RansomNet: Ransomware Classification Using Sub-Graph Mining of Function Call Graphs

Garvit Agarwal*, Yucheng Xie†, Yousef Mohammed Y Alomayri*, Feng Li*
*Purdue University, West Lafayette, IN, USA
{agarw347, yalomayr, fengli}@purdue.edu
†Yeshiva University, New York, NY, USA
yucheng.xie@yu.edu

***Abstract*—The classification of ransomware remains a critical yet challenging task in cybersecurity. Motivated by the increasing sophistication and overlap in behaviors between ransomware and general malware, this work addresses the need for more precise differentiation methods to facilitate targeted mitigation efforts. Our study proposes an innovative approach for ransomware classification using sub-graph mining of Function Call Graphs (FCGs). We employ Cuckoo Sandbox™ to extract dynamic API calls and construct detailed FCGs. Through focused subgraph mining, we isolate critical API call patterns specifically relevant to ransomware behavior. These extracted patterns are then vectorized and classified using a Convolutional Neural Network (RansomNet-CNN), achieving high precision in distinguishing ransomware from general malware. Unlike full-graph or flat-sequence models, our subgraph-level approach precisely captures ransomware-relevant behaviors. The RansomNet-CNN model demonstrates superior performance, achieving a precision of 99% and a recall of 100%, thus underscoring its practical effectiveness in ransomware identification. The dataset and code are publicly available at our Zenodo Repository[1].**

***Index Terms*—Ransomware, Malware, Classification, Cuckoo Sandbox, Graph Analysis, CNN**

I. Introduction

Malware [23] encompasses a variety of malicious software designed to disrupt, damage, or gain unauthorized access to computer systems, with ransomware [24] being particularly devastating as it encrypts a victim's data and demands a ransom for decryption. The ability of ransomware to cause immediate financial damage underscores the urgent need for precise identification and tailored mitigation strategies.

In recent years, ransomware has evolved from opportunistic attacks into highly targeted campaigns aimed at critical infrastructure and public services. Recent studies reveal that modern ransomware strains are increasingly engineered to bypass static detection mechanisms, disable system backups, and exploit zero-day vulnerabilities [29], [30]. This strategic evolution necessitates detection systems capable of capturing nuanced behavioral patterns beyond traditional signature-based or binary classification methods. Among various sectors, the healthcare industry has been particularly impacted, where ransomware-induced downtime can directly threaten patient safety. Unlike general malware, ransomware specifically targets and encrypts sensitive data—disrupting hospital workflows and pressuring institutions to pay ransoms or face prolonged operational paralysis. The growing threat is evident from the increase in incidents targeting patient records, which rose from 55% in 2015 to 64% in 2016 [25].

While these statistics highlight the urgency of defending against ransomware, most machine learning-based classification models approach the problem as a binary task—differentiating ransomware from benign software [29]. This framing, however, ignores the fact that ransomware often operates with behavioral signatures that overlap significantly with other malware types [30]. Consequently, detection systems trained to simply detect "maliciousness" may misclassify ransomware as other malware types, delaying containment and recovery strategies in real-time security workflows.

Despite advances in malware detection, existing methods often fall short in specifically identifying ransomware, due to its sophisticated evasion techniques, encryption methods, and the rapid evolution of attack vectors. Common challenges include high obfuscation, the similarity to legitimate software leading to false negatives, and rapidly changing attack signatures that outpace traditional defenses. In addition, Ransomware shares many operational tactics with general malware, such as execution processes and network communications, which can often be misleading when analyzed through static features commonly used in traditional malware detection approaches [3]. This overlap necessitates the exploration of more dynamic and discriminative features to enhance detection.

To address this limitation and effectively tackle these challenges, we develop a novel method, RansomNet, that uses dynamic analysis of post-execution API call patterns. This approach allows us to capture more granular and temporal behaviors that are indicative of ransomware, distinguishing it from conventional malware. Specifically, we utilize Cuckoo Sandbox™ [1] to extract and analyze detailed behavioral data from API calls, constructing Function Call Graphs (FCGs) [31] that visually represent these interactions. Subgraph mining is then employed to isolate significant patterns within these graphs, which are subsequently vectorized and analyzed using a 1D Convolutional Neural Network (1D-CNN). This model is specifically tailored to address the challenge of dynamic feature analysis by efficiently processing complex data representations and achieving high precision in distinguishing ransomware from other malware types.

[1]Agarwal, G. (2024). RansomNet Dataset. Zenodo. https://doi.org/10.5281/zenodo.11177440

Function Call Graphs (FCGs) enable a structured view of runtime behavior, offering visibility into repeated action motifs, system dependencies, and behavioral context that static features or API sequences alone cannot expose. By combining FCG subgraph analysis with a CNN classifier, RansomNet provides a novel, behavior-aware approach to malware type differentiation.

The contributions of our work are as follows:

- We enhance the precision of distinguishing between ransomware and general malware by analyzing distinctive patterns in API call sequences.
- Our approach uses FCGs to visualize and analyze behavioral differences between malware types.
- We focus on network-related and file-related activities in the FCGs to detect distinctive behavioral patterns indicative of specific malware types.
- We deploy a 1D-CNN that processes vectorized API call sequences from the extracted subgraphs for accurate malware classification.

This paper is organized as follows: Section 2 reviews related work in malware analysis. Section 3 details our methodology, including data collection, FCG creation, subgraph mining, and the classification process. Section 4 presents our results and analysis. Section 5 concludes the paper by summarizing our contributions to the field of Ransomware analysis.

