COS70008 – Technology Innovation Project and Research

Designing and Building Hybrid AI Web Systems for Malware Detection and Cyber Threat Analysis

Assignment – 2

Research Report and Project Brief

Student Name: Arun Ragavendhar Arunachalam Palaniyappan

Student ID: 104837257

Date: 30 / 03 / 2025

Contents

Word Count (excluding Table of Contents, References, etc.): 2682

PART: A

1. Literature Review

This project explores how AI can improve malware detection, especially for both known and unknown threats. Previous research shows that using existing signature-based methods or a single AI model—either a classifier or an anomaly detector—is often not enough. Instead, hybrid models that combine multiple techniques have shown stronger performance and broader detection coverage. This review compares how different supervised, unsupervised, and hybrid models performed in recent studies and explains what technical insights can guide the design and development of the detection systems in this project.

In **supervised learning**, Random Forest stood out. **Alazab et al. (2022)** tested it alongside Decision Trees and K-Nearest Neighbours using the Microsoft Malware dataset, achieving 97.15% accuracy and an F1-score of 0.963. This was due to its ability to combine multiple decision trees and identify strong features. **Dastbaz et al. (2021)**, however, found that Support Vector Machines (SVMs) dropped below 90% accuracy as the number of malware classes increased, showing SVMs can struggle with scaling. **Yang et al. (2023)** used XGBoost on the EMBER dataset, reaching 98.4% accuracy, but the model required heavy tuning and was sensitive to class imbalance, limiting its use in fast-response systems.

Unsupervised models like Autoencoders were more effective for detecting new or unknown threats. These models learn from benign data, and anything that doesn't match is flagged as an anomaly. **Abubakar et al. (2022)** achieved 92.6% accuracy and under 3% false positives using this method. **Huda et al. (2021)** tested Isolation Forests and got 85% accuracy, but results became inconsistent with complex malware patterns. **Rashid et al. (2023)** found One-Class SVMs produced too many false positives, reducing reliability.

Hybrid models help address these gaps. **Patel et al. (2023)** combined an Autoencoder and Random Forest, where the Autoencoder filtered anomalies and Random Forest classified the result. This approach cut false negatives by 22% and scored a 95.3% F1-score. **Chen et al. (2022)** applied a similar method in cloud settings, improving both detection and speed due to the layered structure.

Dataset choice also mattered. CIC-MalMem2022 was identified as the most suitable for hybrid training, as it includes both static and dynamic memory features from real malware. EMBER includes only static data, while ADFA-LD lacks variety. CIC-MalMem2022 supports both classification and anomaly detection tasks.

Alsharafi et al. (2023) also showed how AI models can be used inside a **web-based tool**. They built a system where users upload files through a web page, and a 1D-CNN model checks for malware. The system was built using Python and Flask, and it worked well in real-time. This study highlights that AI detection doesn't need to stay in research—it can run inside everyday web applications, helping users interact with the model easily.

Overall, hybrid AI models offer the best balance between detecting known malware and identifying new threats. Based on these findings, this project adopts multiple hybrid AI combinations to improve accuracy, reliability, and responsiveness.

2. Methodology

This section focuses on the testing methods researchers used to evaluate AI-based malware detection. Each study selected techniques based on the type of threat and dataset involved.

Alazab et al. (2022) evaluated several supervised models like Random Forest and Naïve Bayes, training them on labelled datasets. They used metrics like accuracy, precision, recall, and F1-score, and selected features using Gini impurity. Results were validated using cross-validation. **Dastbaz et al. (2021)** examined scalability by increasing the number of malware classes, finding that SVMs lost accuracy with more complex labels.

Yang et al. (2023) applied XGBoost with feature fusion on the EMBER dataset. They used hyperparameter tuning and class weights to handle imbalance. Their evaluation showed the model was accurate but harder to generalise. In contrast, unsupervised models used different methods.

Abubakar et al. (2022) trained Autoencoders only on clean data. Anomalies were flagged if reconstruction error exceeded a statistical threshold based on validation error. **Huda et al. (2021)** studied Isolation Forests in resource-constrained environments like IoT. Their results were acceptable but unstable. **Rashid et al. (2023)** tested One-Class SVMs but noted high false positives, making them unsuitable for real-world use.

