A Behavioral and Explainable Real-Time Framework for Ransomware Detection Using System Telemetry

Mohammed Abd Alkareem Abd[1], Wasan Saad Ahmed[1] and Hussain Mahdi[2]
[1]*Department of Computer Science, College of Science, University of Diyala, 32001 Baqubah, Iraq*
[2]*Department of Computer Engineering, College of Engineering, National University of Malaysia, 43600 Bangi, Malaysia*
scicompms242503@uodiyala.edu.iq, wasan@uodiyala.edu.iq, hussain.mahdi@ieee.org

Keywords: Ransomware Detection, Behavioural Analysis, Wazuh, Explainable AI, Zero-Day Detection.

Abstract: Ransomware is one of the most devastating cyber threats that can bypass conventional rule-based defenses and inflict serious damage in a few seconds. Despite the fact that Wazuh, an open-source Security Information and Event Management (SIEM) platform, offers efficient monitoring and alerting features, its fixed rules are weak in identifying zero-day and fast-changing ransomware attacks. The paper suggests a behavioral and explainable real-time ransomware detection framework that combines Wazuh with machine learning and anomaly detection. Over 45,000 Wazuh alerts were gathered in realistic attack scenarios and legitimate user activity. Fourteen behavioral features were obtained, such as file creation, renaming activity, extension changes, and event frequency. The framework uses ten detection models: Random Forest, SVM, KNN, Gradient Boosting, Logistic Regression, Gaussian Naive Bayes, K-Means, Isolation Forest, DBSCAN, and a hybrid Random Forest + DBSCAN model. Detection decisions were explained using SHAP and the most influential features were highlighted. The experimental findings indicate that the highest accuracy of 98.67 was obtained with Random Forest and Gradient Boosting. The framework also offers real-time and early-stage detection, which detects ransomware within 0.01-0.20 s of the attack onset, before massive file encryption takes place. The system also facilitates Telegram alerting and MITRE ATT&CK mapping, which makes it appropriate to be used in real-life applications in Security Operations Centers.

1 INTRODUCTION

Ransomware is one of the most threatening cyber threats of recent years as it can encrypt user files, disrupt essential services, and lead to significant financial and operational losses. Ransomware today does not rely solely on basic file encryption. New attacks are usually characterized by fast file alteration, registry manipulation, process injection, and even double-extortion methods where the attackers threaten to release stolen information unless the ransom is paid. Moreover, the advent of zero-day ransomware has complicated detection since such attacks can evade the conventional signature-based antivirus software. Traditional ransomware detection methods are mainly based on signatures and predefined rules. These techniques are effective against known malware families, but tend to fail against new ransomware variants or previously unknown behaviors. This is why recent research has moved towards behavior-based detection, where the emphasis is on observing the behavior of the system instead of detecting a known malware signature [1], [2]. Behavioral ransomware detection relies on observing low-level system events like process creation, file modification, registry changes, and DNS communication. These activities can be seen prior to the actual encryption phase and thus offer a chance of early detection. A number of studies have shown that system telemetry combined with machine learning can greatly enhance the capability of differentiating between normal and malicious activities [3], [4]. Sysmon and Wazuh are among the most popular monitoring tools in contemporary security settings. Sysmon is capable of capturing detailed operating system events like process creation, file access, registry modification, and file renaming. Wazuh is an open-source Security Information and Event Management (SIEM) platform that offers centralized log collection and analysis. Recent research demonstrated that the combination of Wazuh and machine learning models can enhance the detection of suspicious activities and minimize false positives [5]. In our experiments, we discovered that most of the past research was based on small datasets,