II. Related Works

In recent years, numerous studies have explored malware and ransomware detection using both dynamic and static analysis techniques. Static analysis examines malware code without execution to understand its functionality, purpose, and potential impact. Conversely, dynamic analysis involves executing malware in a controlled environment, such as a Sandbox™, to monitor its real-time behavior and interactions with other systems. While static analysis is faster and typically involves extracting API calls directly from code, dynamic analysis often provides superior accuracy by capturing actual runtime behaviors [5], [6].

Many malware classification methods rely extensively on analyzing API calls [2]–[4], [13]–[17]. For instance, Schofield et al. [11] converted API call sequences into binary categorical vectors and applied Term Frequency-Inverse Document Frequency (TF-IDF) analyses, subsequently classifying malware samples using Convolutional Neural Networks (CNN). However, these sequence-based methods typically treat API calls as linear text streams, neglecting the structural relationships and temporal hierarchies inherent in execution flows [32]. Thus, they inadequately distinguish meaningful segments of API sequences essential for accurate classification.

To address these limitations, researchers have explored graph-based representations. Hassen and Chan [26] developed scalable Function Call Graph (FCG)-based classifiers for Windows malware, while Chuang et al. [27] applied graph convolutional techniques to Android malware detection. Recent advancements, such as SeGDroid [35] and MASKDROID [36], utilize sensitive FCG segments or masked graph representations, yet rely on full graph contexts rather than subgraph-level behaviors. Similarly, Jiang et al. proposed the use of FCG embeddings [43], while Singh and Gaurav used graph-based neural networks specifically for ransomware detection [44]. These methods demonstrate that preserving structural dependencies between API calls significantly enhances both model robustness and interpretability, but these methods do not isolate localized behaviors as in our subgraph approach. Additionally, existing graph-centric approaches have not specifically targeted differentiating ransomware from general malware, presenting an important yet underexplored research area.

With the rise of targeted ransomware attacks, studies have specifically addressed ransomware detection using dynamic analysis and machine learning [8], [42]. Hernández-Álvarez et al. [9] employed dynamic analysis combined with machine learning to distinguish between Locker, Encryptor, and benign software by extracting influential dynamic features using Cuckoo Sandbox™. Schoenbachler and Krishnan et al. [10] extended this work by broadly differentiating ransomware from benign software and other malware types, albeit using general behavioral features, leaving room for further specificity and precision.

More recently, research has shifted toward adaptive, lightweight, and real-time ransomware detection strategies, aiming to enhance operational practicality without compromising detection efficacy. CanCal [47], for instance, proposes a lightweight detection system specifically designed for industrial environments, utilizing real-time monitoring and fine-grained behavioral analysis to minimize operational overhead. Similarly, ERW-Radar [45] employs contextual behavioral detection alongside fine-grained content analysis, effectively managing detection of evasive ransomware by dynamically adapting to new obfuscation strategies. These methods highlight the ongoing shift towards more adaptive and context-aware detection solutions. Lightweight alternatives like Mal-HAPGNN [34] and static feature graph construction methods [33], [37] also aim to reduce detection overhead but often sacrifice interpretability or behavior specificity.

Practical operational considerations have also prompted innovative methods like the multi-staged detection framework proposed by [46], which addresses the challenge of ransomware detection amidst substantial I/O overhead. This method emphasizes a critical balance between detection performance and operational efficiency, further underscoring the importance of practical applicability in real-world scenarios. Other real-time strategies target encoding-based evasion [41] and entropy-neutralizing techniques [38], [39], [40], further emphasizing the arms race between detection and obfuscation.

However, despite these advancements, most prior works remain focused on traditional or broadly-defined feature extraction and detection methods [10]–[12], [15]. These methods typically overlook detailed isolation and comparative analysis of behavioral interdependencies specifically distinguishing ransomware from other malware types. A holistic review of ML-based ransomware detection approaches [42] confirms this

gap, particularly in distinguishing ransomware from generic malware beyond binary detection.

To bridge this significant gap, our study proposes RansomNet which is a novel, graph-based methodology that explicitly isolates and captures critical behavioral interdependencies within ransomware. Unlike existing methods, RansomNet employs subgraph mining focused specifically on network and file related behaviors within Function Call Graphs, significantly improving the precision and specificity of ransomware classification. This approach contrasts with conventional full-graph classifiers [31], which may obscure critical localized behaviors through aggregation. Our focused subgraph extraction approach ensures these distinctive behavioral patterns are clearly represented, providing enhanced detection accuracy tailored explicitly for ransomware threats.

III. Methodology

We propose a dynamic analysis approach using Function Call Graphs (FCGs), subgraph mining, and a 1D Convolutional Neural Network to analyze and classify API call sequences to distinguish between ransomware and other malware based on their behavioral patterns. This section details our comprehensive methodology, which includes data collection from Cuckoo Sandbox™ , preprocessing of this data, and specific classification techniques. The entire framework is depicted in Fig 1 and comprises three main phases: dataset preparation, labeled FCG generation, and CNN implementation using mined subgraphs.

A. Dataset Preparation and Generating Labeled Function Call Graphs

For our study, we collected a dataset to capture the dynamic characteristics of each malicious sample. We sourced malware and ransomware samples from various platforms, including NTFS123's MalwareDatabase [18], theZoo [19], Virustotal [20], and several public GitHub repositories. Goodware samples were obtained from the PE Machine Learning dataset [21]. All samples were executed in a controlled environment using Cuckoo Sandbox™, from which we extracted dynamic features such as API calls, PCAP data for network analysis, DLLs, logs, memory dumps, and screenshots. Specifically, we extracted and maintained a sequential list of API calls for each malicious and benign file using Python. The number of samples is detailed in Table I.