The most reliable results came from hybrid models. **Patel et al. (2023)** used a two-stage approach—Autoencoder for anomaly detection, followed by Random Forest for classification. This method improved accuracy and reduced missed threats. **Chen et al. (2022)** implemented a similar hybrid system in the cloud and tested its speed, scalability, and detection rates with positive outcomes.

Most papers included preprocessing steps like cleaning data, scaling, and converting labels. Datasets like CIC-MalMem2022, EMBER, and the Microsoft Malware Challenge were widely used because they support both classification and anomaly detection.

In their setup, **Alsharafi et al. (2023)** used Flask for the backend and basic web tools like HTML, CSS, and JavaScript for the frontend. Their malware detection model was trained in Python and linked to the web interface using APIs. The app was containerized using Docker, making it easy to run on different systems. These choices show how lightweight tools can support AI in real-time systems and guide this project's web app structure.

These methods helped identify what model combinations worked best in different scenarios. The findings now guide this project in selecting and evaluating hybrid AI setups that are practical for real-time malware detection.

PART: B

3. Background of the Project

Cyber threats are evolving and becoming more complex, making traditional malware detection tools less effective. Many of these tools depend on fixed signatures or rule-based systems, which often miss new or modified malware. Techniques like code obfuscation and fileless attacks now help threats

bypass older tools, especially in cyber-physical systems (CPS) where software directly controls physical devices. These gaps create a need for smarter, more adaptable detection methods.

This project looks at how artificial intelligence can close that gap. The literature review shows that hybrid models combining supervised and unsupervised learning can improve accuracy, handle new threats better, and reduce false alerts. Instead of using one model type, this approach uses multiple different combinations of machine learning models to achieve both classification and anomaly detection, to flag unknown behaviours while also identifying known malware. Building a web-based system using hybrid AI models to detect and analyse various threats, including those affecting CPS, allows for better threat visibility and response in real-world environments.

The idea 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 using hybrid AI models to detect and analyse different types of cyberattacks. It focuses on model design, system integration, and behaviour analysis, divided into three main goals.

The **first goal** is to create five hybrid AI model designs, each designed to detect threats like DDoS attacks, static or dynamic malware, and phishing through URLs or SMS. One model is intended to combine anomaly detection with classification, enabling it to identify both known and unknown threats. Each setup will use a unique and distinct combination of machine learning techniques and be trained on a dataset suitable for the type of threat it aims to handle.

The **second goal** is to test how these models work in different system formats. **Five Figma prototypes** will be designed, showing how users interact with each system. Each model will focus on detecting a different type of threat, ensuring clear differences in what each system identifies and analyses. **Three designs** are to be chosen by the client to be implemented as **full stack web applications** using a three-tier structure. While all systems follow the same layered setup, the backend integration may vary—some using APIs, others following an MVC pattern.

The **third goal** is to explore behaviour-based detection. This means observing how programs act in controlled environments and using machine learning to spot actions that deviate from normal behaviour. This helps detect threats that aren't easily visible through static file analysis, especially in smart devices or physical control systems.

Together, these goals support detection, analysis, and system design by applying AI to different threat types in flexible, testable ways.

5. Desired Outcomes and Benefits

This project delivers outcomes that align directly with its three main goals:

From Goal 1, five hybrid AI models will be designed and presented, each using a different machine learning setup to target distinct threats like malware, phishing, or DDoS. This gives the system broader and stronger threat coverage. Each model is trained on a matching dataset and tested to confirm accuracy and usefulness. These models can also be reused or adapted for future tools. **From Goal 2**, five Figma prototypes will show how each model would work in a real system if implemented. Out of these, three will be fully built and tested. This helps compare system designs and makes it easier to reuse the setups in future developments. **From Goal 3**, behaviour-based detection will be applied to one or more systems. This approach flags threats by observing what programs do, not just how they look. It helps catch hidden or unknown attacks, which is especially useful in cyber-physical systems like smart homes or factories.

Together, these results will show how AI can be used to build usable, secure systems for real-time threat detection. They also provide a foundation for future research or product development in cybersecurity.

