AMRITA VISHWA VIDYAPEETHAM | Online

21CSA697A – Minor Project

DATE :

INTRODUCTION

FORNEW EMPLOYEES

- Project Title : Malware Classification using Api Calls
- Student Name:Gokul Pradeep
- Register Number:AA.SC.P2MCA24070134
- Program and Specialization: MCA Regular
- Semester : 3rd SEM
- Department :Computer Appliction
- Institutuion:Amrita Vishwa Vidhyapeetham

Introduction

- Malware is a growing cybersecurity threat that traditional signature-based antivirus systems fail to detect effectively
- Behavioral analysis using API call sequences provides a more reliable detection approach by analyzing program runtime activities instead of static code.
- Deep learning-based sequence modeling enables automatic learning of complex temporal patterns in malware behavior.
- This project proposes a hybrid TCN-BiGRU architecture to accurately classify malware

Domain overview

- Domain: Behavior -Based Malware Detection
- Models like GRU, and TCN are used to capture temporal dependencies in sequential data such as API call logs
- Combining Temporal Convolutional Networks (TCN) and Bidirectional GRU enhances feature extraction and contextual learning for better malware classification.
- Combining Temporal Convolutional Networks (TCN) and Bidirectional GRU enhances feature extraction and contextual learning for better malware classification.

Problem Statement

- The rapid evolution of malware techniques, including polymorphism and obfuscation, makes traditional signature-based detection methods ineffective against zero-day and advanced persistent threats.
- Static analysis methods fail to accurately detect malware that modifies its code structure while maintaining malicious behavior.
- here is a need for a robust behavior-based detection system that can analyze runtime API call sequences to identify malicious patterns.
- Existing machine learning approaches often struggle to effectively capture long-term temporal dependencies in sequential API data.

Objectives

- To overcome the limitations of signature-based and static analysis methods by implementing a behavior-based malware detection approach using runtime API call sequences.
- capture complex temporal patterns in sequential API data by designing a hybrid deep learning architecture combining TCN and BiGRU.
- To effectively model both short-term and long-term dependencies in malware behavior for improved detection of zero-day and obfuscated attacks.
- To validate the effectiveness of the proposed system through performance evaluation using metrics such as Accuracy, Precision, Recall, and F1-Score.

Scope of the Project

Project Boundaries

- The model performs binary classification only (Malware vs Benign), not multi-class malware family classification.
- The system relies solely on API call sequence data and does not use additional features such as network packets or file hashes.
- The dataset is pre-collected and labeled; real-time API capture is outside the current implementation.

Constraints and Assumptions

Constraints

- Model performance depends heavily on the quality and size of the dataset.
- Training deep learning models requires computational resources (GPU preferred)

Assumptions

- API call sequences accurately represent program behavior.
- Malware exhibits distinguishable API patterns compared to benign software.

Applicability Domain

- Endpoint security systems (Antivirus software).
- Endpoint security systems (Antivirus software).
- Security Operations Centers (SOC) for automated threat detection.
- Malware analysis research.
- Intrusion detection systems.

Literature Review / Existing System

1. "Malware Detection by Analyzing API Calls using Machine Learning"

   Key Points:

   - Proposes malware detection using API call behavior rather than static signatures.
   - Extracts API sequences from executable samples.
   - Uses traditional ML algorithms (e.g., Random Forest, SVM) on feature vectors derived from API patterns.
   - Stronger classification signals than static features.

2. “Deep Learning for Malware Detection: A Survey”

Key Points:

- Comprehensive survey covering deep learning approaches in malware detection.
- Discusses CNN, RNN, LSTM, and GRU based models for static and dynamic malware features.
- Highlights sequence modeling benefits for behavior-based detection.

3. “A Comparative Study of Sequence Modeling Approaches for Malware Detection”

Key Points:

- Compares LSTM, GRU, and Transformer models for sequential API call data.
- Measures performance on labeled malware behavior datasets.
- Highlights pros/cons of different sequence models.