TABLE I
Dataset Samples

Sample Type	Number of Samples
Malware	229
Ransomware	394
Goodware	271
Total	944

The generation of labeled Function Call Graphs (FCGs) is a pivotal step in our methodology, encapsulating the behavior of malware and ransomware samples via their API call sequences. This involves extracting API calls from the dynamic analysis reports generated by Cuckoo Sandbox™ and constructing graphs where nodes represent API calls and directed edges denote the sequence of these calls.

1) Function Call Graph Construction: Function Call Graphs (FCGs) use API call sequences as their primary data source and are a recognized method in malware detection research [26], applicable to various operating systems including Windows [27] and Android [28]. Nodes in the graph represent unique API calls, and directed edges indicate the sequence of API calls, providing a visual representation of the sample's execution flow.

2) Node Labeling Based on API Call Category: Each API call captured during execution in Cuckoo Sandbox™ offers insights into the sample's actions. These are categorized based on the type of operation, such as network, process, system, registry, miscellaneous, cryptographic, and file operations [22]. Building on this, each node in the Function Call Graph (FCG) is labeled according to the category of the corresponding API call, focusing on network and file operations as these have been shown to be most effective for classifying malware and ransomware [7].

The categories used for labeling are:

- **Network category API Calls:** Nodes for API calls that facilitate network communication, including data transmission, DNS queries, and network connections.
- **File category API Calls:** Nodes for API calls related to file manipulation, such as creation, deletion, reading, and writing.
- **Other API Calls:** Nodes for all other API calls are labeled as 'other'.

1) For each API call $s_i \in S$, create a node $v_i \in V$.
2) For each consecutive pair $s_i, s_{i+1} \in S$, create a directed edge $e_{i,i+1} = (v_i, v_{i+1}) \in E$.
3) Apply the labeling function L to each node v_i based on the nature of the corresponding API call s_i.

B. Subgraph Mining and Feature Vectorization

In our earlier study [7], we discovered that communicative (network-related) and systemic (file-related) features played a crucial role in enhancing the accuracy of malware classification using basic classifiers such as Random Forest and Decision Trees. Motivated by these findings, our current research emphasizes these communicative and systemic features for subgraph mining. This approach allows us to use network category and file category subgraphs for our classification task.

Given a labeled function call graph $G = (V, E, L)$, with subsets S and R of vertices in V defined as $S = \{v \in V \mid L(v) = \text{network}\}$ and $R = \{v \in V \mid L(v) = \text{file}\}$, we outline the process for extracting subgraphs $G_s = (V_s, E_s)$ and $G_r = (V_r, E_r)$ corresponding to communicative nodes $v_s \in S$ and systemic nodes $v_r \in R$, respectively. The key components of the subgraphs are defined as follows:

- $V_s \subseteq V$ and $V_r \subseteq V$ are the sets of nodes that lie within a maximum distance d from v_s and v_r, including v_s and v_r themselves.
- $E_s \subseteq E$ and $E_r \subseteq E$ are the sets of edges that connect the nodes within V_s and V_r.

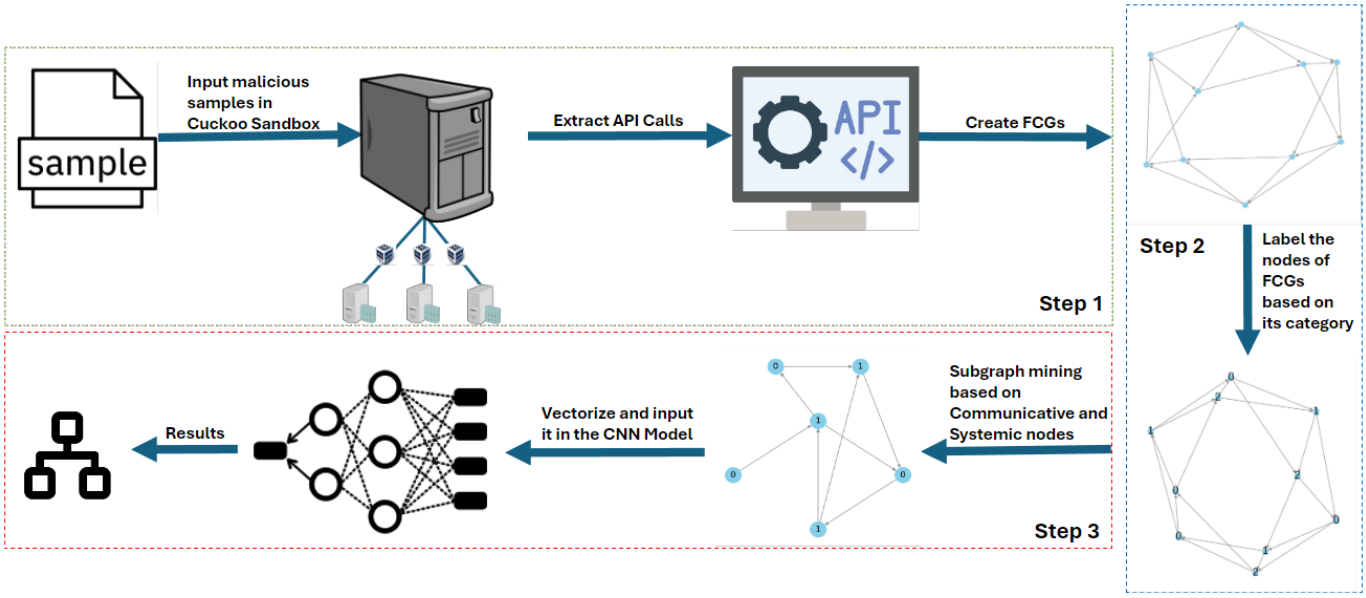

Fig. 1. RansomNet Framework

Subgraph mining is performed on the Function Call Graphs (FCGs) to identify and extract significant patterns indicative of malicious behavior. Each node within an FCG is evaluated to determine if it corresponds to the network category or the file category based on the following structure:

1) Initiate a breadth-first search from each node v in $S \cup R$ limited to a distance d, to identify and assemble the nodes for V_s or V_r.
2) Include in E_s or E_r all edges from E that connect the nodes within V_s or V_r.