6. Learning issue

The individual learning issue in this project is how to design and integrate a hybrid machine learning model for detecting malware and analysing behaviour in cyber-physical systems (CPS), within a complete, real-time web application. Unlike models that exist only in research or isolated test environments, this work focuses on building a system that runs end-to-end—from taking user input to delivering threat analysis results—all within a fully working web platform.

One of the proposed hybrid setups uses an Autoencoder to detect abnormal behaviour and a Random Forest to classify known malware. The Autoencoder is trained on normal CPS activity and flags anomalies based on reconstruction error. If an input is within normal bounds, it moves to the Random Forest model, which classifies the threat using labelled data. This structure is based on research showing that combining anomaly detection with classification reduces missed threats and improves overall coverage.

Other model combinations are also planned as part of the learning process. These include CNN-LSTM to detect evolving malware behaviour, Random Forest with PCA for identifying DDoS patterns, TF-IDF with Logistic Regression for phishing URL detection, and DistilBERT with LSTM for analysing SMS-based phishing. Each pairing is matched to a specific threat type and data format, requiring different preprocessing, model architecture, and evaluation strategies. Understanding the logic and limitations behind each combination is part of the core technical learning.

These models are to be embedded in a full-stack, three-tier web system. The learning includes using a frontend JavaScript framework (like React.js or Vue.js), python libraries for model training, and an SQL database for managing inputs, results, and logs. The user interface is to support login, file or text input, real-time detection results, and history tracking. Admin tools are also included for viewing detection logs and managing users.

The literature reviewed informs these decisions—helping select suitable models, define thresholds, and match detection logic with practical system requirements. Autoencoders are effective for flagging zero-day threats, while supervised classifiers like Random Forest and Logistic Regression are well-suited for identifying known attack patterns. This learning issue is being addressed by studying how different models behave across use cases and building them into working, web-based detection systems that respond in real time to real-world threats.

7. Project Scope and Exclusions

This project focuses on designing five hybrid malware detection systems, each using different AI model combinations and datasets tailored to specific attack types. From these, three are to be fully developed into web-based applications that users can interact with. Each system includes a web interface, a trained hybrid model for detecting, classifying, and predicting threats, and a SQL-based backend for storing inputs and results. The frontend is built using a suitable JavaScript framework, while the AI models are developed using Python libraries. All systems are based on real datasets and deliver functional prototypes that clearly demonstrate how the application works.

Features include secure registration and login for both admin and user, executable file upload, URL upload for malware detection, malware classification and anomaly detection, prediction results history, phishing detection, exportable CSV export, notifications alert to users and data visualization dash boards.

Some advanced features were explored but excluded due to the short six-week timeline. These include dynamic sandboxing, transformer-based models, multi-level admin roles. They require more setup, tuning, or testing than the current schedule allows. Also, deploying the web apps to a cloud-based solution like AWS was considered, but was then excluded as they exceed the current budget as well as time constraint of the project. The web apps are currently intended to run and operate on the localhost.

Other future upgrades, if funded, adaptive AI (6-12 months), improved reliability (9-12 months), third-party integration (6+ months). The current version focuses on building a working prototype with core detection functions. But the idea is to design it to support upgrades in future phases.

8. Project Deliverables

The project will deliver three fully working malware detection systems, each based on a unique hybrid AI model.

These are to be developed from **five Figma prototypes designed by Week 6**, showing UI flow and system logic. These are to be submitted to the client, who would then choose three designs to be implemented. All three implementations will follow a three-tier architecture—with suitable JavaScript Framework based front ends, python-based AI models and backend, and SQL based databases. The complete development and internal testing with all the proposed features are to be finished by Week 11.

In **Week 12, the final presentation to be client** will show system features and live predictions. Along with the demo, the team will submit the **full source code** and a **user setup guide**. Each member will also submit an individual reflection report explaining their technical role.

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.

