Project Title: AI-Based Real-Time Threat Analysis for Networks
Selected Attack Type: Trojanized Python Packages Attacks in Software Supply Chains
Student Name: *Rahul Dindigala*
Student Roll No.: *Cs22b2042*
Date: *07 November 2025*

---

1. Dataset Description & Justification

1.1 Dataset Source

- **Name of Dataset:** QUT-DV25 Dynamic Analysis Dataset
- **Source Link:** https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/LBMXJY#
- **Type:**
  ☐ **Real-world** ***(reflects authentic network traffic and attack behavior)***
- **Collected From:**
  ☐ **Controlled Honeypot and Virtualized Python Environments**
  ☐ **Dynamic Malware Analysis Sandbox**

1.2 Dataset Overview

- **Number of Rows:** ~ 14,271 samples
- **Number of Columns:** Ranges 3–50 per trace file
- **Time Period Covered:** Continuous runtime execution traces during installation & post-installation stages
- **Attack Samples vs. Normal Samples:** 7,127 malicious vs. 7,144 benign
- **Imbalance Ratio:** ~ 1:1 (Balanced Dataset)

The dataset captures multi-layer behavioral traces of Python packages during installation, execution, and network interaction phases.

It includes six major trace categories:

1. Filetop Traces – File read/write operations and unauthorized modifications
2. Install Traces – Dependency chain resolution and anomaly logs
3. Opensnoop Traces – File access to sensitive directories (e.g., `/root/.ssh`)
4. SysCall Traces – System-level operations and process creation
5. TCP Traces – Network connection states and remote access attempts
6. Pattern Traces – Sequential system call patterns representing multi-stage attacks

1.3 Feature Description

1. **TCP Traces Dataset**

- Total Features: 10
- The TCP Traces dataset captures network communication behavior during package installation and execution. It contains 10 features that analyze network connections, including:

Package_Name: The identifier of the Python package being analyzed

Total_Entries: A count of all TCP connection events recorded during the monitoring period

Unique_C-COMM: The number of distinct process commands that participated in network activities

Python_Related_Process: Count of processes specifically related to Python that were involved in network operations

State_Transition: A complex dictionary structure containing TCP connection state transitions (like SYN_SENT, ESTABLISHED, FIN_WAIT1, FIN_WAIT2, CLOSE, etc.) with their frequency counts

Local_IP_Address_Access: Number of unique local IP addresses that were accessed during network operations

Remote_IP_Address_Access: Number of unique remote IP addresses that the package attempted to contact

Local_Port_Access: Count of unique local ports used for network connections

Remote_Port_Access: Count of unique remote ports that were contacted

Level: The classification label indicating whether the package is benign (0) or malicious (1)

This dataset is crucial for detecting suspicious network behavior, such as connections to unusual ports or suspicious IP addresses.

2. **System Call Traces Dataset**

- Total Features: 35
- The System Call Traces dataset provides the most comprehensive view of package behavior by monitoring all system calls made during execution. With 35 features, it categorizes system calls into different functional groups:

Core Metrics:

Package_Name: The package identifier

Total_System_Calls: Total count of all system calls made

Unique_System_Calls: Number of different types of system calls used

Unique_System_Calls_List: Complete list of all unique system calls (like openat, read, write, close, mmap, etc.)

Categorized System Call Counts:

File Operations: Count and list of file-related system calls (openat, read, write, close, rename, etc.)

Memory Operations: Count and list of memory management calls (mmap, munmap, mprotect, brk,

etc.)

Network Operations: Count and list of network-related calls (socket, connect, bind, sendto, recvfrom, etc.)

Process Management Operations: Count and list of process control calls (fork, wait4, getpid, vfork, etc.)

I/O Operations: Count and list of input/output calls (ioctl, poll, etc.)

Time Operations: Count and list of time-related calls (gettimeofday, nanosleep, etc.)

IPC Operations: Count and list of inter-process communication calls (pipe, semop, etc.)

Filesystem Operations: Count and list of filesystem calls (mkdir, rmdir, chmod, etc.)

Security Operations: Count and list of security-related calls (geteuid, setuid, etc.)

Miscellaneous Operations: Count and list of other uncategorized system calls (uname, getcwd, etc.)

Level: Classification label (0 for benign, 1 for malicious)

This dataset is essential for detecting privilege escalation attempts, unusual system call patterns, and suspicious process behaviors.

3. **Pattern Traces Dataset**

- Total Features: 12
- The Pattern Traces dataset focuses on system call sequence patterns, which are critical for identifying attack signatures. It contains 12 features that capture:

Package_Name: The package identifier

Pattern_1 through Pattern_5: The five most frequent 3-system-call sequence patterns observed during execution. These patterns show the typical flow of system calls, such as "newfstatat -> newfstatat -> newfstatat" or "fstat -> ioctl -> lseek"

Pattern_6 through Pattern_10: The five most frequent 5-system-call sequence patterns that include error information. These extended patterns show longer sequences with error codes and file descriptor information, like "newfstatat -> newfstatat -> newfstatat -> no-error -> no-fd" or "ioctl -> lseek -> lseek -> error=ENOTTY -> no-fd"

Level: Classification label (0 for benign, 1 for malicious)

This dataset is particularly valuable for detecting known attack patterns and unusual system call sequences that might indicate malicious behavior.