The maximum distance d, also known as the Degree of Separation, is chosen based on the desired scope of interaction around communicative and systemic operations within the graph. To find the optimal maximum distance, we employed an experiment with varying degrees of separation in each extracted communicative and systemic subgraph. We deployed our classification model with a small number of samples and extracted varying lengths of subgraphs. Based on this experiment, we used a maximum distance of 10 nodes for each subgraph extraction. The results of this experiment are shown in Figure 2.

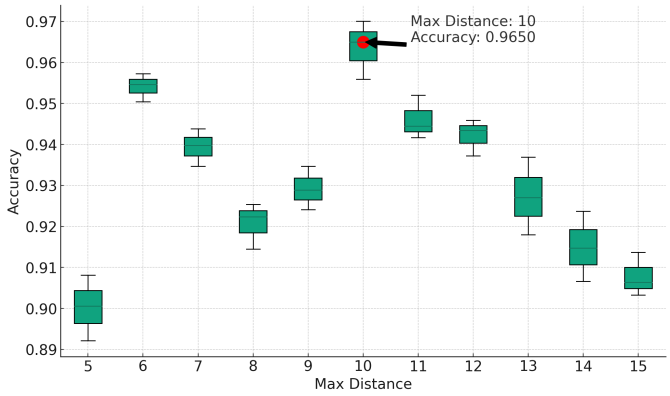

Fig. 2. Optimizing distance for subgraph mining

The extracted subgraphs undergo a process of vectorization, where each subgraph is transformed into a numerical feature vector. This vectorization encapsulates the structural and behavioral characteristics of the subgraphs, including the count of communicative, systemic, and normal nodes, as well as the overall connectivity within the subgraph. The resulting feature vectors serve as the input for the classification model. The structure of this task is detailed in Algorithm 1.

C. CNN-Based Classification

Convolutional Neural Networks (CNNs) have been widely recognized for their efficacy in processing sequential data, such as API call sequences in malware classification [11]. Inspired by these precedents, we designed a 1-D CNN model as the classifier to leverage the sequential nature of API calls for distinguishing between different types of malware.

1) Model Architecture: Our model processes input sequences of API calls represented by the matrix $X \in \mathbb{R}^{l \times d}$,

Algorithm 1 Subgraph Mining and Feature Vectorization

Input: Function call graph $G = (V, E, L)$, set of network nodes $S = \{v \in V \mid L(v) = \text{'network'}\}$, set of file nodes $R = \{v \in V \mid L(v) = \text{'file'}\}$, degree of separation d.

Output: List of feature vectors for classification model derived from subgraphs $G_s = (V_s, E_s)$ and $G_r = (V_r, E_r)$.

```
Initialize list of subgraphs subgraphs
for each node v ∈ S ∪ R do
    Initiate breadth-first search from v, limited to distance d
    Collect all reachable nodes to construct V_s (if v ∈ S) or V_r (if v ∈ R)
    Include in E_s or E_r all edges from E that connect nodes within V_s or V_r
    Add the subgraph (V_s or V_r, E_s or E_r) to subgraphs
end for
Initialize list feature_vectors
for each subgraph G' in subgraphs do
    Vectorize G' by extracting structural and behavioral characteristics
    Append the resulting feature vector to feature_vectors
end for
return feature_vectors
```

where $l = 4$ denotes the sequence length, and $d = 1$ signifies the feature dimension per sequence step. The architecture comprises the following layers:

1) **Convolutional Layer**: Applies filters $W \in \mathbb{R}^{3\times1}$ to input data, extracting features through convolution. Features undergo transformation via the Rectified Linear Unit (ReLU) activation function:

$$C_i = \text{ReLU}\left(\sum_{j=0}^{2} W_j \cdot X_{i+j} + b\right)$$

where i indexes the output sequence, and b as the bias term.

2) **Pooling Layer**: After convolution, an average pooling operation summarizes the feature maps, reducing their dimensionality while retaining essential information:

$$P_i = \text{Average}\left(C_{i\times2}, C_{i\times2+1}\right)$$

3) **Dense and Output Layers**: The sequence ends with a dense layer followed by a softmax layer, classifying input into categories by generating a probability distribution:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

where z_i are logits for each of the N categories.

Figure 3 illustrates the process of dimensionality reduction in our CNN model, providing a clearer understanding of the data flow and transformations within the network.

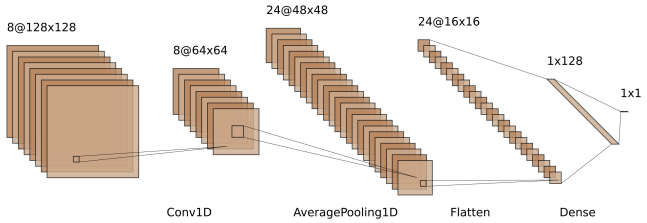

Fig. 3. Detailed design of our CNN architecture.

