Cyber Capstone Experience

Sean Hodgson and Puumaya Tahiru

Capstone Overview

- Senior level course
- Team based
- Real world problems

Projects Offered

University of Idaho
Department of Computer Science

Developing Cybersecurity Tutorials

Client Organization: Department of Computer Science at University of Idaho
Client: Daniel Conte de Leon (dcontedeleon@uidaho.edu)
Mentor: Ananth Jillepalli
Preferred weekly meeting method: Individual updates once every week
Scope: 2 or 3 members

Abstract & motivation:

Cybersecurity is a field where industry jobs are driven by requirements of hands-on skills. However, cybersecurity coursework at universities is often theoretical in nature. This creates a gap between academia training and industry expectations. Our project CERES (Cybersecurity Education RESources) has been tackling the academia-industry training gap since 2015. We create open-source walkthroughs and tutorials to facilitate hands-on training for cybersecurity topics. Prior student teams have created tutorials on topics such as desktop penetration (pen) testing, web app pen testing, network vulnerability scanning, packet capture and analysis, firewalls, and active directory management.

We seek a team that is interested in developing walkthroughs and/or tutorials for cybersecurity of Wireless and Distributed systems. Such work would involve researching open literature and identifying some technologies, scoping problems, and creating tutorials to solve the scoped problems with the identified technologies.

Outcomes of Project:

- Cybersecurity tutorials on topics/tools such as:
  - Mobile app security testing with MobSF (Static), Frida/Burp Suit (Dynamic)
  - Packet capture and analysis of 4G/5G data
  - WPA/2/3 pen testing
  - RAID security using mdadm (Linux), diskmgmt.msc (Windows), diskutil (macOS)
  - Security analyses using PIPE (Petri Net Editor) and PyMC3/Markovify (Markov Chains)
  - Central security using SaltStack or Ansible or Chef or Puppet
- Scholarly academic publication possible upon excellent achievement of project work

Expectation of Group:

- Understanding of Python, testing tools, and different OS environments
- Computers powerful enough to run moderately complex tools

Resources from sponsor & client:

- Prior tutorials for inspiration: https://github.com/ajillepalli/HandsOnCyberTutorials
- Regular mentoring

CROWDSTRIKE

Developing a Penetration Testing Toolkit

Client Organization: CrowdStrike
Client: Dayton Dekam (daytondekam@crowdstrike.com)
Mentor: Ananth Jillepalli
Preferred weekly meeting method: Individual updates once every week
Scope: 1 member

Abstract & motivation:

Penetration testing is an important skill and ability when ensuring that your software and hardware is secure. However, it faces a lot of faults, particularly with tools being standalone, no way to easily store logs of the data, and a clunky command line interface. Although testing like this is usually done by professionals who have experience working with these kinds of tools daily, this helps new cybersecurity professionals break into the scene but also simplifies the workflow of professionals with this single grouped GUI based interface.

Security penetration testers use a variety of tools, which collect data from various tools such as network mapping, password cracking, and others, so it is crucial to have these tools right at your fingertips at a moment's notice. For this project, we will build a tool that aggregates all these tools into a single multi-tool which implements many other components that assist in penetration testing, particularly a GUI making all the prior only command line-based tools to have a graphical interface to read and making understanding not a task of memorization. A built-in terminal for any needs that may arise while using the tool. An encrypted notebook for keeping track of any information. Lastly, incorporating basic automation.

Outcomes of Project:

- Become familiar with the nmap, burp suite, and other common penetration testing tools.
- Become familiar with encryption standards and how to maintain keys from being leaked
- Become familiar with a GUI platform and how to publish these GUIs for easy downloading
- Understand how to setup and configure automation
- Practice developing secure, fault tolerant, and testable code

Expectation of Group:

- Understanding of Python, testing tools, and different OS environments
- Computer capable of hosting a virtual machine

Resources from sponsor & client:

- Industry guidance, consultation

Machine Learning Based Malware Detection with Automated Rationale Documentation for Interpretability: Capstone Project for CptS 432 Fall 2025

Client: Scalable Algorithms for Data Science Laboratory (SCADS)
Primary Contact Person: Tashi Stirewalt, tashi.stirewalt@wsu.edu
Secondary Contact persion: Dr. Assefaw Gebremedhin, assefaw.gebremedhin@wsu.edu
Preferred Meeting Method: Online (can meet in person in the beginning and as needed)
Preferred Meeting Frequency: Weekly for a maximum of 1 hour
Scope: 2-3 students

Motivation:
In recent years, organizations have faced a significant increase in malware attacks. Advancing machine learning (ML) techniques have become essential for improving malware detection efforts. While ML solutions are effective for classification tasks, they primarily depend on statistical analysis methods, which can make their predictions and underlying reasoning difficult to interpret. This lack of clarity makes it challenging to trust the reliability of ML models in critical situations without a thorough understanding of their architecture, implementation, training, and evaluation. This trust can be enhanced by providing precise and accessible explanations that offer insight into the model's rationale.