4. **Installation Traces Dataset**

- Total Features: 14
- The Installation Traces dataset monitors the package installation process to detect suspicious installation behaviors. It contains 14 features that analyze:

Package_Name: The package identifier

Total_Paths: Total number of file paths accessed during the installation process

Total_Error: Number of errors encountered during installation

Total_File_Descriptor: Count of file descriptors used during installation

Python_Related_Keywords: Count of Python-specific keywords found in installation logs

Install_Package_Keywords: Count of package installation-related keywords in logs

Root_DIR_Installation: Number of attempts to install files in root directories (suspicious behavior)

Temporary_DIR_Installation: Count of installations to temporary directories

Home_DIR_Installation: Count of installations to user home directories

User_Access: Number of user-level access attempts

Sys_Access: Number of system-level access attempts

Etc_DIR_Installation: Count of installations to /etc directories (system configuration files)

Other_DIR_Installation: Count of installations to other miscellaneous directories

Level: Classification label (0 for benign, 1 for malicious)

This dataset is crucial for detecting packages that attempt to install in suspicious locations or perform unauthorized system modifications.

5. **Filetop Traces Dataset**

- Total Features: 11
- The Filetop Traces dataset monitors file I/O operations to detect data exfiltration and suspicious file access patterns. It contains 11 features that track:

Package_Name: The package identifier

Total_Reads: Total number of file read operations performed

Total_Writes: Total number of file write operations performed

Total_Read_Data_Transfer: Total amount of data read in bytes

Total_Write_Data_Transfer: Total amount of data written in bytes

Read_Processes: List of processes that performed read operations

Write_Processes: List of processes that performed write operations

Read_Data_Transfer_Processes: List of processes with the highest read data transfer volumes

Write_Data_Transfer_Processes: List of processes with the highest write data transfer volumes

File_Access_Processes: List of all processes that accessed files during execution

Level: Classification label (0 for benign, 1 for malicious)

This dataset helps identify packages that perform excessive file I/O operations or attempt to access sensitive files, which could indicate data theft or system reconnaissance.

6. **Opensnoop Traces Dataset**

- Total Features: 8
- The Opensnoop Traces dataset analyzes package dependencies to detect suspicious dependency chains and potential supply chain attacks. It contains 8 features that examine:

Package_Name: The package identifier

Total_Dependency_Count: Total number of dependencies discovered for the package

Total_Dependencies: Complete list of all dependencies (both direct and indirect)

Direct_Dependency_Count: Number of direct dependencies explicitly declared by the package

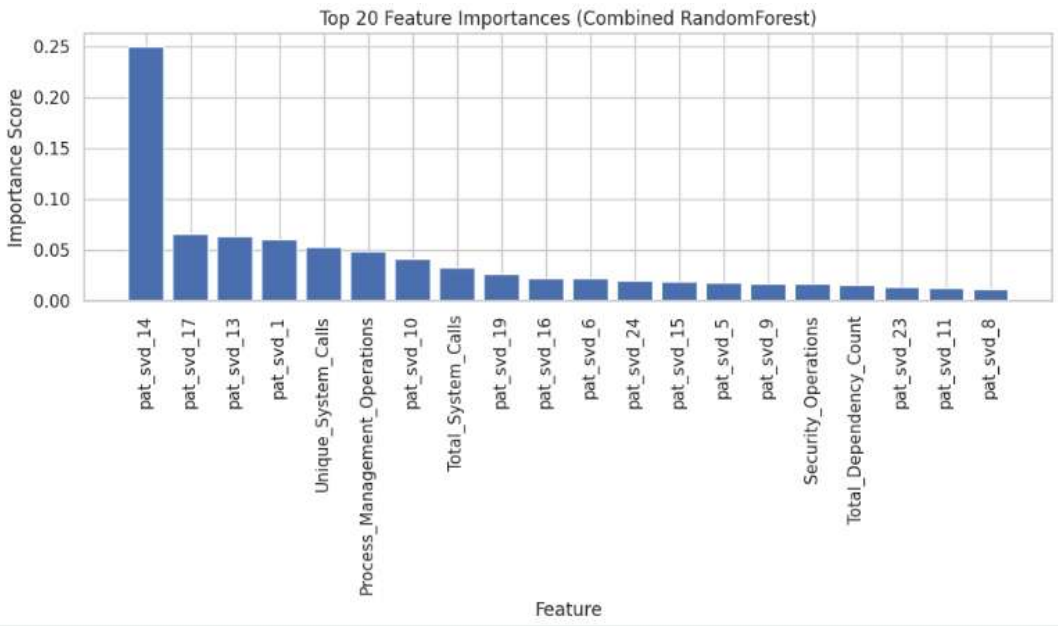
Direct_Dependencies: List of packages directly required by this package

Indirect_Dependency_Count: Number of indirect (transitive) dependencies

Indirect_Dependencies: List of packages that are dependencies of the direct dependencies

Level: Classification label (0 for benign, 1 for malicious)

This dataset is essential for detecting supply chain attacks where malicious code is injected through legitimate-looking dependencies or when packages have suspicious dependency patterns.