IV. Results

Following the detailed presentation of our CNN-based methodology, this section provides an analysis of the results obtained from the application of RansomNet to malware classification. We compare our model's performance with established baselines and evaluate its effectiveness in distinguishing between ransomware, malware, and benign software.

A. Dataset and Baselines

Our dataset consists of 612 malicious samples, including 394 ransomware and 218 other malware types, along with 271 benignware samples. We established baselines using three methodologies for malware classification. The first two, inspired by Schofield et al. [11], involve translating API calls into a categorical vector space and applying Term Frequency-Inverse Document Frequency (TF-IDF) to create weighted vectors from API call sequences. The third approach, adapted from Schoenbachler and Krishnan et al. [10], uses dynamic analysis to examine network and file behaviors via Cuckoo Sandbox™ . All experiments and data collection were conducted on a laptop running Ubuntu 20.04, equipped with an Intel Core i7-10750H processor, 16 GB of RAM, and an NVIDIA GeForce GTX 1660 Ti GPU. The training spanned 1000 epochs, yielding an accuracy pinnacle of 98.99%.

Unlike traditional vector-based methods, our study utilizes Function Call Graphs (FCGs) to achieve a deeper and more structured analysis of malware behaviors. This approach is particularly effective in identifying intricate patterns and interactions that help distinguish between different malware types, especially in separating ransomware from generic malicious software. We provide a comparative analysis with these baseline methods in the section on Function Call Graph Construction, demonstrating the advantages of our approach.

In our analysis, we evaluated three binary classification tasks: (1) ransomware vs. general malware, (2) general malware vs. goodware, and (3) ransomware vs. goodware. For each task, we report Accuracy, Precision, Recall and F1-Score. Task (1) represents our primary evaluation-differentiating ransomware from other malware—while Tasks (2) and (3) serve as secondary evaluations to demonstrate RansomNet's generalization and to benchmark its performance against established baselines on both malware–benign and ransomware–benign splits.

B. Ransomware vs Malware

The effectiveness of RansomNet in distinguishing between ransomware and malware showcased outstanding results, achieving a precision of 99% and a recall of 100%. These results not only confirm RansomNet's robustness but also demonstrate its superior performance compared to traditional methods, which typically show lower precision and recall in similar settings. For instance, categorical encoding methods generally achieve around 79% in both metrics, and even the more sophisticated TF-IDF vectorization methods do not surpass a precision of 93% and recall of 94%. Our approach significantly outperforms these by focusing on dynamic analysis which captures the nuanced behaviors of ransomware more effectively.

Moreover, RansomNet attains an overall accuracy of 99% and an F1-score of 99.5%, reflecting the harmonic mean of precision and recall. The area under the ROC curve (AUC) reaches 0.998, indicating near-perfect separability between classes. A closer look at the confusion matrix reveals zero false negatives and a false positive rate below 1%, underscoring the model's reliability in operational environments.

These metrics together demonstrate that mining subgraphs of file and network-related API calls allows RansomNet to capture critical behavioral signatures of ransomware, enabling rapid and precise threat triage in real-world security deployments.

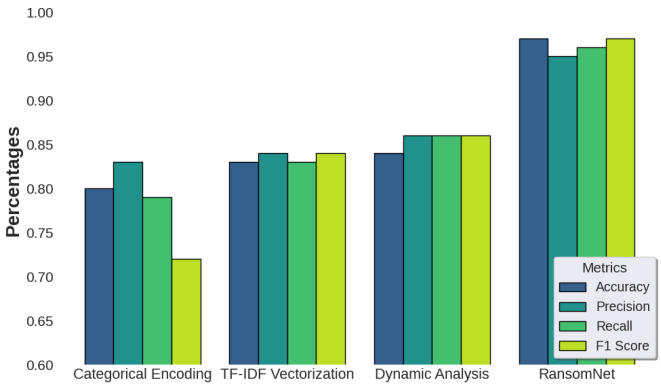

Fig. 4. Ransomware vs Malware Classification

C. File and Network API-Call Pattern Visualization

To better understand the distinction between ransomware and general malware in the dynamic domain, we project the high-dimensional file and network API-call features into a 2D embedding using t-Distributed Stochastic Neighbor Embedding (TSNE). Figure 5 presents this visualization, illustrating clear clusters of ransomware behaviors alongside overlapping regions where separation is more challenging.

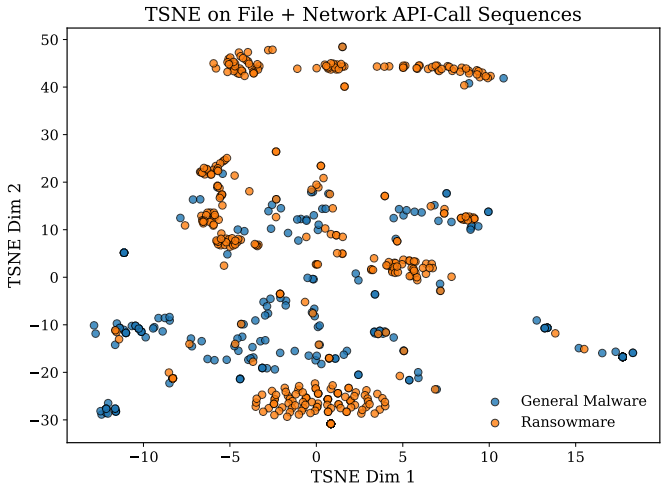

Fig. 5. TSNE projection of file and network API-call sequences for ransomware and general malware.