few types of events, or a single operating system. These constraints decrease the generalization capability of the model when applied in a real world. In contrast to the past Wazuh-based solutions, the suggested framework enables real-time and early-stage ransomware detection, explainable machine learning decisions, and detection-delay measurement in 0.01-0.20 s. The main experiment in [5] combined Wazuh with various machine learning algorithms based on about 15,427 security events gathered on 12 endpoints in three months. Their most successful model, Random Forest, had an accuracy of 97.2%. Even though the research showed the efficiency of the integration of Wazuh and machine learning, it lacked explainable AI, real-time notification, ransomware-specific behavioral characteristics, and the ability to detect previously unknown attacks. To overcome these issues, we collected security events for five months from several Windows and Ubuntu endpoints and built a real-time behavior-based framework for ransomware detection. The framework combines Sysmon and Wazuh to continuously collect, correlate, and analyze security events. Over 45,000 alerts were gathered and converted into behavioral sessions that reflected normal and ransomware activity. Fifty-two ransomware behaviors based on MITRE ATT&CK, Atomic Red Team, Microsoft threat intelligence reports, and past ransomware research were examined and transformed into fourteen behavioral features. In contrast to the past, the proposed framework combines both supervised and unsupervised learning to aid in early and zero-day ransomware detection. Six supervised models were used to classify known ransomware behavior: Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gradient Boosting, Logistic Regression, and Gaussian Naive Bayes. In addition, three unsupervised techniques-DBSCAN, K-Means, and Isolation Forest-were employed to detect previously unseen or anomalous behavior. A hybrid Random Forest + DBSCAN model was also proposed to gain confidence when both models concur. The framework also incorporates SHAP explainability to explain model decisions and maps suspicious behavior to MITRE ATT&CK techniques to assist analysts in the investigation. Unlike the past research, the proposed system also facilitates real-time Telegram notification and immediate ransomware detection. In the experiments, the framework could detect malicious behavior in 0.01-0.20 s since the start of the attack, frequently in the initial file creation and renaming processes and prior to large-scale encryption. The greatest contributions of this work can be summarized as follows [6]-[9]:

- Over 45,000 alerts were collected by Wazuh agents on various Windows and Ubuntu endpoints in five months.
- Creation of a dataset of 6,000 labeled behavioral sessions, half of which were normal and the other half ransomware activity.
- Fourteen behavioral ransomware features were extracted out of 52 known ransomware behaviors.
- Comparison of ten detection models, six of which are supervised models, three unsupervised methods, and a hybrid Random Forest + DBSCAN model.
- Maximum detection accuracy of 98.67.
- Support for real-time and early-stage ransomware detection with a detection delay of only 0.01-0.20 s.
- SHAP-based explainability, MITRE ATT&CK mapping, and real-time Telegram alerting.

The rest of this paper is structured in the following way. Section 2 shows the proposed system architecture and framework. Section 3 describes the data collection and feature engineering process. Section 4 describes the machine learning models and anomaly detection strategy. Section 5 discusses the experimental results and compares them with previous work. Lastly, Section 6 summarizes the paper and provides future research directions.

2 PROPOSED FRAMEWORK AND SYSTEM ARCHITECTURE

The proposed framework was designed as a real-time ransomware detection system that monitors the behavior of several endpoints and tries to recognize suspicious activities before the encryption stage becomes large [10], [11]. The system is not solely reliant on signatures since most contemporary ransomware families can evade classical antivirus software. Rather, the framework tracks what the process is executing, the number of files accessed, whether extensions are altered, whether registry keys are altered, and the speed at which the process is running.

Figure 1 illustrates the overall architecture of the proposed real-time ransomware detection framework. The framework consists of five main stages: telemetry collection, feature extraction, hybrid detection, explainable AI (XAI), and decision making.

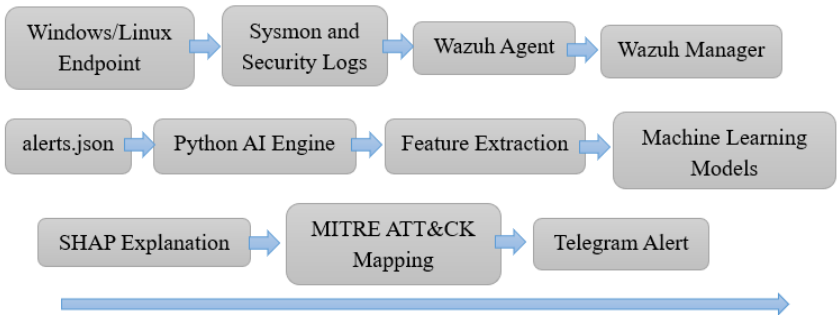

Figure 1: Overall architecture of the proposed real-time ransomware detection framework.

2.1 Event Collection and Filtering

At the Windows side, Sysmon was installed because it provides more detailed logs than the standard Windows Event Log. Wazuh agents were installed on both Windows and Ubuntu machines to forward security events to the central Wazuh manager. Figure 2 illustrates the telemetry collection and alert-generation workflow used in the proposed framework.

The framework did not keep all generated events because many alerts have no direct relation with ransomware behavior. Therefore, a filtering stage was applied before any analysis.