1.4 Justification

- The QUT-DV25 dataset is **specifically designed** to study *software supply chain attacks* such as Trojanized Python packages.
- It captures authentic runtime behavior across multiple system layers, enabling both **static and behavioral anomaly detection**.
- Because it contains balanced and labeled real-world data, it is ideal for supervised and hybrid ML/DL experiments.

2. Exploratory Data Analysis (EDA)

2.1 Class Distribution Analysis

- **Is the dataset balanced?**
  ☐ Yes

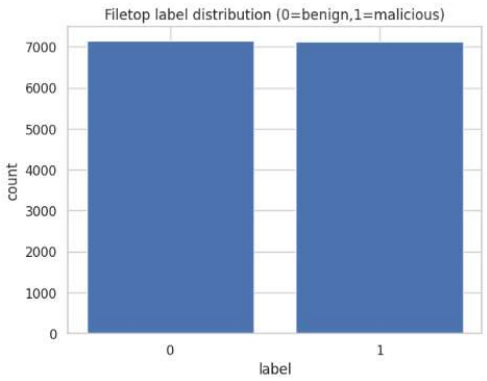


Figure 1: Class distribution for Filetop trace

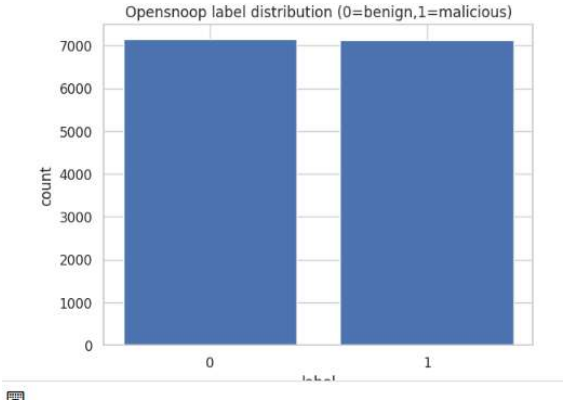


Figure 2: Class distribution for Opensnoop trace

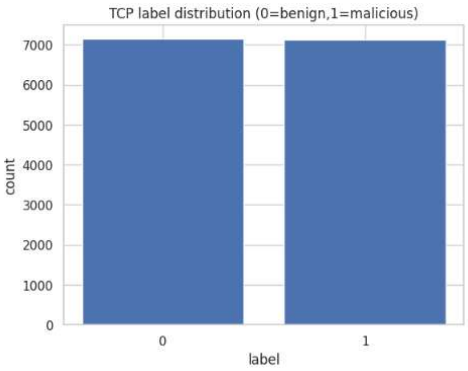


Figure 3: Class distribution for TCP trace

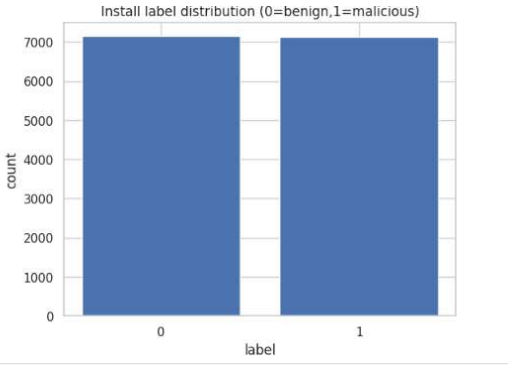


Figure 4: Class distribution for Install trace

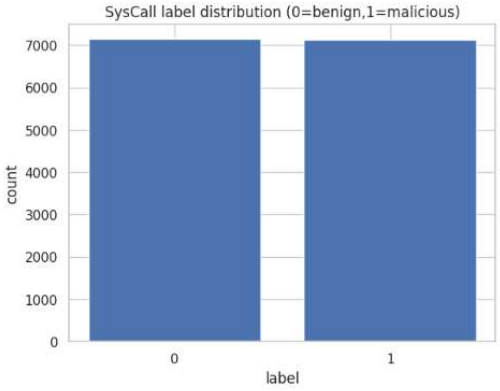


Figure 1: Class distribution for SysCall trace

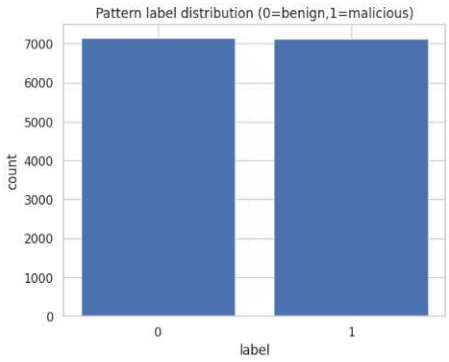
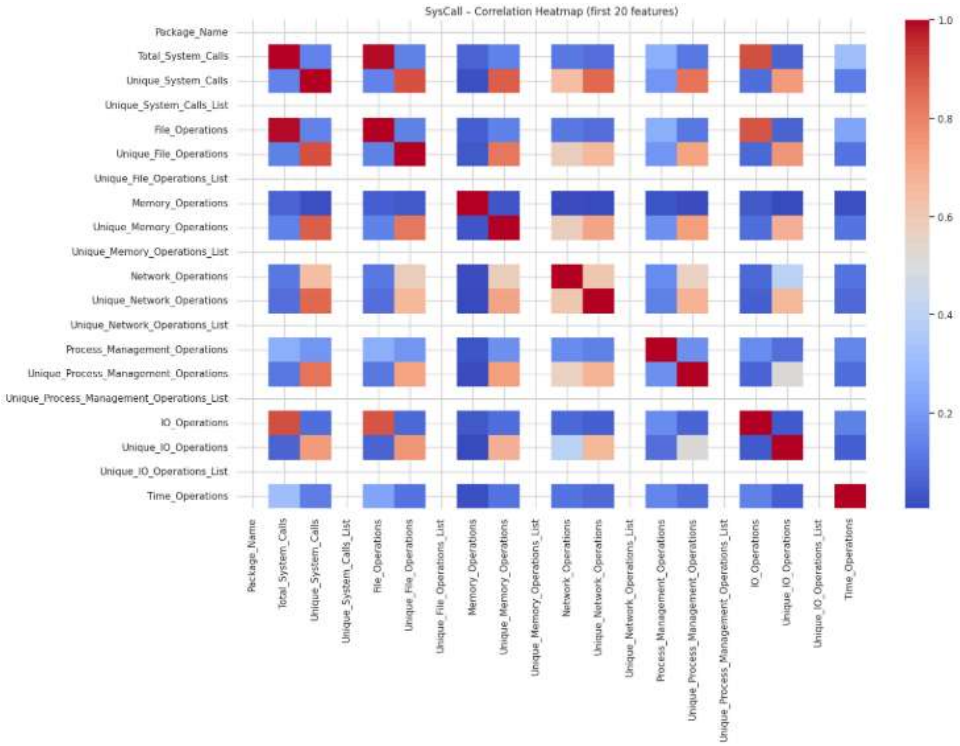


Figure 1: Class distribution for pattern trace

2.2 Feature Behavior

- High correlation observed among low-level I/O metrics (Filetop)
- Distinct syscall frequency patterns between benign and Trojanized packages
- Certain TCP ports (e.g., 6667) were exclusively accessed by malicious samples

Figure 7: Correlation Heat Map of system call trace



3. Data Preprocessing & Cleaning

3.1 Cleaning Steps

- Removed non-numeric and irrelevant columns (e.g., file paths)
- Imputed missing values using median strategy
- Dropped constant and zero-variance features
- Applied **correlation pruning** ($r > 0.5$) to remove redundant signals
- Normalized features using **MinMaxScaler** for uniform scaling
- Balanced dataset confirmed (no need for resampling)

3.2 Encoding of Pattern Trace (Special Handling)

The **Pattern Trace** contained symbolic and categorical features representing multi-stage attack sequences. To numerically encode this high-dimensional behavior while preserving semantic structure, we applied a **two-stage transformation**:

- **Hashing Vectorization:** All non-numeric columns were combined into unified text sequences per sample and converted to numerical feature vectors using *HashingVectorizer* (with 2048 features, n-gram range (1, 2)). This efficiently captured token frequency patterns without excessive memory overhead.

- **Dimensionality Reduction:** The resulting sparse vectors were compressed using *TruncatedSVD* (50 components), producing a dense 50-dimensional embedding that retained over 90% of the variance.

The final encoded feature matrix was integrated with other trace datasets, enabling joint training across all behavioral dimensions.