Figure 5 highlights two main observations: (1) distinct "islands" of ransomware samples, corresponding to families with stereotyped execution patterns, and (2) a mixed central region where ransomware and malware exhibit similar dynamic behaviors. These insights confirm that while file and network calls are discriminative for certain ransomware groups, incorporating richer sequence and contextual features is necessary to resolve ambiguous cases.

D. Ransomware vs Goodware

The classification of ransomware versus goodware (benign software) also highlighted the efficacy of our model. RansomNet achieved a precision of 97% and a recall of 99%, indicating its high accuracy and low rate of false negatives. These results are particularly important in practical scenarios to avoid the costly mistake of misidentifying benign applications as ransomware, which can lead to unnecessary disruption and operations overhead.

TABLE II
RANSOMWARE VS GOODWARE CLASSIFICATION RESULTS

Method	Accuracy	Prec.	Recall	F1 Score
Cat. Enc. [11]	0.84	0.81	0.81	0.82
TF-IDF Vec. [11]	0.94	0.92	0.92	0.93
Dyn. Anal. [10]	0.85	0.88	0.88	0.84
RansomNet	**0.99**	**0.99**	**0.99**	**0.98**

E. Malware vs Goodware

For the task of differentiating malware from goodware, RansomNet again demonstrated strong performance with a precision of 95% and a recall of 96%. The high F1 score of 97% reflects the balanced accuracy and precision of the model, ensuring that it effectively identifies malware while minimizing false positives, which is crucial for maintaining system integrity and user trust in security applications. Furthermore,

TABLE III
MALWARE VS GOODWARE CLASSIFICATION RESULTS

Method	Accuracy	Prec.	Recall	F1 Score
Cat. Enc. [11]	0.80	0.83	0.79	0.79
TF-IDF Vec. [11]	0.83	0.84	0.83	0.84
Dyn. Anal. [10]	0.79	0.82	0.86	0.86
RansomNet	**0.97**	**0.95**	**0.96**	**0.97**

F. Impact Of Classifiers

When we compared our CNN model with traditional algorithms on the same dataset, the CNN model was much superior, outstripping Random Forest by approximately 8%, Decision Tree by nearly 15%, and SVM by a notable 22%. The CNN's hierarchical feature extraction captures local sequence motifs in subgraph-structured API call data, which flat classifiers like RF and DT fail to model effectively These results can be seen in Table IV.

TABLE IV
RANSOMWARE VS MALWARE RESULTS USING DIFFERENT CLASSIFIERS

Classifier	Accuracy	Precision	Recall	F1 Score
Random Forest	0.91	0.88	0.98	0.94
Decision Tree	0.84	0.88	0.94	0.92
SVM	0.77	0.78	0.84	0.87
CNN	**0.99**	**1.00**	**0.99**	**1.00**

G. Consistency and Generalizability Evaluation via 5-Fold Cross-Validation

Further testing the model's robustness, a 5-fold cross-validation revealed consistent accuracy with minimal variance, underscoring the model's generalizability. This is presented in Table V. Unlike simpler TF-IDF or categorical vectorization, which reduces text to a bag of words or tokens, RansomNet's approach to feature extraction—from graph representations of API call sequences—captures the behavioral complexity of ransomware. This technique enables the recognition of sequential and contextual interdependencies within API calls, contributing to the high precision and recall rates.

TABLE V
ACCURACIES OBTAINED FROM 5-FOLD CROSS-VALIDATION.

Fold Number	Accuracy (%)
1	97.76
2	99.23
3	99.61
4	98.69
5	99.61
K-Fold Cross-Validation Accuracy: 98.99 ± 0.69	

The superior accuracy and detailed feature extraction capabilities of the RansomNet model make it exceptionally well-suited for deployment in incident response systems. Its ability to discern between ransomware and general malware ensures that incident responders can quickly isolate and mitigate ransomware threats, significantly enhancing response efficacy.

V. CONCLUSION

In this work, we introduced RansomNet, an innovative framework designed to effectively differentiate ransomware from other types of malware through dynamic analysis, subgraph mining, and convolutional neural network (CNN)-based classification. We utilized Cuckoo Sandbox™ for extracting dynamic API call sequences, subsequently constructing Function Call Graphs (FCGs) to capture intricate behavioral interactions. By applying subgraph mining techniques specifically to network and file-related API call sequences, we successfully isolated distinctive patterns indicative of ransomware behavior.

Our proposed methodology leveraged vectorized representations of these mined subgraphs as inputs for a 1D CNN model (RansomNet-CNN), achieving exceptional classification performance. Specifically, the RansomNet-CNN model demonstrated a precision of 99% and recall of 100

We further validated the robustness and generalizability of our model through extensive evaluation, including comparative analysis with multiple baseline methods and classifiers, as well as 5-fold cross-validation, which consistently demonstrated high accuracy with minimal variance. These results emphasize RansomNet's capability in accurately identifying ransomware by effectively capturing behavioral nuances missed by simpler detection techniques.

Overall, RansomNet provides a sophisticated yet practical solution to ransomware classification challenges, enabling enhanced real-time threat identification and response. In future work, we aim to integrate RansomNet into real-time cybersecurity infrastructures, such as EDR systems or antivirus engines. By streaming dynamic API traces into the subgraph mining pipeline, RansomNet could serve as a lightweight behavioral layer to augment traditional signature-based detection. This hybrid deployment would enable rapid and precise identification of ransomware threats post-execution, even in resource-constrained environments. We also plan to explore adaptive retraining mechanisms to maintain effectiveness against evolving ransomware variants.

REFERENCES

[1] Cuckoo Sandbox™ , "Cuckoo Sandbox: Automated Malware Analysis,". Available: https://cuckoosandbox.org/.

