National College of Ireland

Configuration Manual

MSc Cybersecurity

EMPLOYING SVM AND RANDOM FOREST FOR ENHANCED DETECTION OF LINUX-BASED MALWARE

MSCCYB_SEP2023

Vedant vaidya

Student ID: x22194304

School of Computing

National College of Ireland

Supervisor: Niall Heffeman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vedant Vaidya
Student ID:	X22194304
Programme:	MSc cybersecurity **Year:** 2023/24
Module:	Practicum
Lecturer:	Niall Heffeman
Submission Due Date:	12/08/2024
Project Title:	EMPLOYING SVM AND RANDOM FOREST FOR ENHANCED DETECTION OF LINUX-BASED MALWARE
Word Count:	1098 **Page Count:** 10

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 bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other

author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: v.vaidya

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

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

Configuration Manual

Vedant Vaidya

Student ID: x22194304

1. Introduction

It is a highly secure machine learning-based system [1] developed by Yahoo to detect potentially harmful executable files, the Malware Detection System. Our system uses state of the art algorithms such as Support Vector Machines (SVM) [2], Random Forest [3] to classify vector files into two classes: malware and benign. Below is an install guide for setting up the Malware Detection System.

1.1 Purpose

This system is designed to improve security by automatically analyzing suspicious applications and identifying potential malware. The system utilizes machine learning to extract and analyze executable files [4] rapidly, automatically making an early risk assessment ensuring that security experts spend their time appropriately on high-risk cases.

1.2 Scope

The entirety of the Malware Detection System setup and usage is covered in this manual which includes;

- System requirements and installation
- Data preparation and preprocessing
- Model training and evaluation
- New file analysis
- Customization and troubleshooting

2. System Overview

The following key components make up the Malware Detection System [5]:

2.1 Data Preprocessing Module

This module takes care about the first step of preprocessing on malware dataset.

- Loading the CSV file

- Handling missing values
- Encoding the target variable
- Feature scaling

2.2 Machine Learning Models [6]

Three Core Models Used by the System

- Support Vector Machine (SVM)
- Random Forest
- Introduction to SVM (Using GridSearchCV as Optimizer)

2.3 Feature Extraction Engine

A new file analysis — the feature extraction function for processing PE files and extracting features.

2.4 Evaluation Tools

There are evaluation metrics [7], a system to view predictions / targets side-by-side and other helpful tools you can used on Kaggle.

3. System Requirements

3.1 Hardware Requirements

- Processor: Multi-core processor (4+ cores recommended)
- RAM: 8GB minimum, 16GB or more recommended
- Storage: 50GB free space for dataset and model storage

3.2 Software Requirements

- Operating System: Windows 10, macOS 10.14+, or Linux (Ubuntu 18.04+ recommended)
- Python: Version 3.7 or higher
- Required Python libraries:
    - pandas
    - numpy
    - scikit-learn
    - matplotlib
    - seaborn
    - pefile

3.3 Additional Requirements

- Google Account (for Google Colab usage)
- Internet connection for downloading dependencies and accessing Google Colab

4. Data Preparation

4.1 Dataset Requirements

The system expects a CSV file with the following structure:

- 'hash': Unique identifier for each file (string)
- 'classification': Target variable ('malware' or 'benign')
- 33 numerical feature columns

```
hash,feature1,feature2,...,feature33,classification
abc123,0.5,1.2,...,0.8,malware
def456,0.3,0.9,...,1.1,benign
```

4.2 Data Loading

1. If using Google Colab, upload the 'Malware.csv' file to your Google Drive
2. Mount Google Drive in Colab:
3. Load the dataset:

```
df = pd.read_csv('/content/drive/MyDrive/dataset/Malware.csv')
print(df.shape)
df.head()
```

4.3 Data Preprocessing

The system automatically handles the following preprocessing steps:

1. Removing the 'hash' column:
2. Encoding the 'classification' column:
3. Handling missing values:
4. Feature scaling:
5. Splitting the data into training and test sets:

```
features = df.drop(['hash', 'classification'], axis=1)
target = df['classification']
target = (target == 'malware').astype(int)
features = features.fillna(features.mean())
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

```
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target, test_size=0.2, random_state=42)
```

5. Model Training

5.1 Support Vector Machine (SVM)

```
svm_model = SVC(kernel='rbf', random_state=42, probability=True)
svm_model.fit(X_train, y_train)

# Predictions
svm_predictions = svm_model.predict(X_test)
```

5.2 Random Forest

```
# Cell 7: Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
rf_predictions = rf_model.predict(X_test)
```

5.3 Optimized SVM

```
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001]}
grid = GridSearchCV(SVC(probability=True), param_grid, refit=True, verbose=2)
grid.fit(X_train, y_train)
```

6. Model Evaluation

6.1 Performance Metrics

For each model, run the following code to get performance metrics:

```
for name, predictions in models.items():
    print(f"\n{name} Model:")
    print("Accuracy:", accuracy_score(y_test, predictions))
    print("Classification Report:")
    print(classification_report(y_test, predictions))
```

6.2 Confusion Matrices

To visualize confusion matrices:

```
# Cell 10: Plotting Confusion Matrices
fig, axes = plt.subplots(1, 3, figsize=(20, 5))

for i, (name, predictions) in enumerate(models.items()):
    cm = confusion_matrix(y_test, predictions)
    sns.heatmap(cm, annot=True, fmt='d', ax=axes[i])
    axes[i].set_title(f'{name} Confusion Matrix')
    axes[i].set_xlabel('Predicted')
    axes[i].set_ylabel('Actual')

plt.tight_layout()
plt.show()
```

6.3 ROC Curves

```
plt.figure(figsize=(10, 8))
for name, model in [('SVM', svm_model), ('Random Forest', rf_model), ('Optimized SVM', grid.best_estimator_)]:
    y_pred_proba = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

7. New File Analysis

7.1 Feature Extraction Function

```
def extract_features(file_path):
    try:
        if not os.path.exists(file_path):
            print(f"File does not exist: {file_path}")
            return None

        pe = pefile.PE(file_path)

        # Extract the 8 features we can get from the PE file
        filesize = os.path.getsize(file_path)
        num_sections = len(pe.sections)
        num_imports = sum(len(entry.imports) for entry in pe.DIRECTORY_ENTRY_IMPORT) if hasattr(pe, 'DIRECTORY_ENTRY_IMPORT') else 0
        num_exports = len(pe.DIRECTORY_ENTRY_EXPORT.symbols) if hasattr(pe, 'DIRECTORY_ENTRY_EXPORT') else 0

        def entropy(data):
            return -sum(p * math.log(p, 2) for p in (float(data.count(byte)) / len(data) for byte in set(data)) if p > 0)

        with open(file_path, 'rb') as f:
            total_entropy = entropy(f.read())

        has_debug = 1 if hasattr(pe, 'DIRECTORY_ENTRY_DEBUG') else 0
        has_tls = 1 if hasattr(pe, 'DIRECTORY_ENTRY_TLS') else 0
        has_resources = 1 if hasattr(pe, 'DIRECTORY_ENTRY_RESOURCE') else 0

        # Create a list of 33 features, filling the rest with 0
        features = [filesize, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, num_sections, num_imports, num_exports, total_entropy,
                    has_debug, has_tls, has_resources, 0, 0, 0, 0, 0]

        print("Features extracted successfully")
        return np.array(features)
```

7.2 File Upload and Analysis

1. Upload a file using Colab's file upload widget:
2. Extract features:
3. Make predictions:

```
uploaded = files.upload()
file_name = list(uploaded.keys())[0]

print(f"Uploaded file: {file_name}")
```

```
if new_features is not None:
    new_features = new_features.reshape(1, -1)


    svm_prediction = svm_model.predict(new_features)
    rf_prediction = rf_model.predict(new_features)
    optimized_svm_prediction = grid.predict(new_features)

    print("SVM Prediction:", "Malware" if svm_prediction[0] == 0 else "Benign")
    print("Random Forest Prediction:", "Malware" if rf_prediction[0] == 0 else "Benign")
    print("Optimized SVM Prediction:", "Malware" if optimized_svm_prediction[0] == 0 else "Benign")
else:
    print("Could not extract features from the uploaded file.")
```

8. Conclusion

Malware Detection System Configuration Manual provides detailed instructions for installing and configuring the Malware Node. Step by step solution: Now hopefully you can

Know why the system was built in first place, what it should and shouldn't do and its key parts

Configure the environment (possibly hardware and software, but it seems problematic)

Form Data and Preprocess it to be used for analysis

Conduct parameter grid search for SVM and Random Forest models

Check the performance of your model with a number of metrics and plots

Analyze Files (New) for Malware Spin up an encyclopedic analysis of new files

Be sure to update and maintain your signature database regularly for the system to be relevant against new types of malware. Learn more machine learning and cybersecurity to help you stay ahead of he curve with Malware Detection System

References

[1] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011. [Online]. Available: https://scikit-learn.org/stable/

[2] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995. [Online]. Available: https://link.springer.com/article/10.1007/BF00994018

[3] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[4] Microsoft, "PE Format," *Windows Dev Center*, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

[5] R. Sihwail, K. Omar, and K. A. Zainol Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *International Journal of Advanced Science and Technology*, vol. 7, no. 7, pp. 1-10, 2018.

[6] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2016.

[7] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011.