Problem Statement:
This project aims to address several challenges in ML-based malware detection:

- **"Black-Box" ML Models**: While ML improves accuracy, it often lacks interpretability, making it difficult for analysts to understand alerts.
- **Need for Justifiable Predictions**: Analysts require justifications for flagged files to bolster trust, convey learned patterns, and expedite incident response and resolution.

Project Objective:
This research project focuses on creating an ML-based system for detecting malware in executable binaries while providing explanations for its decisions. Key objectives include:

- **Feature Extraction**: Develop a pipeline to analyze binaries, extracting static features like opcode mnemonic sequences for model training.
- **Model Training**: Train various ML models (e.g., Random Forests, SVMs, NNs) on labeled samples of malware and benign software.
- **Identifying Indicators**: Implement an interpretability mechanism to highlight key features influencing malware predictions.
- **Structured Rationale Preparation**: Organize decision factors, including the model's prediction, confidence score, top features, and relevant metadata, to facilitate clear and concise reasoning.
- **LLM-Powered Explanation Generation**: Utilize an LLM to convert structured data into natural language explanations, automating the documentation process.

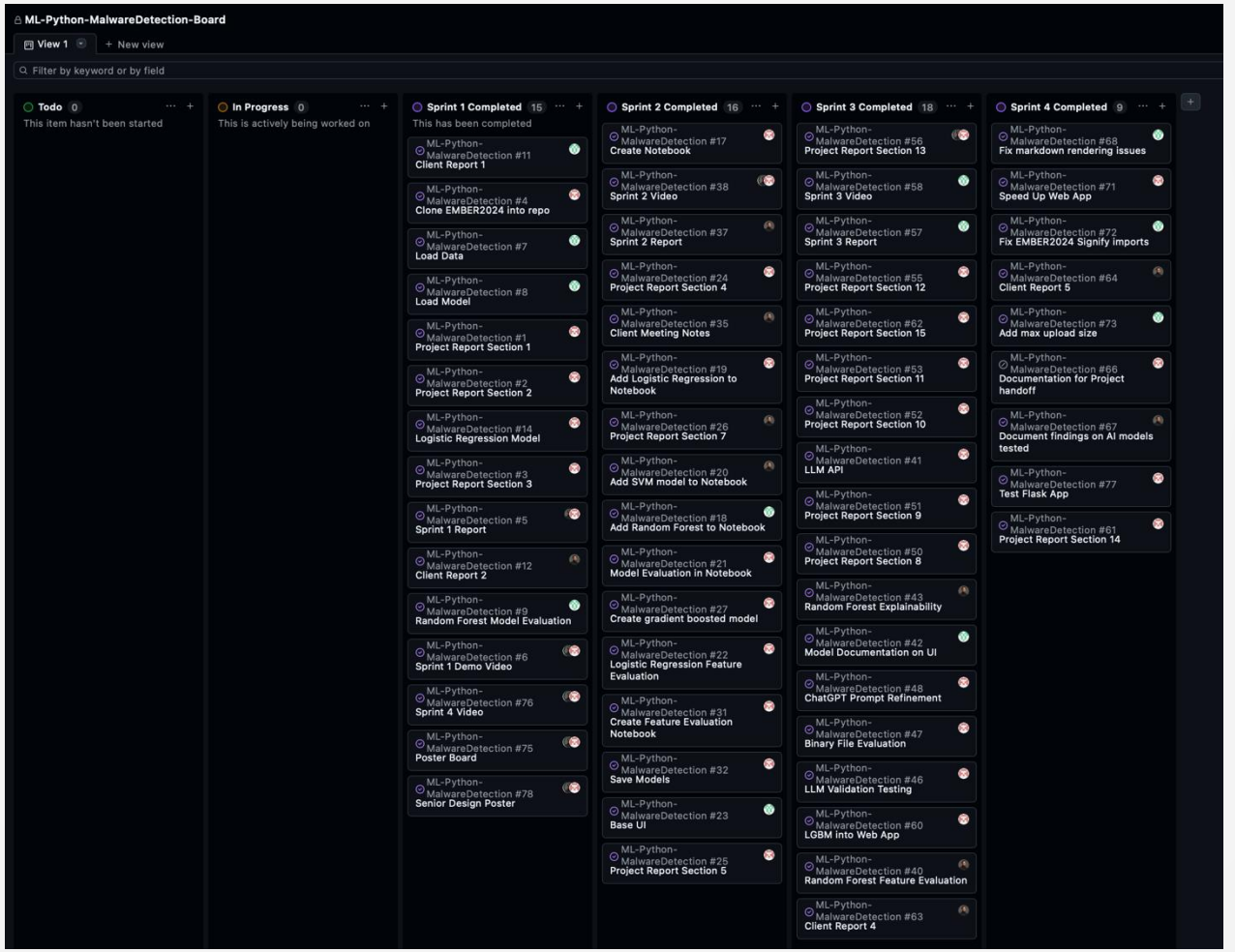
Capstone Process

- Weekly meetings and check-ins
  - Separate coach and client meetings
- Four sprints spread across 16 weeks

Sprints

- Four sprints, one every month
  - Code and associated items
    - GitHub issues, to-do boards, branches, and pull requests
  - Sprint report
  - Client meetings report
  - Sprint demo video

To-Do Board


ML-Python-MalwareDetection-Board
View 1
New view
Filter by keyword or by field
Todo 0
This item hasn't been started
In Progress 0
This is actively being worked on
Sprint 1 Completed 15
This has been completed
ML-Python-MalwareDetection #11 Client Report 1
ML-Python-MalwareDetection #4 Clone EMBER2024 into repo
ML-Python-MalwareDetection #7 Load Data
ML-Python-MalwareDetection #8 Load Model
ML-Python-MalwareDetection #1 Project Report Section 1
ML-Python-MalwareDetection #2 Project Report Section 2
ML-Python-MalwareDetection #14 Logistic Regression Model
ML-Python-MalwareDetection #3 Project Report Section 3
ML-Python-MalwareDetection #5 Sprint 1 Report
ML-Python-MalwareDetection #12 Client Report 2
ML-Python-MalwareDetection #9 Random Forest Model Evaluation
ML-Python-MalwareDetection #6 Sprint 1 Demo Video