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 and Milestones

WEEK 1 - 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6
Group Formation Introduction to the project	Individual Literature review on whole project Identifying learning gaps	Deeper, specific Technical review Listing out the features for 5 unique design solutions Searching for well detailed datasets	Narrowing down on the technical research Finalizing the 5 designs Start Wire framing and developing the Figma prototypes	Finalizing and refining the Figma prototypes Presenting it to the client
WEEK 7	**WEEK 8 - 9**	**WEEK 10**	**WEEK 11**	**WEEK 12**
Start the development of the 3 designs based on the client's choices Cleaning, pre-processing and splitting of collected datasets for AI model training	Hybrid AI model training - 3 different and unique design combinations	Three tier Web App Development with database , user interface and back-end logic	Integration of the Web apps with the hybrid AI models Final testing and debugging of the full stack system	Project Delivery and presentation to the client

Figure :1 Ghantt chart laying out the project Timeline

9.3 Team Roles

Name	Figma Prototype	Implementation Group	Main Responsibilities	Individual Learning Focus
Arun (Myself)	Prototype 1	Implementation 1	Team leader, AI Model development, Full stack web app development, AI model integration for malware detection	Hybrid Machine Learning for - Malware classification and behavior analysis in CPS
Vatana	Prototype 2	Implementation 2	Hybrid ML model development and tuning	Phishing - Classification hybrid machine learning algorithms and techniques
Tien	Prototype 3	Implementation 2	Data preprocessing, pipeline integration, model input setup	Data preprocessing for machine learning
Klos	Prototype 4	Implementation 3	UI/UX design, dashboard and visual output	Interface and data visualization dashboard
Nicha	Prototype 5	Implementation 3	Frontend (ReactJS), testing, data handling, basic security setup	Data security framework

Figure 2: Team roles and work split

10. Conclusion

This project is designed to address the gaps in traditional malware detection by developing three web-based systems that use hybrid machine learning models. These systems are proposed to detect both known and unknown threats by combining supervised classifiers with unsupervised anomaly detectors. The focus is not just on building accurate models, but also on integrating them into usable web applications with complete frontend, backend, and database components.

The development process starts with five Figma design prototypes, each linked to a different hybrid AI setup. From these, three implementations are to be developed, supporting real-time predictions, user input handling, detection history, and admin tools. Hybrid model combinations such as Autoencoder with Random Forest, CNN-LSTM, Random Forest with PCA, TF-IDF with Logistic Regression, and DistilBERT with LSTM are to be explored—each suited to specific attack types like phishing, malware, DDoS, and behavioural threats.

To build this end-to-end system, the project involves learning how to prepare datasets, train models, design interfaces, build APIs, and manage stored results. Research studies help define which models to use, how to set thresholds, and how to evaluate performance, so that each solution is aligned with practical needs and real-world threat scenarios.

Overall, the work is planned to show how hybrid AI models can be applied in real-time security tools. The systems are to be fully usable, allowing interaction through web interfaces and providing results with built-in history and admin controls. With a scalable and modular structure, these implementations will also be ready for future improvements such as cloud hosting, smarter learning, and integration with other monitoring systems.

11. Appendix

11.1 Abbreviations

AI: Artificial Intelligence
API: Application Programming Interface
CSV: Comma-Separated Values
CIC: Canadian Institute for Cybersecurity
CPS: Cyber-Physical Systems
CNN: Convolutional Neural Network
DDoS: Distributed Denial-of-Service
Flask: Python-based Web Framework
IoT: Internet of Things
LSTM: Long Short-Term Memory
MVC: Model-View-Controller
MVP: Minimum Viable Product
NLP: Natural Language Processing
PCA: Principal Component Analysis
PE: Portable Executable
RF: Random Forest
RESTful: Representational State Transfer
SQL: Structured Query Language
SVM: Support Vector Machine
TF-IDF: Term Frequency-Inverse Document Frequency
UI: User Interface
URL: Uniform Resource Locator
XGBoost: Extreme Gradient Boosting

11.1 List of Figures

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. Alsharafi, A., Al-Hawari, T., Alzubi, A., & Khan, W. Z. (2023). Real-time malware detection using deep learning and web-based automation. *Machine Learning and Knowledge Extraction*, *5*(3), 607–621. https://doi.org/10.3390/make5030037
4. Canadian Institute for Cybersecurity. (2022). *CIC-MalMem2022 dataset*. University of New Brunswick. https://www.unb.ca/cic/datasets/malmem-2022.html
5. 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

6. Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches* (4th ed.). Sage Publications.
7. 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
8. 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
9. 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
10. 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