From Windows Security logs, only the following Event IDs were selected:

- 4688: Process Creation;
- 4663: File Access;
- 4657: Registry Modification [12].

From Sysmon, the following events were selected:

- 1: Process Creation;
- 11: File Creation;
- 13: Registry Set Value;
- 15: File Rename;
- 2: File Time Change;
- 22: DNS Query.

These events were selected because most ransomware families perform one or more of these actions. For example, ransomware usually creates a process, starts modifying many files, changes the extension, and sometimes modifies the registry in order to keep persistence.

The Python script reads every line from alerts json and keeps only the events that contain one of the previous Event IDs. If the event does not contain one of these IDs, it is ignored. All filtered events were stored in: /var/ossec/logs/alerts/*.json.

More than 45,000 security events were collected during five months from several Windows and Ubuntu machines.

2.2 Building Behavioral Sessions

During the first experiments, it was found that a single alert is not enough to describe ransomware [13], [14]. A file creation event alone may be normal. But if the same process creates many files, renames them, modifies the registry, and does this very quickly, then the behavior becomes suspicious.

Because of that, the events were grouped into sessions.

Each session contains:

- Events generated by the same Process ID;
- The same Parent Process;
- Events that happened within three minutes.

The selected time window was 180 seconds. Smaller windows were tested, however they sometimes split the ransomware behavior into incomplete parts. Three minutes was enough to keep most related events together.

The session is represented as:

$$S(p,t) = \{e_i \mid PID(e_i) = p,\ t \le time(e_i) \le t + 180\}$$

Where:

- (p) is the process identifier;
- (e_i) is an event;
- 180 means three minutes.

For example, if one process generates:

- 10:00:05 → Process creation;
- 10:00:40 → File creation;
- 10:01:15 → File rename;
- 10:02:20 → Registry modification.

All these events are merged into one session because they belong to the same process and happened inside the selected window.

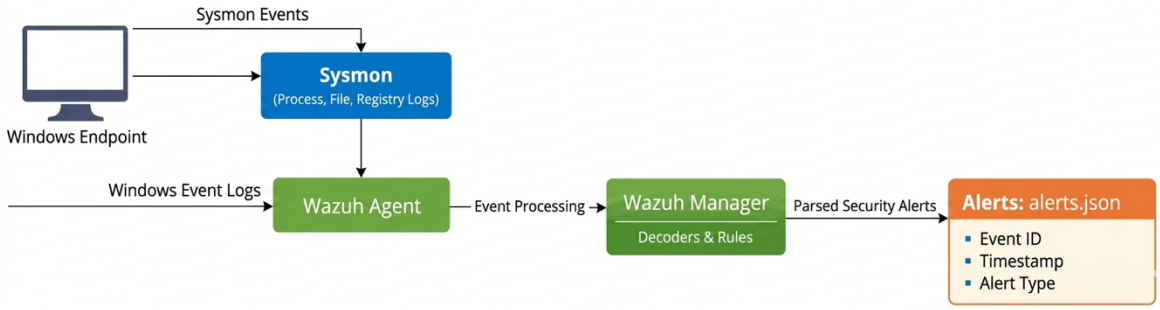

Figure 2: Telemetry collection and alert-generation workflow of the proposed framework..

The generated behavioral sessions are continuously analyzed in real time by the proposed detection framework. Once at least five suspicious events are accumulated within the same session, the extracted behavioral features are immediately passed to the machine learning models. This allows the system to detect ransomware during the early stages of the attack, often within the first file creation and renaming operations and before large-scale encryption occurs.

2.3 From 52 Behaviors to 15 Features

The features used in the framework were not selected directly. To begin with, approximately 52 ransomware behaviors were gathered out of:

- MITRE ATT&CK;
- Atomic Red Team;
- Microsoft Threat Intelligence;
- SANS Institute;
- Kaspersky;
- Sophos;
- Previous ransomware research papers.

The collected behaviors included:

- Encrypting a large number of files within a short time period.
- Renaming files and changing extensions.
- Touching Desktop, Documents and user folders.
- Modifying registry keys.
- Creating many child processes.
- Repeating the same extension on many files.
- Creating temporary files.
- Generating many actions in few seconds.
- Sending suspicious DNS requests.