[2] Z. Salehi, A. Sami and M. Ghiasi, "Using feature generation from API calls for malware detection," Computer Fraud & Security, vol. 2014, no. 9, pp. 9–18, 2014.

[3] Y. Cheng, W. Fan, W. Huang and J. An, "A shellcode detection method based on full native API sequence and support vector machine," IOP Conference Series: Materials Science and Engineering, Changsha, China, vol. 242, no. 1, pp. 12124, 2017.

[4] Christodorescu, M., Jha, S.: Static analysis of executables to detect malicious patterns. In: In Proceedings of the 12th USENIX Security Symposium, pp. 169–186 (2003)

[5] Anderson, B., Quist, D., Neil, J. et al. Graph-based malware detection using dynamic analysis. J Comput Virol 7, 247–258 (2011). https://doi.org/10.1007/s11416-011-0152-x

[6] M. Ijaz, M. H. Durad and M. Ismail, "Static and Dynamic Malware Analysis Using Machine Learning," 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2019, pp. 687-691, doi: 10.1109/IBCAST.2019.8667136.

[7] Murli, Sathvik & Nandakumar, Dhruv & Kushwaha, Prabhat & Wang, Cheng & Redino, Christopher & Rahman, Abdul & Israni, Shalini & Singh, Tarun & Bowen, Edward. (2023). Cross-temporal Detection of Novel Ransomware Campaigns: A Multi-Modal Alert Approach.

[8] Madani, H., Ouerdi, N., Boumesaoud, A. et al. Classification of ransomware using different types of neural networks. Sci Rep 12, 4770 (2022). https://doi.org/10.1038/s41598-022-08504-6

[9] Juan A. Herrera-Silva and Myriam Hernández-Álvarez. 2023. Dynamic feature dataset for ransomware detection using machine learning algorithms. Sensors 23, 3(2023),1053. https://doi.org/10.3390/s23031053

[10] Joshua Schoenbachler, Vinay Krishnan, Garvit Agarwal, and Feng Li. 2023. Sorting Ransomware from Malware Utilizing Machine Learning Methods with Dynamic Analysis. In Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23). Association for Computing Machinery, New York, NY, USA, 516–521. https://doi.org/10.1145/3565287.3617632

[11] Schofield, Matthew, Comparison of Malware Classification Methods Using Convolutional Neural Network Based on Api Call Stream (2021). International Journal of Network Security & Its Applications (IJNSA) Vol.13, No.2, March 2021, Available at SSRN: https://ssrn.com/abstract=3822934

[12] Kamran Shaukat, Suhuai Luo, Vijay Varadharajan, A novel machine learning approach for detecting first-time-appeared malware, Engineering Applications of Artificial Intelligence, Volume 131, 2024, 107801, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2023.107801.

[13] Yamany, B.; Elsayed, M.S.; Jurcut, A.D.; Abdelbaki, N.; Azer, M.A. A Holistic Approach to Ransomware Classification: Leveraging Static and Dynamic Analysis with Visualization. Information2024,15,46. https://doi.org/10.3390/info15010046

[14] G. Liang, J. Pang, and C. Dai, "A Behavior-Based Malware Variant Classification Technique," Int. J. Inf. Educ. Technol., vol. 6, pp. 291–295, 2016.

[15] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," J. Comput. Virol. Hacking Tech., vol. 12, no. 2, pp. 59–67, 2016.

[16] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," Int. J. Distrib. Sens. Networks, vol. 2015, no. 6: 659101, pp. 1–9, 2015.

[17] C.-I. Fan, H.-W. Hsiao, C.-H. Chou, and Y.-F. Tseng, "Malware Detection Systems Based on API Log Data Mining," in 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015, pp. 255–260.

[18] NTFS123, "MalwareDatabase," GitHub repository, [Online]. Available: https://github.com/NTFS123/MalwareDatabase.

[19] Ytisf. 2014. YTISF/thezoo: A repository of Live Malwares for your own joy and pleasure. thezoo is a project created to make the possibility of malware analysis open and available to the public. https://github.com/ytisf/theZoo

[20] VirusTotal, "VirusTotal," [Online]. Available: https://www.virustotal.com/.

[21] Michael Lester. Practical Security Analytics - PE Malware Machine Learning Dataset. https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/.

[22] Miller, C., Glendowne, D., Cook, H., Thomas, D., Lanclos, C., Pape, P.: Insights gained fromconstructing a large scale dynamic analysis platform. Dig. Invest. 22, S48–S56 (2017)

[23] Kramer, S., Bradfield, J.C. A general definition of malware. J Comput Virol 6, 105–114 (2010). https://doi.org/10.1007/s11416-009-0137-1

[24] Ali, Azad. "Ransomware: A research and a personal case study of dealing with this nasty malware." Issues in Informing Science and Information Technology 14 (2017): 087-099.

[25] Thamer, Noor, and Raaid Alubady. "A survey of ransomware attacks for healthcare systems: Risks, challenges, solutions and opportunity of research." 2021 1st Babylon International Conference on Information Technology and Science (BICITS). IEEE, 2021.

[26] Mehadi Hassen and Philip K. Chan. 2017. Scalable Function Call Graph-based Malware Classification. In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY '17). Association for Computing Machinery, New York, NY, USA, 239–248. https://doi.org/10.1145/3029806.3029824

[27] H. -Y. Chuang, J. -L. Chen and Y. -W. Ma, "Malware Detection and Classification Based on Graph Convolutional Networks and Function Call Graphs," in IT Professional, vol. 25, no. 3, pp. 43-53, May-June 2023, doi: 10.1109/MITP.2023.3264509.