ML-Python-MalwareDetection #76 Sprint 4 Video
ML-Python-MalwareDetection #75 Poster Board
ML-Python-MalwareDetection #78 Senior Design Poster
Sprint 2 Completed 16
ML-Python-MalwareDetection #17 Create Notebook
ML-Python-MalwareDetection #38 Sprint 2 Video
ML-Python-MalwareDetection #37 Sprint 2 Report
ML-Python-MalwareDetection #24 Project Report Section 4
ML-Python-MalwareDetection #35 Client Meeting Notes
ML-Python-MalwareDetection #19 Add Logistic Regression to Notebook
ML-Python-MalwareDetection #26 Project Report Section 7
ML-Python-MalwareDetection #20 Add SVM model to Notebook
ML-Python-MalwareDetection #18 Add Random Forest to Notebook
ML-Python-MalwareDetection #21 Model Evaluation in Notebook
ML-Python-MalwareDetection #27 Create gradient boosted model
ML-Python-MalwareDetection #22 Logistic Regression Feature Evaluation
ML-Python-MalwareDetection #31 Create Feature Evaluation Notebook
ML-Python-MalwareDetection #32 Save Models
ML-Python-MalwareDetection #23 Base UI
ML-Python-MalwareDetection #25 Project Report Section 5
Sprint 3 Completed 18
ML-Python-MalwareDetection #56 Project Report Section 13
ML-Python-MalwareDetection #58 Sprint 3 Video
ML-Python-MalwareDetection #57 Sprint 3 Report
ML-Python-MalwareDetection #55 Project Report Section 12
ML-Python-MalwareDetection #62 Project Report Section 15
ML-Python-MalwareDetection #53 Project Report Section 11
ML-Python-MalwareDetection #52 Project Report Section 10
ML-Python-MalwareDetection #41 LLM API
ML-Python-MalwareDetection #51 Project Report Section 9
ML-Python-MalwareDetection #50 Project Report Section 8
ML-Python-MalwareDetection #43 Random Forest Explainability
ML-Python-MalwareDetection #42 Model Documentation on UI
ML-Python-MalwareDetection #48 ChatGPT Prompt Refinement
ML-Python-MalwareDetection #47 Binary File Evaluation
ML-Python-MalwareDetection #46 LLM Validation Testing
ML-Python-MalwareDetection #60 LGBM into Web App
ML-Python-MalwareDetection #40 Random Forest Feature Evaluation
ML-Python-MalwareDetection #63 Client Report 4
Sprint 4 Completed 9
ML-Python-MalwareDetection #68 Fix markdown rendering issues
ML-Python-MalwareDetection #71 Speed Up Web App
ML-Python-MalwareDetection #72 Fix EMBER2024 Signify imports
ML-Python-MalwareDetection #64 Client Report 5
ML-Python-MalwareDetection #73 Add max upload size
ML-Python-MalwareDetection #66 Documentation for Project handoff
ML-Python-MalwareDetection #67 Document findings on AI models tested
ML-Python-MalwareDetection #77 Test Flask App
ML-Python-MalwareDetection #61 Project Report Section 14