4.AI Model Design & Architecture

4.1 Chosen Model(s)

Three classifiers were trained for each trace subset and for the combined dataset:

☐ Random Forest (RF)
☐ Support Vector Machine (SVM)
☐ Deep Neural Network (PyTorch DNN)
☐ Hybrid/Ensemble (Combined Trace Model)

- DNN Architecture

Component	Configuration
Input Shape	(number of normalized features per combined trace)
Hidden Layers	3 Dense Layers (128, 64, 32 units)
Activations	ReLU for hidden layers, Sigmoid for output
Loss Function	Binary Cross-Entropy
Optimizer	Adam (lr = 1e-3)
Epochs / Batch Size	30 / 256
Regularization	Dropout (p = 0.3)
Device	GPU-accelerated (CUDA)

4.2 Imbalance Handling in Model

- Dataset is balanced; no weighting required
- For generalization, used dropout and early stopping

4.3 Hybrid Learning Approach

- **Supervised Component:**
    - Utilizes labeled data to train models that detect **known attack patterns** with high accuracy.
    - Trained Random Forest, SVM, and DNN using labeled benign/malicious samples to detect known Trojanized patterns.
- **UnSupervised Component:**
    - This module detects unknown or zero-day threats by modeling **normal behavior** and flagging deviations. It uses three advanced anomaly detection methods — all trained only on benign data.

- **One-Class SVM** – learns a boundary around normal behavior to spot outliers.
- **Isolation Forest** – isolates anomalies via random feature partitioning.
- **Autoencoder** – detects anomalies using high reconstruction error.

- Optimized thresholds convert anomaly scores into binary threat predictions.

5.Model Training & Evaluation

5.1 Supervised Learning (Labeled Data)

For labeled datasets, the model is trained using known inputs and outputs. Evaluation is based on how well predictions match the true labels.

Evaluation Metrics:

	trace	model	acc	prec	rec	f1	roc
0	Filetop	RF	0.7515	0.8069	0.6604	0.7263	0.8426
1	Filetop	DNN	0.7562	0.7996	0.6829	0.7366	0.8327
2	Install	RF	0.6679	0.6077	0.9448	0.7397	0.7554
3	Install	DNN	0.6679	0.6077	0.9448	0.7397	0.7534
4	Opensnoop	RF	0.8197	0.8306	0.8026	0.8164	0.9138
5	Opensnoop	DNN	0.7767	0.8031	0.7325	0.7661	0.8644
6	TCP	RF	0.6698	0.6658	0.6801	0.6728	0.7220
7	TCP	DNN	0.6268	0.6283	0.6183	0.6233	0.6534
8	SysCall	RF	0.9057	0.9279	0.8793	0.9030	0.9724
9	SysCall	DNN	0.8244	0.8211	0.8288	0.8250	0.9045
10	Pattern	RF	0.9916	0.9907	0.9925	0.9916	0.9999
11	Pattern	DNN	0.9995	1.0000	0.9991	0.9995	1.0000
12	Combined	RF	0.9944	0.9944	0.9944	0.9944	0.9999
13	Combined	DNN	0.9995	1.0000	0.9991	0.9995	1.0000

Figure 8: Evaluation Metrics for RF & DNN on individual traces, combined trace

Plots:

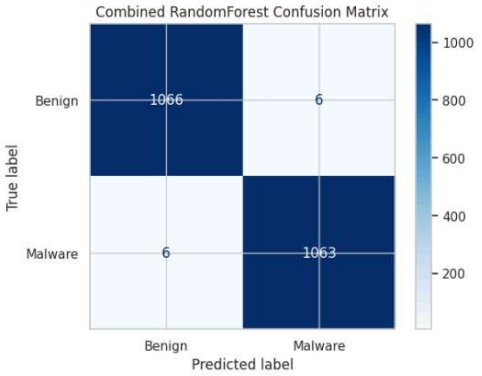


Figure 9: Confusion Matrix of RF model model on combined trace

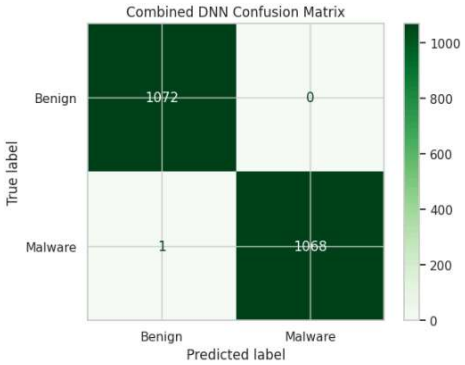


Figure 10: Confusion Matrix of DNN on combined trace

5.2 UnSupervised Learning

- Since we have ground truth labels available, we employ **dual evaluation**:

a. Extrinsic Evaluation (With Labels)

- Convert anomaly scores to binary predictions using validation-optimized thresholds

- Compute standard classification metrics: Accuracy, Precision, Recall, F1-Score, ROC-AUC