[28] Y. Du, J. Wang and Q. Li, "An Android Malware Detection Approach Using Community Structures of Weighted Function Call Graphs," in IEEE Access, vol.

[29] Jawad, Shayma & Ahmed Salman, Hanaa. (2024). Machine Learning Approaches to Ransomware Detection: A Comprehensive Review. International Journal of Safety and Security Engineering. 14. 1963-1973. 10.18280/ijsse.140630.

[30] B. P. Gond and D. P. Mohapatra, "Deep Learning-Driven Malware Classification with API Call Sequence Analysis and Concept Drift Handling," arXiv preprint arXiv:2502.08679, Feb. 2025. [Online]. Available: https://arxiv.org/html/2502.08679v2

[31] H. Shokouhinejad, R. Razavi-Far, H. Mohammadian, M. Rabbani, S. Ansong, G. Higgins, and A. A. Ghorbani, "Recent Advances in Malware Detection: Graph Learning and Explainability," arXiv preprint arXiv:2502.10556, Feb. 2025. [Online]. Available: https://arxiv.org/html/2502.10556v1

[32] Li, J., Yang, G., & Shao, Y. (2024). Ransomware Detection Model Based on Adaptive Graph Neural Network Learning. Applied Sciences, 14(11), 4579. https://doi.org/10.3390/app14114579

[33] H. Mohammadian, G. Higgins, S. Ansong, R. Razavi-Far, and A. A. Ghorbani, "Explainable Malware Detection through Integrated Graph Reduction and Learning Techniques," arXiv preprint arXiv:2412.03634, Dec. 2024. [Online]. Available: https://arxiv.org/html/2412.03634v1

[34] W. Guo et al., "MalHAPGNN: An Enhanced Call Graph-Based Malware Detection Framework Using Hierarchical Attention Pooling Graph Neural Network," *Sensors*, vol. 25, no. 2, p. 374, 2025, doi: 10.3390/s25020374.

[35] Z. Liu et al., "SeGDroid: An Android Malware Detection Method Based on Sensitive Function Call Graph Learning," *Expert Systems with Applications*, vol. 235, p. 121125, 2024, doi: 10.1016/j.eswa.2023.121125.

[36] J. Zheng et al., "MASKDROID: Robust Android Malware Detection with Masked Graph Representations," *arXiv preprint arXiv:2409.19594*, 2024. [Online]. Available: https://arxiv.org/abs/2409.19594

[37] B. Zou et al., "Feature Graph Construction with Static Features for Malware Detection," *arXiv preprint arXiv:2404.16362*, 2024. [Online]. Available: https://arxiv.org/abs/2404.16362

[38] J. Bang, J. N. Kim, and S. Lee, "Entropy Sharing in Ransomware: Bypassing Entropy-Based Detection of Cryptographic Operations," *Sensors*, vol. 24, no. 8, p. 1446, 2024, doi: 10.3390/s24081446.

[39] K. Lee et al., "Effective Ransomware Detection Using Entropy Estimation of Files for Cloud Services," *Sensors*, vol. 23, no. 7, p. 3023, 2023, doi: 10.3390/s23073023.

[40] J. Lee and K. Lee, "A Method for Neutralizing Entropy Measurement-Based Ransomware Detection Technologies Using Encoding Algorithms," *Entropy*, vol. 24, no. 2, p. 239, 2024, doi: 10.3390/e24020239.

[41] Lee J, Kim J, Jeong H, Lee K. A Machine Learning-Based Ransomware Detection Method for Attackers' Neutralization Techniques Using Format-Preserving Encryption. Sensors (Basel). 2025 Apr 10;25(8):2406. doi: 10.3390/s25082406. PMID: 40285096; PMCID: PMC12031287.

[42] A. Alenezi and A. Altaleb, "Machine Learning Approaches for Ransomware Detection: A Systematic Literature Review (2020–2024)," *Applied Sciences*, vol. 14, no. 3, p. 1334, 2024, doi: 10.3390/app14031334.

[43] H. Jiang, T. Turki, and J. T. L. Wang, "DLGraph: Malware Detection Using Deep Learning and Graph Embedding," in *Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, USA, 2018, pp. 1029–1033, doi: 10.1109/ICMLA.2018.00168.

[44] D. Singh and K. S. Gaurav, "Graph-Based Deep Learning for Real-Time Ransomware Detection," in *Proc. of the 2024 IEEE Symposium on Security and Privacy (SP)*, pp. 842–857, 2024, doi: 10.1109/SP58591.2024.00123.

[45] L. Zhao, Y. Zhang, Z. Wang, F. Yuan, and R. Hou, "ERW-Radar: An Adaptive Detection System against Evasive Ransomware by Contextual Behavior Detection and Fine-grained Content Analysis," in Proceedings of the 32nd Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, Feb. 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/erw-radar-an-adaptive-detection-system-against-evasive-ransomware-by-contextual-behavior-detection-and-fine-grained-content-analysis/

[46] C. van Sloun, V. Woeste, K. Wolsing, J. Pennekamp, and K. Wehrle, "Detecting Ransomware Despite I/O Overhead: A Practical Multi-Staged

Approach," in Proceedings of the 32nd Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, Feb. 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/detecting-ransomware-despite-i-o-overhead-a-practical-multi-staged-approach/

[47] Shenao Wang, Feng Dong, Hangfeng Yang, Jingheng Xu, and Haoyu Wang. 2024. CanCal: Towards Real-time and Lightweight Ransomware Detection and Response in Industrial Environments. In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS '24). Association for Computing Machinery, New York, NY, USA, 2326–2340. https://doi.org/10.1145/3658644.3690269