Sprint and Client Reports

Sprint 1 Report (8/19/25 - 9/13/25)

What's New (User Facing)

- Imported the EMBER dataset, vectorized it, and trained it using a Random Forest classifier

Work Summary (Developer Facing)

The team had a late start meeting the client so we were not able to get as much work done as we would have liked to have. Sean was able to work on training the model while Andrew worked on the project report with the help of Puumaaya. Puumaaya also worked on the client report. Our team has effective communication with a Discord server to send files and a SMS groupchat for quick communication. Now we understand that we have to work quickly and be proactive in our work and we will carry this into sprint 2.

Unfinished Work

We were able to complete all the issue we had for sprint 1. Sprint 2 will be more challening because that is when we will assign the core functions of the project which will be much harder to do in one sprint.

Completed Issues/User Stories

Here are links to the issues that we completed in this sprint:

- Sprint report 1 Issue: #5
- Sprint report 1 Video: #6
- Random Forest Model Evaluation: #9
- Client Report 1: #11
- Client Report 2: #12
- Report Section 1: #1
- Report Section 2: #2
- Report Section 3: #3
- Clone EMBER Repo: #4
- Load Data: #7
- Load Model: #8
- LR Model: #14
- Assign points: #13

Code Files for Review

Please review the following code files, which were actively developed during this sprint, for quality:

- main.py
- logisticRegression.py
- svm.py

Retrospective Summary

Here's what went well:

- Setting up repo
- Working on getting some models trained

Here's what we'd like to improve:

- The amount of trained models

Here are changes we plan to implement in the next sprint:

- More proactivity
- Start the UI interface
- Train more models

Client Meetings Report for September 2nd

Agenda - September 2nd

- Introduction
- What will this project look like?
- Overview of machine learning topics.

Minutes - September 2nd

- Client and Research team introductions.
- Background behind why the client wants this product.
- Quick overview on machine learning.
- Overview of the EMBER2024 dataset.
- Our team asked questions about machine learning and the project itself.

Retrospective Summary - September 2nd

Here's what went well:

- Our client is very clear on the direction of the project.
- We have a good understanding of what we need to work on.

Here's what we'd like to improve:

- We didn't have much to show in our repository, though this wasn't very avoidable. We could have played with some machine learning models beforehand.

Here are changes we plan to implement as soon as possible:

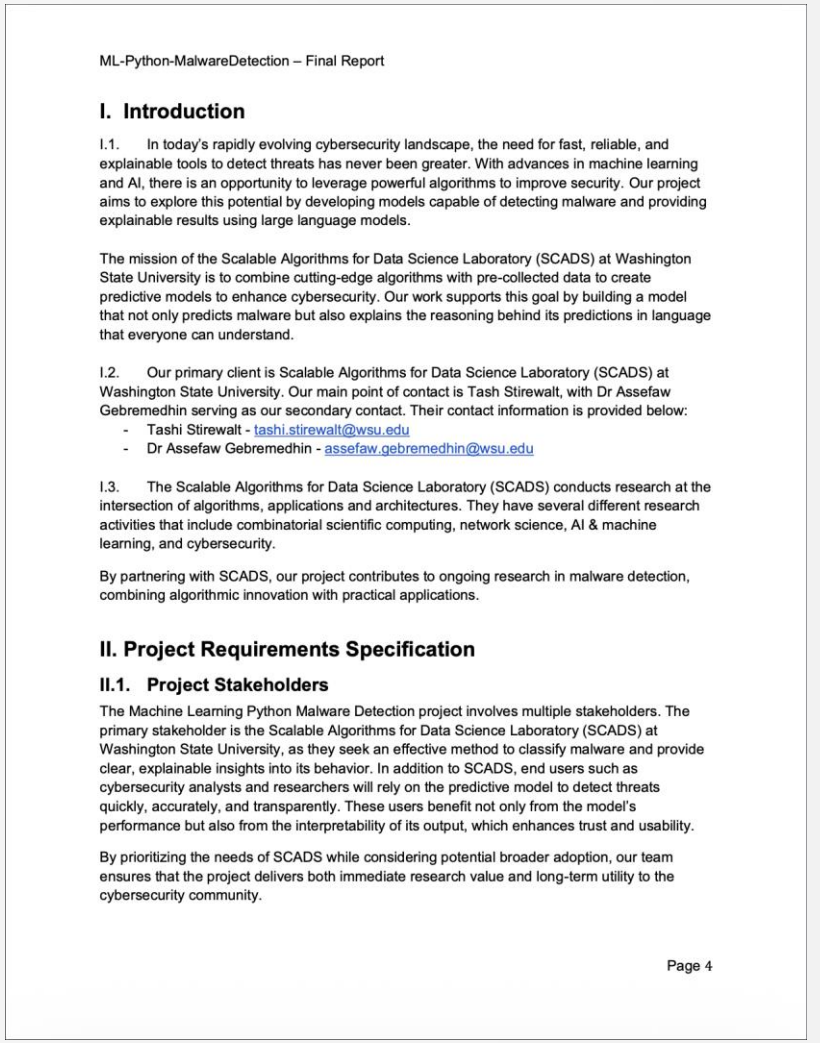
- Take a look at the dataset and understand what it is.
- Getting the dataset into our repo.
- Loading the dataset into a basic model to generate a classification report with atleast one model.

Final Deliverables

- Poster
- Final Report
- Final client meeting
- EECS Poster Showcase

Final Report

- 29 Pages



ML-Python-MalwareDetection – Final Report

I. Introduction

I.1. In today's rapidly evolving cybersecurity landscape, the need for fast, reliable, and explainable tools to detect threats has never been greater. With advances in machine learning and AI, there is an opportunity to leverage powerful algorithms to improve security. Our project aims to explore this potential by developing models capable of detecting malware and providing explainable results using large language models.

The mission of the Scalable Algorithms for Data Science Laboratory (SCADS) at Washington State University is to combine cutting-edge algorithms with pre-collected data to create predictive models to enhance cybersecurity. Our work supports this goal by building a model that not only predicts malware but also explains the reasoning behind its predictions in language that everyone can understand.

I.2. Our primary client is Scalable Algorithms for Data Science Laboratory (SCADS) at Washington State University. Our main point of contact is Tash Stirewalt, with Dr Assefaw Gebremedhin serving as our secondary contact. Their contact information is provided below:

- Tashi Stirewalt - tashi.stirewalt@wsu.edu
- Dr Assefaw Gebremedhin - assefaw.gebremedhin@wsu.edu

I.3. The Scalable Algorithms for Data Science Laboratory (SCADS) conducts research at the intersection of algorithms, applications and architectures. They have several different research activities that include combinatorial scientific computing, network science, AI & machine learning, and cybersecurity.

By partnering with SCADS, our project contributes to ongoing research in malware detection, combining algorithmic innovation with practical applications.

II. Project Requirements Specification

II.1. Project Stakeholders

The Machine Learning Python Malware Detection project involves multiple stakeholders. The primary stakeholder is the Scalable Algorithms for Data Science Laboratory (SCADS) at Washington State University, as they seek an effective method to classify malware and provide clear, explainable insights into its behavior. In addition to SCADS, end users such as cybersecurity analysts and researchers will rely on the predictive model to detect threats quickly, accurately, and transparently. These users benefit not only from the model's performance but also from the interpretability of its output, which enhances trust and usability.