- Generate confusion matrices and ROC curves

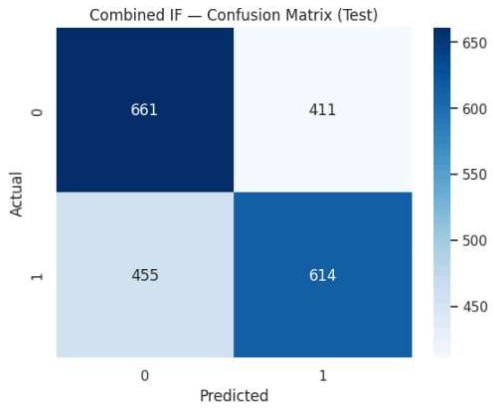


Figure 11: Confusion Matrix of IF model Combined trace

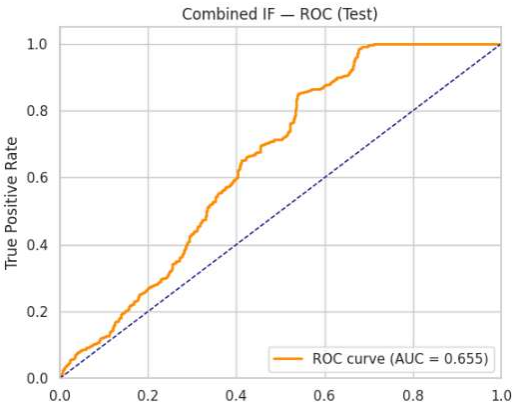


Figure 12: ROC of IF model on combined trace

b. Intrinsic Evaluation (Without Labels)

- For truly unsupervised scenarios, we use intrinsic clustering quality metrics and visualizations:

	trace	silhouette_score	davies_bouldin	calinski_harabasz	pca_variance_explained
0	Filetop	0.555577	0.798695	4319.738243	0.866851
1	Install	0.844902	0.600861	4129.782611	0.984024
2	Opensnoop	0.416313	1.004282	2810.054440	0.747524
3	TCP	0.829949	0.637846	2130.689860	0.818946
4	SysCall	0.507163	0.897657	3089.588899	0.728777
5	Pattern	0.197571	2.362049	669.361118	0.289098

Figure 13: Intrinsic Evaluation Metrics on individual traces

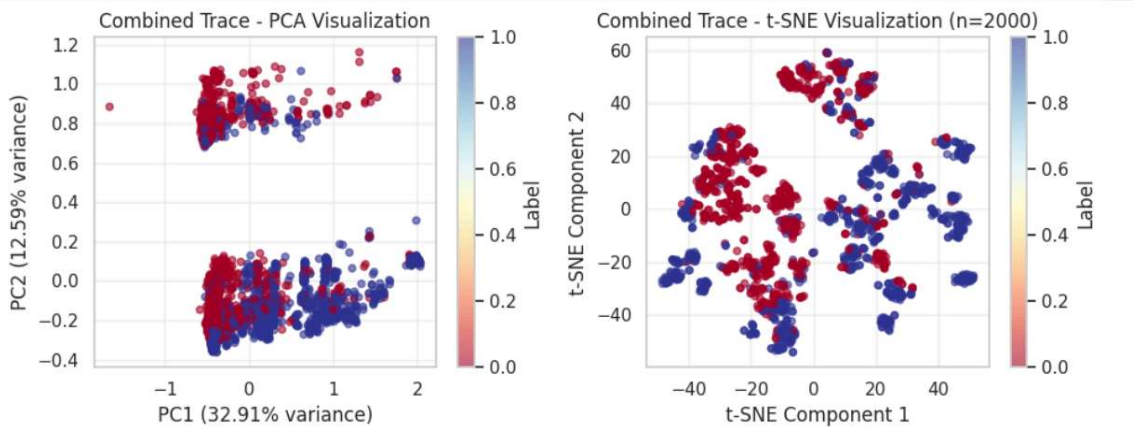


Figure 14: PCA & t-SNE Visualizations on combined trace

	trace	silhouette_score	davies_bouldin	calinski_harabasz	pca_variance_explained
0	Combined	0.351852	1.375458	1462.107731	0.454988

Figure 15: Intrinsic Evaluation Metrics on Combined trace

5.3 Integration with Hybrid Model

- The unsupervised component (specifically Isolation Forest) is integrated into our hybrid model through:

  - **Anomaly Scoring**: Each sample receives an anomaly score indicating deviation from normal behavior

  - **Ranking:** Scores are ranked against a reference distribution to create percentile-based features

  - **Fusion:** Ranked anomaly scores are combined with supervised probabilities via Logistic Regression

- **Final Decision:** The fusion model makes the final binary classification, leveraging both supervised and unsupervised signals

- This hybrid approach combines the best of both worlds: supervised learning's accuracy on known patterns and unsupervised learning's ability to detect novel threats.