Comparative Table – Existing Systems vs Proposed System

Approach	Model Used	Dataset	Accuracy	Limitations
Signature-Based Detection	Pattern Matching	Static Executables	Low-Moderate	Fails for zero-day & obfuscated malware
Machine Learning (Traditional)	SVM / Random Forest	API Call Features	Moderate-High	Limited temporal modeling
Deep Learning (Single Model)	LSTM / CNN	API Sequences	High	May miss long-range dependencies
Proposed System	Hybrid TCN + BiGRU	API Call Sequences	Very High (~98-99%)	Requires large dataset & tuning

Proposed System

Research Methodology / Proposed Approach

- *Data Collection*
- *Data Preprocessing*
- *Model Design*
- *Embedding layer*
- *Model Training*
- *Performance Evaluation*

Overall Workflow

API Call Dataset
Data Preprocessing
Sequence Encoding & Padding
Embedding Layer
TCN Layer
BiGRU Layer
Dense Layer
Malware / Benign
PredictionPerformance Evaluation

Justification of Approach

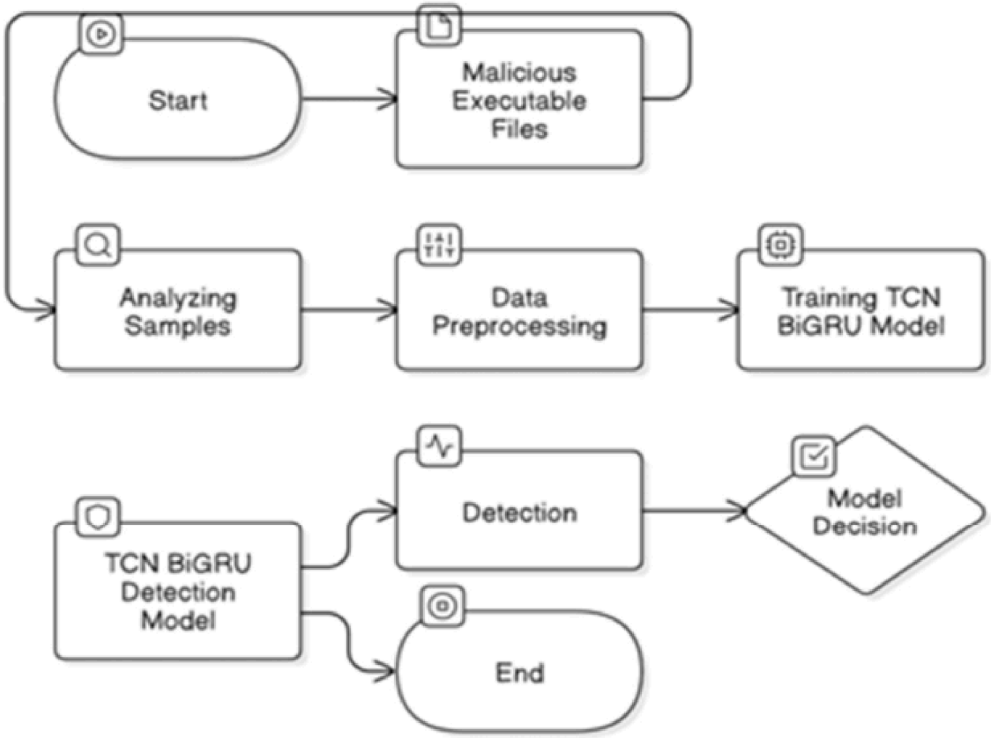
- Behavior-based detection is more robust than static signature-based detection against zero-day attacks.
- API call sequences contain temporal patterns that require sequence modeling techniques.
- TCN provides:Parallel computation,Stable gradients,Long-range dependency capture
- BiGRU provides:Bidirectional context understanding,Efficient learning compared to LSTM

Novelty / Innovation

- Integration of TCN and BiGRU in a unified hybrid architecture for malware behavioral classification.
- Improved detection performance over traditional ML and single deep learning models.
- Focus on dynamic API call behavior rather than static file characteristics.

