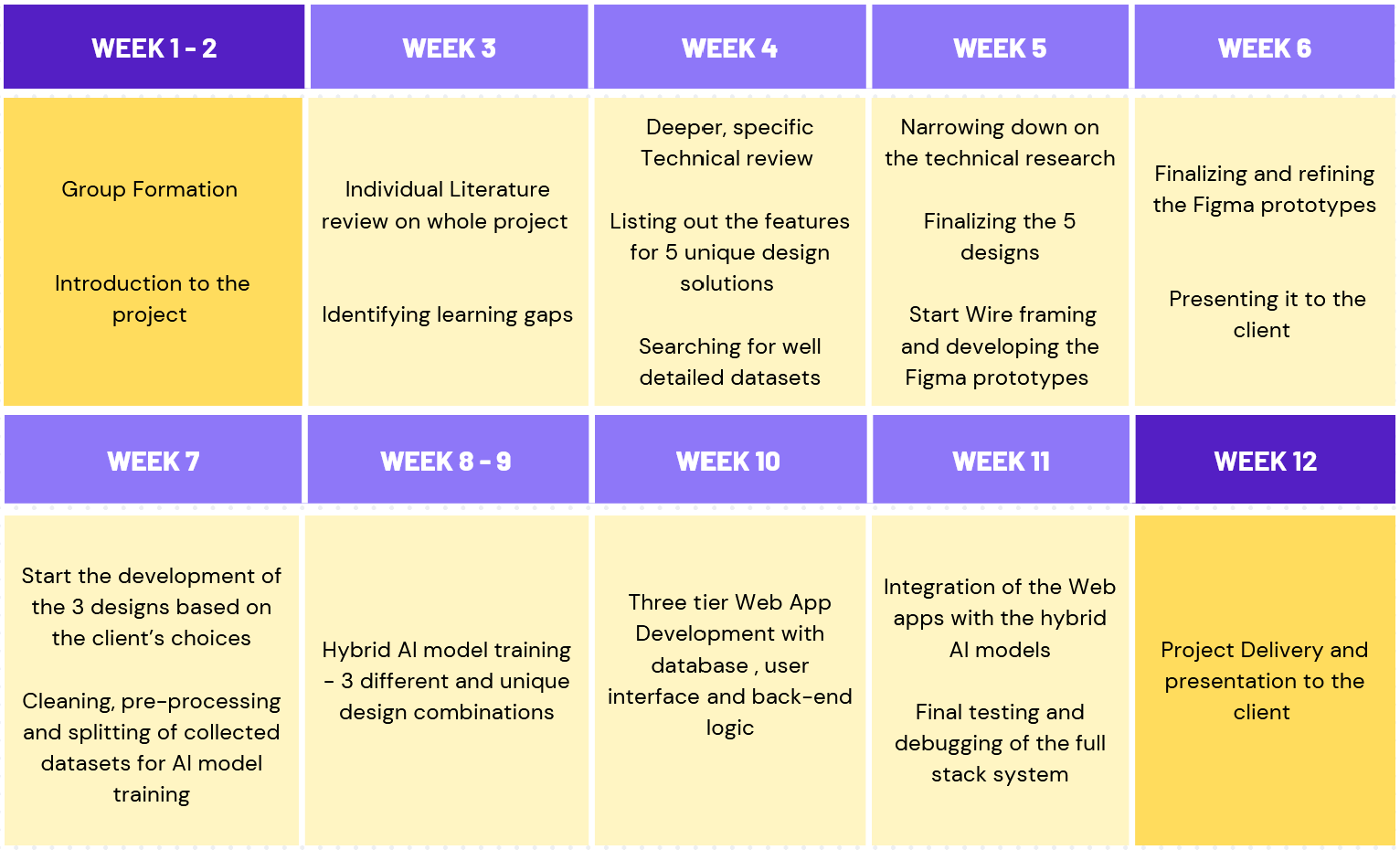


Figure :1 Ghantt chart laying out the project Timeline

## **9.2 Milestones**

**Week 6:** Final prototype submission

**Week 9:** System implementation and model validation complete

**Week 11:** MVP testing finalised

**Week 12:** Final presentation and report submission

## **9.3 Team Roles**

**Project Lead:** Backend development, ML integration, and report coordination

**Frontend Developer:** React.js UI development and system responsiveness

**ML Researcher:** Dataset preprocessing, model design, and evaluation

**QA/Documentation:** System testing, bug tracking, reporting, and presentation support

# **10. Conclusion**

This project addresses the limits of traditional malware detection by developing a web-based system that uses a hybrid machine learning model. The system combines an Autoencoder to detect unknown or unusual behaviour and a Random Forest classifier to identify known types of malware. This two-step process is used to detect both familiar and unfamiliar threats.

The approach includes dataset preparation, model training, system integration, and deployment using Flask, React.js, and MySQL. Input data is first checked by the Autoencoder. If no anomaly is found, the data is passed to the Random Forest model for classification.

The project is planned to deliver five design prototypes, three full system implementations, and one Minimum Viable Product (MVP). The MVP includes login, input upload, prediction display, history tracking, and admin functions. The aim is to show how machine learning models can be applied in a working system to support malware detection and help users review threat behaviour. The project links research findings with practical system development and can be used as a base for future extensions.

# **11. Appendix**

## **11.1 Abbreviations**

MVP: Minimum Viable Product

API: Application Programming Interface

UI: User Interface

CSV: Comma-Separated Values

RESTful: Representational State Transfer

PE: Portable Executable

RF: Random Forest

