Final Year Project Report

Author Attribution of Binaries with Machine Learning

Luka van Rooyen

submitted in part fulfilment of the degree of

BSc Computer Science (Information Security)

Supervisor: Dr. Rachel Player

Department of Information Security
Royal Holloway, University of London

October 2025

Contents

Chapter 1: **Preliminary Project Plan**

1.1 Abstract

The problem of attributing a piece of code, particularly a binary file, to a known author using machine learning is complex and must be decomposed into several logical steps[1]. Moreover, the issue has applications in both malware forensics[2] and threat detection, as it allows us to automatically identify and categorise malicious code authors[3]. This project explores predicting authorship by extracting and analysing features from compiled code and training machine learning models to interpret these features, assessing whether extracted features correspond to known malicious code or authors. The early objectives include: reviewing existing techniques for binary feature extraction, building a dataset of binaries from multiple authors, implementing and testing preliminary machine learning classifiers on extracted features, and evaluating early test results and refining the approach accordingly. The ultimate goal of the project is to evaluate whether distinctive patterns and features in compiled binaries can be analysed for reliable authorship attribution via machine learning methods. In doing so, we can determine whether the features of the binary are indicative of malicious code or known malicious authors.

1.2 Timeline

Weeks 1-2 (*September 29th - October 10th*)

- Review recommended literature, particularly pertaining to binary feature abstraction, as this will enforce early prototypes of feature extraction tools
- Explore supplementary readings using Google scholar to find academic literature on binary feature extraction and control graphs[4]

Deliverables:
- Preliminary Project Plan
- Initial coding repository ready for coding experiments

Weeks 3-5 (*October 13th - October 24th*)

- Begin drafting report on binary feature extraction
- Set up coding environments (*i.e.*- importing and installing necessary libraries/tools) based on research
- Begin prototyping different methods of binary feature extraction
  - Implement small test scripts on individual compiled binaries

Deliverables:
- Minimal scripts/algorithms for extracting features from binary files
- Rough draft of binary feature extraction report

Weeks 6-8 (*October 27th - November 7th*)

- Refine development of binary feature extraction tool
- Continue binary feature extraction report
- Familiarise myself with machine learning techniques in Python (particularly using the `scikit-learn` library[5]) and build a plan for my machine learning algorithm

Deliverables:
- A successful binary feature extraction tool
- A completed binary feature extraction report
- A plan for the machine learning aspect of the project

Weeks 9-11 (*November 10th - November 21st*)

- Locate open source datasets that can be used in training (such as from the Google Code Jam[6])
- Begin producing prototype machine learning algorithms, built to specifically analyse the features extracted by my completed extraction tool
- Assess results of preliminary machine learning experiments, in order to refine the algorithm
- Document key insights to include in my interim report and gauge what additional work needs to be done

Deliverables:
- Preliminary machine learning algorithms/experiments
- Evaluation of model performance and challenges

Weeks 12-14: (*November 24th - December 5th*)

- Refine model further using evaluation from previous week
- Begin training and using external datasets to determine efficacy on untrained datasets
- Develop interim report further, including binary feature extraction and evaluations from previous weeks

Deliverables:
- A more refined machine learning algorithm that can measurably produce more accurate results
- An updated interim report that reflects this development

Week 15 (*final week*)

- Conduct any further necessary code revisions, whether it be for the binary extraction tool or the machine learning algorithm
- Finalise interim report

1.3 Risk Assessment & Mitigations

Risk Category	**Risk Description**	**Likelihood (1-5)**	**Impact (1-5)**	**Risk-Level (1-25)**	**Mitigation Strategies**
Technology	Hardware failure/ Data loss	2	4	8	Use Version Control Systems, such as Gitlab, to ensure any files are backed up externally, as well as committing regularly. Constantly save work after editing locally.
	Computational resource limitations	3	3	9	Design code with efficiency in mind, making sure to not unnecessarily drain resources. Use my main PC with better hardware rather than laptop when performing large tasks.
Security	Including malicious binaries in datasets	2	5	10	Include only non-malicious binaries in dataset initially as proof of concept, then move to malicious binaries with very **limited** access to any external programs. Consult with supervisor on how to handle these malicious binaries.
Technical	Scarcity of viable Datasets	4	4	16	There are not many varied datasets that are easy to acquire, so I will have to manually search for some external binary files with their authors and build on top of minimal datasets.

Risk Category	**Risk Description**	**Likelihood (1-5)**	**Impact (1-5)**	**Risk-Level (1-25)**	**Mitigation Strategies**
Personal	Machine Learning and Binary Feature Extraction Overhead	5	3	15	Due to my limited experience with Machine Learning *and* Binary Feature Extraction, I will have to allocate time to learn these technologies in themselves, as well as best practices/ optimisations that can be done.
	Poor planning/task estimation	3	5	15	Due to my inexperience, I will need to carefully consider how I break down the project into sections. I will consistently meet with my supervisor in order to evaluate my task estimation.
	Imbalance between coding and report-writing	3	3	9	Continually update my report alongside coding, so that there is not a deficit between the two components.

Bibliography

[1] N. Rosenblum, X. Zhu, and B. P. Miller, “Who wrote this code? identifying the authors of program binaries,” in *European Symposium on Research in Computer Security*, 2011, pp. 172–189.

[2] S. Alrabaee, N. Saleem, S. Preda, L. Wang, and M. Debbabi, “OBA2: An Onion approach to Binary code Authorship Attribution,” *Digital Investigation*, vol. 11, pp. S94–S103, 2014, doi: https://doi.org/10.1016/j.diin.2014.03.012.

[3] V. Kalgutkar, R. Kaur, H. Gonzalez, N. Stakhanova, and A. Matyukhina, “Code Authorship Attribution: Methods and Challenges,” *ACM Comput. Surv.*, vol. 52, no. 1, Feb. 2019, doi: 10.1145/3292577.

[4] H. Theiling, “Extracting safe and precise control flow from binaries,” in *Proceedings seventh international conference on real-time computing systems and applications*, 2000, pp. 23–30.

[5] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[6] A. Caliskan-Islam *et al.*, “De-anonymizing programmers via code stylometry,” in *24th USENIX security symposium (USENIX Security 15)*, 2015, pp. 255–270.