Architecture

- Architecture Diagram



Module Interaction Explanation

- Data Preprocessing Module-Converts API call text into numerical form.
- Embedding Layer-Converts integers into dense vector representations.
- TCN Module-Applies dilated causal convolutions,Extracts temporal features from API sequence.
- BiGRU Module-Processes sequence forward and backward,Learns contextual relationships.
- Dense + Output Layer-Applies fully connected layer,Uses Sigmoid activation

Data Flow

- API call logs are collected.
- Dataset is cleaned and converted into tokenized sequences.
- Sequences are padded to fixed length.
- Embedding layer transforms tokens into vectors.
- TCN extracts temporal behavior patterns.
- BiGRU refines patterns using bidirectional context.
- Dense layer computes classification score.
- Final output predicts malware or benign.

Methodology / Algorithms

Step-by-step workflow

- Data Collection-Obtain labeled API call sequence dataset.

Each sample is labeled as Malware or Benign.

- Data Preprocessing-Remove noise or irrelevant entries,Convert API calls into tokens (numerical encoding),Apply sequence padding to maintain fixed length, Split dataset into training and testing sets.
- Feature Representation-Use an Embedding Layer to convert integer tokens into dense vectors.This helps capture semantic similarity between API calls.
- Step 4: Temporal Feature Extraction (TCN),Apply dilated causal convolutions,Capture long-term dependencies in API sequences,Extract high-level temporal features.

- Contextual Learning (BiGRU)-Process sequence in forward and backward direction,Learn contextual relationships between API calls,Improve understanding of malware behavior pattern
- Classification-Fully connected (Dense) layer,Sigmoid activation for binary output,Output probability of malware.
- Model Evaluation-Evaluate using :

1. Accuracy
2. Precision
3. Recall
4. F1-Score
5. Confusion Matrix

Algorithm/Model Used

- Temporal Convolutional Network (TCN)
- Bidirectional GRU (BiGRU)
- Dense + Sigmoid Layer
- Optimization Algorithm

Justification for Selection

- Tcn Avoids vanishing gradient problem,Parallel computation (faster training),Captures long-term dependencies using dilation.
- GRU uses Fewer parameters than LSTM,Faster training,Comparable performance,Lower risk of overfitting.

Implementation Details

Algorithms / ML / DL models

- Temporal Convolutional Network (TCN)
- Bidirectional GRU (BiGRU)
- Binary Classification Layer
- Optimization Algorithm

Mathematical Formulation

- **TCN Layer**: The dilated convolution at layer l is given by:
  $y(t) = \sum_{i=0}^{k-1} w_i \cdot x(t - d_i)$
  where w_i represents convolutional filters, k is the kernel size, and d_i is the dilation factor.
- **BiGRU Layer**: The forward and backward GRU equations are defined as:
  $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$ $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$ $h_t = (1 - z_t)h_{t-1} + z_t \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$
  The BiGRU combines hidden states from both directions to improve feature extraction.

Dataset Description

Source of data

The dataset consists of dynamic API call sequences collected from executable files.

It contains labeled samples categorized as:

Malware

Benign

Preprocessing Steps

- Data Cleaning
- Tokenization
- Sequence Padding
- Label Encoding
- Normalization

Train-Test Split / Validation Strategy

Train-Test Split

Dataset split into:

80% Training

20% Testing

(or 70-30 depending on your implementation)

Implementation Details

Tools, frameworks, libraries

- Programming Languages :python
- Framework: TensorFlow ,Keras Api
- IDES : Jupiter Notebook and VS Code
- MAchine Learning Utilities: Scikit-Learn
- version control : Github.

Data Collection Module

- Load API call sequence dataset from CSV file.
- Extract features (API sequences) and labels

Data Preprocessing Module

- Convert raw API call data into model-ready format
- Data cleaning (remove null values).

Embedding Module

- Convert integer-encoded API tokens into dense vector representations.

TCN Module (Temporal Feature Extraction)

- Extract temporal patterns using dilated convolution.

BiGRU Module (Context Learning)

- Learn contextual relationships in both forward and backward direction
- Improve contextual understanding.