By prioritizing the needs of SCADS while considering potential broader adoption, our team ensures that the project delivers both immediate research value and long-term utility to the cybersecurity community.

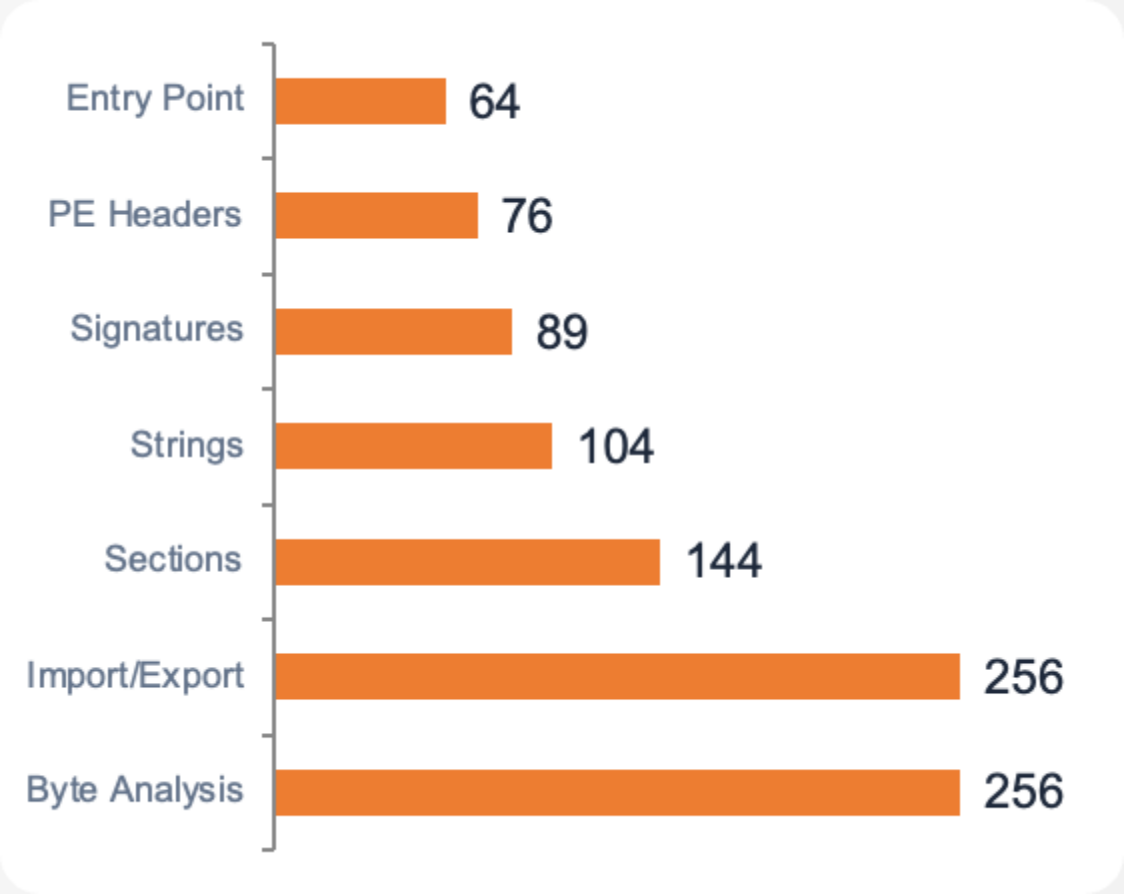
Page 4

ML Malware Analysis

- Millions of new malware emerge each year
- Goal: To build a malware detection platform that is accurate, explainable, and trustable

Ember 2024 Dataset

- 3.2M malicious & benign files
- 6 file types (Win32, Win64, .NET, APK, PDF, ELF)
- Data from Sep 2023 - Dec 2024
- VirusTotal-sourced submissions
- 6,315 evasive challenge samples



Pipeline

PE file upload

Feature Extraction

LightGBMClassification

SHAP + Lime Explainability

LLM Report

Explainable AI

Shapely Additive Explanations(SHAP) – Which features matter the most and why

Local Interpretable Model-agnostic Explanations(LIME) - If we changed small things about the file, what happens

LLM Report

- Classification Verdict
- Key features
- Risk Content

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AI-Powered Malware Detection and Documentation

Sponsor: Scalable Algorithms for Data Science Laboratory (SCADS)
Mentors: Tashi Stirewalt and Dr. Assefaw Gebremedhin
Sean Hodgson, Andrew Edson, Puumaaya Tahiru

SCADS
Scalable Algorithms for Data Science Lab

Abstract

Modern malware evolves rapidly, demanding automated systems capable of detection, interpretation, and documentation. Our project is an end-to-end AI-powered malware analysis pipeline that integrates machine learning models, explainability frameworks, and a user-friendly web interface. Our system enables real-time classification of Windows Portable Executable (PE) files and generates transparent explanations for each prediction, supporting analysts with both accuracy and interpretability.