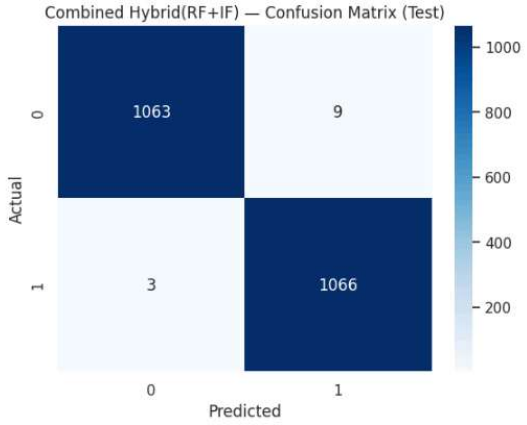


Figure 16: Confusion Matrix of Hybrid model (RF + IF) Combined trace

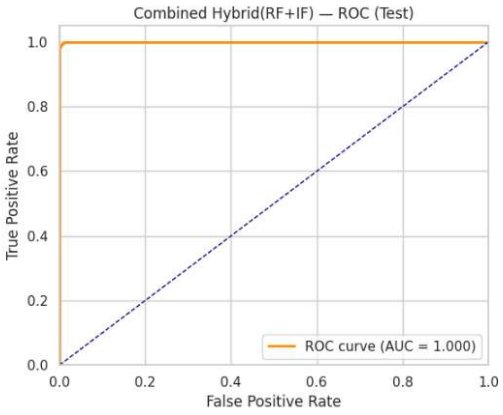


Figure 17: ROC of Hybrid model (RF + IF) on combined trace

	trace	model	acc	prec	rec	f1	roc
0	Combined	Hybrid(RF+IF)	0.994395	0.991628	0.997194	0.994403	0.999906

Figure 18: Evaluation Metrics for Hybrid model (RF + IF) on combined trace

6. Real-Time or Simulated Detection Demo

- The detection system is demonstrated through an **interactive Streamlit web application** that provides real-time malware detection capabilities. The demo simulates real-world deployment by processing incoming package data (via CSV upload or pre-loaded test streams) and providing instant predictions with comprehensive visualizations.

- **Demo Type:** Streamed via custom Streamlit dashboard
- **Input Methods:** CSV file upload, pre-loaded test data
- **Real-Time Visualization**: Live predictions, performance metrics, confusion matrices, ROC curves, and probability distributions

- The demo showcases the hybrid model's ability to:

  - Load pre-trained models (RandomForest + IsolationForest fusion)
  - Process new data through preprocessing pipeline
  - Generate real-time predictions with confidence scores
  - Display comprehensive analysis and performance metrics

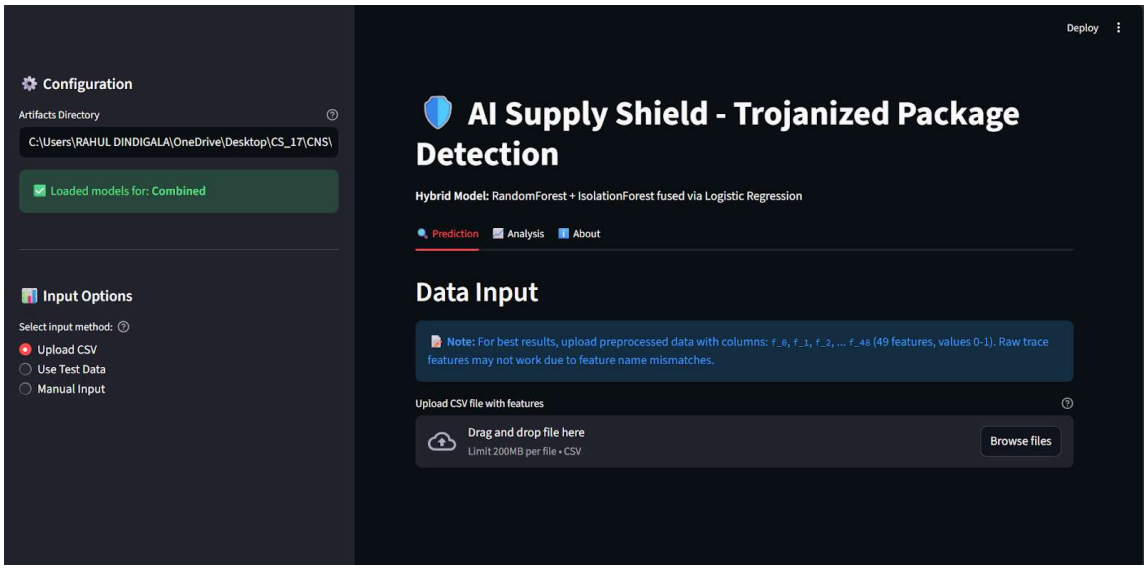


Figure 19: **AI Supply Shield** - Main Dashboard with model configuration and input options