Classification Module

- Perform binary classification.
- Fully Connected (Dense) layer

Training Module

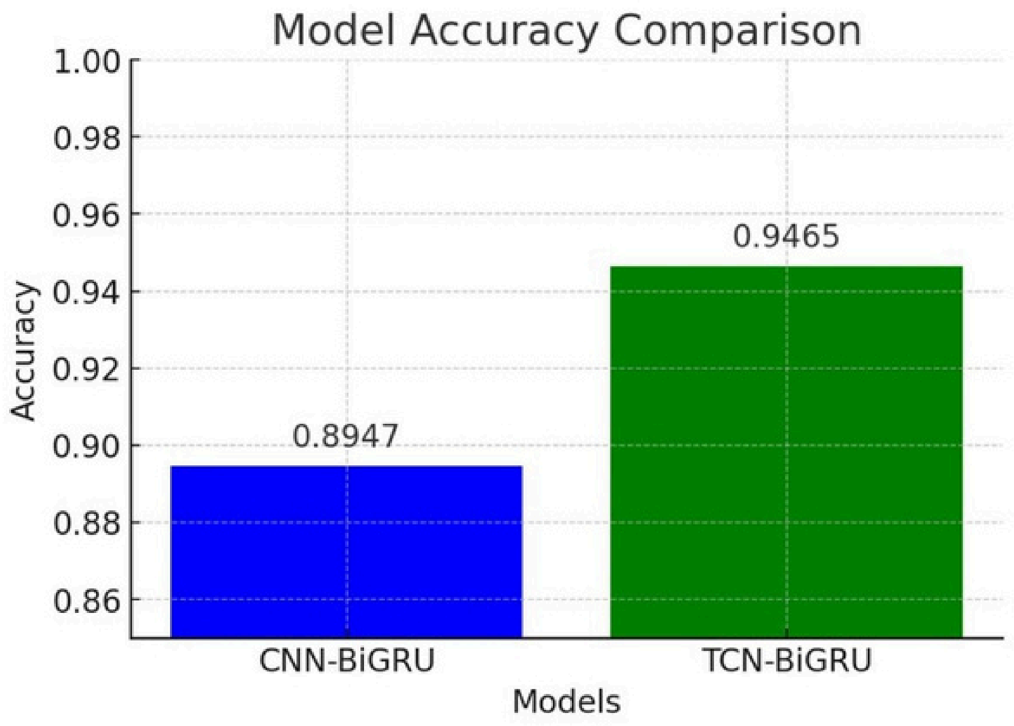
- Train the model using training data.
- Loss: Binary Cross-Entropy Optimizer: Adam.

Evaluation Module

- Evaluate model performance

Results & Output

Quantitative results (tables, graphs)



Testing & Validation

- Test cases / validation strategy

Input: Known malicious API call sequence.

Expected Output: Malware (1)

Purpose: Verify correct detection of malicious behavior.

- Benign Application Sample

Input: Legitimate software API sequence.

Expected Output: Benign (0)

Purpose: Ensure low false positives.

Validation Stratergy

Train-Test Split

80% Training

20% Testing

Internal Validation

10-20% of training data used as validation set.

Used for:

Monitoring overfitting

Hyperparameter tuning

Early stopping

Error Analysis

- False Positives (FP)Benign predicted as Malware.
- False Negatives (FN)Malware predicted as Benign.

Robustness Checks

- Overfitting Check
- Sequence Length Sensitivity
- Hyperparameter Sensitivity
- Generalization Check

Conclusion and Future enhancements

- Summary of work done
- Achievement of objectives

Conclusion and Future enhancements

Summary of work done

- This project focused on the development of a deep learning-based malware classification system using API call sequences
- Traditional signature-based malware detection systems fail to detect zero-day attac

Obfuscated malware,Polymorphic malware variants

To overcome this limitation, a behavior-based detection approach was implemented.

References

- Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). API-MalDetect: Deep Learning for API Call Sequence-Based Malware Detection. arXiv preprint arXiv:2407.13355.

- *Bensaoud, A., & Tekerek, A. (2025). A Survey of Malware Detection Approaches in Windows and Cross-Platform Environments.Journal of Information Security and Applications, 58, 102-113*

THANK YOU

onlineamrita.com