Introduction

Cyber threats continue to grow in scale and sophistication, making traditional signature-based detection insufficient. This project aims to bridge the gap between high-performance malware detection and interpretable decision-making. By leveraging the EMBER 2024 feature extraction framework and ML models, we developed a system capable of analyzing unknown binaries, making predictions, and documenting model reasoning.

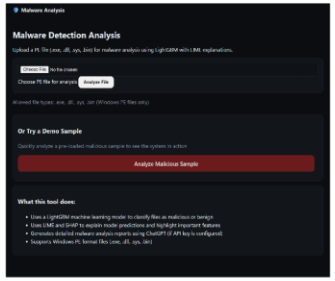
Project Approach

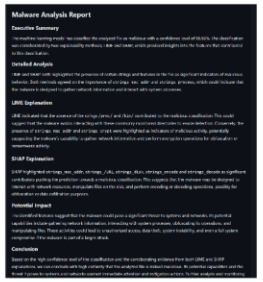
This project was primarily research-oriented rather than a traditional web development effort. During the first two sprints, we focused heavily on training and evaluating a variety of machine learning models and explainability tools. Much of this work lives in our codebase but does not directly appear in the final web application. The models we trained and evaluated included Random Forest, LightGBM, SVM, and Logistic Regression, along with an additional Random Forest model for malware family analysis. We also developed multiple notebooks to experiment with different explainability packages.

LightGBM

Implementation

Our final web application accepts PE binaries (.exe, .dll, .sys, .bin) and processes them using the EMBER2024 feature extractor to produce a vectorized representation of the file. This vectorized data is then passed into our trained LightGBM model for classification. Using the model's output, we generate explanations through both SHAP and LIME. The results from these explainers are then provided to the ChatGPT API, which produces a detailed malware analysis summary for the user.





Top Model Performance

- LightGBM achieved strong performance with high detection accuracy and efficient inference times.
- Random Forest performed consistently well, but its training time on the PE dataset was prohibitively long. However, it delivered strong, reliable results on the PDF dataset.

LightGBM

Classification Report:

	precision	recall	f1-score	support
0	0.9272	0.9890	0.9571	12000
1	0.9882	0.9223	0.9541	12000
accuracy			0.9557	24000
macro avg	0.9577	0.9557	0.9556	24000
weighted avg	0.9577	0.9557	0.9556	24000

Random Forest

Classification Report:

	precision	recall	f1-score	support
0	0.9091	0.9948	0.9500	12000
1	0.9943	0.9005	0.9451	12000
accuracy			0.9477	24000
macro avg	0.9517	0.9477	0.9475	24000
weighted avg	0.9517	0.9477	0.9475	24000

Conclusion

Over the course of this project, our team gained substantial experience working with machine learning models. A major focus of our work was bridging the gap between the inherent opacity of ML models and the need for clear, trustworthy explanations. To achieve this, we combined traditional explainability tools like SHAP and LIME with generative AI, while taking deliberate steps to reduce hallucinations and ensure the analyses remained grounded in the underlying model outputs. This project also gave us practical experience in model training, feature extraction, experimentation, and integrating research-oriented workflows into a functional web application. The work emphasized not only building accurate models but also making their decisions interpretable and useful for real-world malware analysis.

Glossary

- PE File: Portable Executable format used for Windows binaries.
- EMBER2024: A malware dataset and feature extraction framework for ML research.
- SHAP: A game theoretic approach for interpreting machine learning models.
- LIME: A local explanation tool that approximates model behavior near a specific instance.
- LightGBM: A gradient boosting framework optimized for high performance.

Acknowledgements

We thank the Scalable Algorithms for Data Science Laboratory (SCADS)

Tashi Stirewalt, Dr. Assefaw Gebremedhin, and Ananth Jillepalli.

Coug Coders