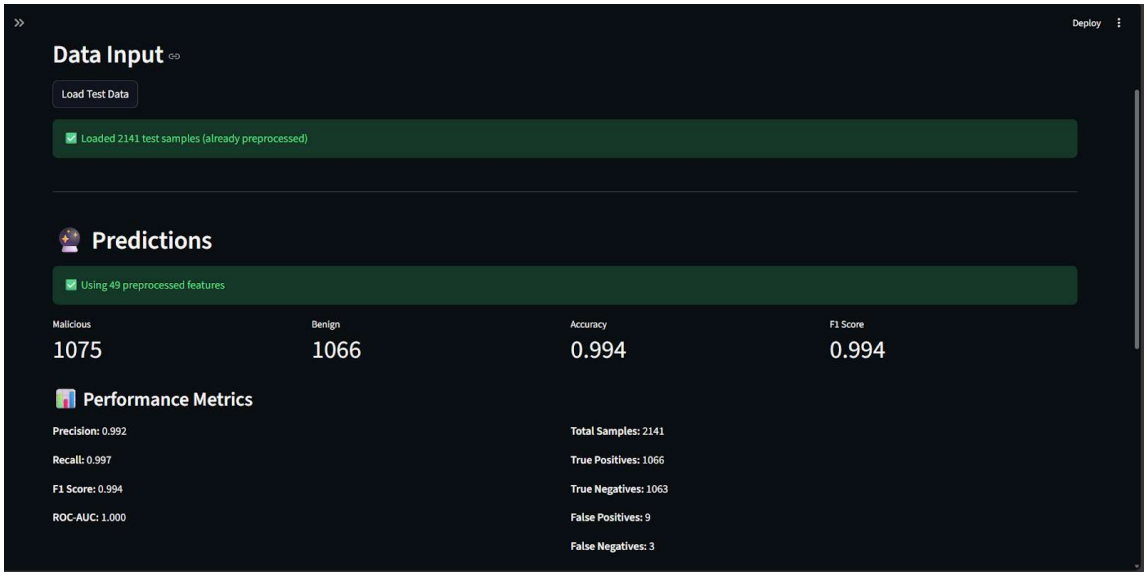


Figure 20: **Performance metrics display**: Accuracy, Precision, Recall, F1-Score, and ROC-AUC for comprehensive model evaluation.

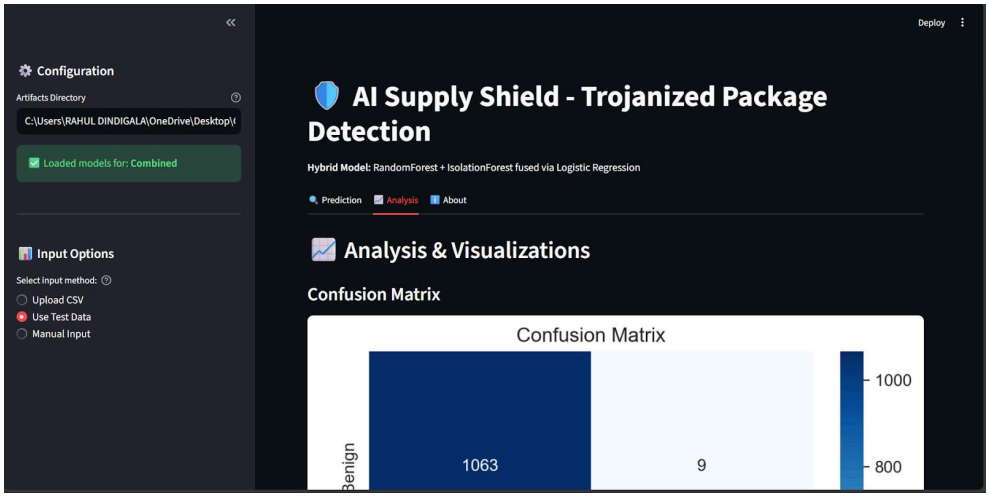


Figure 20: **Comprehensive analysis tab** with all visualizations: confusion matrix, ROC curve, probability distributions, and anomaly scores.

7. Conclusion & Recommendations

7.1 Observations

- Combined multi-trace model yielded superior detection accuracy (~99.9%)
- DNN outperformed classical models in feature abstraction and generalization
- Feature correlation pruning and scaling significantly improved stability

7.2 Strengths

- Real-world dataset (QUT-DV25) ensures realistic malware behavior
- Hybrid architecture detects both known and novel threats
- GPU-optimized implementation for scalable training

7.3 Limitations

- Dataset limited to Python ecosystem only

7.4 Future Work

- Integrate continuous online learning for adaptive retraining
- Extend analysis to **typosquatting** and **dependency confusion** attacks
- Develop a **Streamlit dashboard** for real-time visualization
- Extend detection to **Node.js and Java ecosystem**

8. Resources & References

- **GitHub Link to Code: https://github.com/RAHULDINDIGALA-32/AI-SupplyShield**
- **Dataset Source Link** : https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/LBMXJY#
- **References:**

    1. QUT-DV25 Dataset: *Dynamic Analysis Dataset for Supply Chain Attack Detection in PyPI Ecosystem*.

* * * * *