Flask: Python-based Web Framework

React: JavaScript Frontend Library

MySQL: Relational Database Management System

## **11.1 List of Figures and Tables**

Figure :1 Ghantt chart laying out the project Timeline

# **12. References**

1. Abubakar, A. I., Pranggono, B., & Hussain, S. (2022). Deep learning models for zero-day malware detection using memory dump analysis. *Journal of Network and Computer Applications, 202*, 103356. https://doi.org/10.1016/j.jnca.2022.103356
2. Alazab, M., Awajan, A., Abd El-Latif, A. A., Khan, S., & Jolfaei, A. (2022). Intelligent forensic techniques for classifying and analyzing malware variants using supervised learning algorithms. *Future Generation Computer Systems, 129*, 213–225. https://doi.org/10.1016/j.future.2021.11.008
3. Canadian Institute for Cybersecurity. (2022). *CIC-MalMem2022 dataset*. University of New Brunswick. https://www.unb.ca/cic/datasets/malmem-2022.html
4. Chen, L., Zhang, M., & Wang, Y. (2022). Hybrid deep learning system for anomaly-based malware detection in cloud environments. *IEEE Transactions on Cloud Computing, 10*(1), 145–157. https://doi.org/10.1109/TCC.2020.2967896
5. Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches* (4th ed.). Sage Publications.
6. Dastbaz, M., Mahmud, R., & Hamdulla, H. (2021). Evaluation of support vector machine and random forest in malware classification. *Computers & Security, 108*, 102345. https://doi.org/10.1016/j.cose.2021.102345
7. Huda, S., Ahmed, E., & Alrubaian, M. (2021). Lightweight anomaly detection in edge-enabled IoT networks using isolation forests. *Sensors, 21*(7), 2494. https://doi.org/10.3390/s21072494
8. Patel, D., Sharma, R., & Jaiswal, R. (2023). Combining Autoencoders and Random Forests for robust malware detection. *ACM Transactions on Cybersecurity, 11*(2), Article 22. https://doi.org/10.1145/3576123
9. Yang, H., Liu, X., & Gao, J. (2023). Feature fusion and XGBoost-based malware detection on the EMBER dataset. *Computers, Materials & Continua, 75*(3), 541–559. https://doi.org/10.32604/cmc.2023.029468