After collecting these behaviors, many of them were found to be similar. An example is that numerous file changes and numerous file additions both characterize intensive file activity. Execution speed is also characterized by rapid encryption and numerous events within a short period. Thus, the similar behaviors were combined into more general numerical features. Lastly, 15 features were developed:

- events_count;
- unique_files;
- unique_dirs;
- extensions_count;
- processes_count;
- parent_processes;
- rename_count;
- create_count;
- has_registry;
- userdir_ratio;
- nonsystem_ratio;
- dominant_ext_ratio;
- time_span;
- events_per_sec;
- file_activity_ratio.

As an illustration, the extension-changing behaviors were combined into extensions_count and dominant_ext_ratio. Registry persistence and registry modification became has_registry. Rapid execution and excessive operations in small time became events per second. Once this step is completed, every session is converted into a numerical vector like: X = [125, 60, 8, 5, 1, 1, 30, 40, 1, 0.85, 0.90, 0.72, 25.4, 4.92] This vector indicates that the process created 125 events, 60 files in 8 directories, 30 file renames and 40 file creations, changed the registry, primarily user folders, and created about 4.92 events per second in 25.4 s.

2.4 Detection Layer

After extracting the behavioral features, the sessions were passed to two complementary detection layers. The first layer uses supervised learning in order to classify the session as either normal or ransomware [15]. Six monitored algorithms were tested:

- Random Forest;
- Support Vector Machine (SVM);
- K-Nearest Neighbors (KNN);
- Gradient Boosting;

- Logistic Regression;
- Gaussian Naive Bayes.

The second layer uses anomaly detection in order to identify previously unseen ransomware behavior. Three unsupervised techniques were experimented [16]-[18]:

- DBSCAN;
- K-Means;
- Isolation Forest.

DBSCAN was the most successful of the anomaly detection methods with parameter tuning of eps = 2.5 and min samples = 5. DBSCAN identifies suspicious sessions by setting the value to -1, and normal sessions are part of a valid cluster. Thus, the sessions that are not part of any cluster are regarded as potentially malicious and can be a sign of zero-day ransomware activity. Moreover, a hybrid algorithm that combines Random Forest and DBSCAN was proposed. The hybrid model is only triggered when the Random Forest and DBSCAN identify the same session as malicious. This model was applied as a supplementary confidence indicator and not as an alternative to the other models.

2.5 Final Risk Score

The framework is not based on the output of one model. Rather, the overall confidence score is obtained by summing the results of all ten detection models based on their accuracy measured in testing. The models are used to provide a weighted score to the final confidence value. The weight of each model is the same as its measured accuracy. The final framework used the following weights:

- Random Forest = 0.9867;
- SVM = 0.9100;
- KNN = 0.8908;
- Gradient Boosting = 0.9867;
- Logistic Regression = 0.9667;
- Gaussian Naive Bayes = 0.9742;
- Isolation Forest = 0.7792;
- K-Means = 0.7617;
- DBSCAN = 0.8558;
- Hybrid Random Forest + DBSCAN = 0.8800.

The confidence score is calculated as the ratio between the sum of active model weights and the sum of all model weights. If the resulting score exceeds 0.40, the session is classified as ransomware.

2.6 Explainability and Alerting

SHAP was related to the Random Forest model to simplify the decision. SHAP indicates the features that had the most significant impact on the prediction. The most significant features in most ransomware sessions were extensions count, events per second, rename count, unique files, and dominant extratio. After that, the suspicious behavior is mapped to MITRE ATT&CK techniques. For example:

- T1486 → Data Encrypted for Impact;
- T1112 → Registry Modification;
- T1059 → Command Execution.

Lastly, the framework notifies in real-time via Telegram. The message includes the name of the process, the confidence score, the activated machine learning models, the most suspicious features, the corresponding MITRE ATT&CK techniques, the precise detection time, and the delay of the ransomware detection. The detection delay during the experiments was between 0.01 and 0.20 s, which indicates the real-time and early-stage detection of the proposed framework.

3 EXPERIMENTAL SETUP AND VALIDATION

The behavioral data gathered on a number of Windows and Ubuntu endpoints during a period of five months was used to test the proposed framework. Over 45,000 alerts were created and gathered using Sysmon and Wazuh. Nevertheless, most of these alerts did not help in detecting ransomware; hence, only the filtered events as outlined in Section 2 were retained. The final dataset consisted of 6,000 sessions after sorting the remaining events into behavioral sessions. Among these sessions, 3,000 were labeled as normal behavior and the other 3,000 were labeled as ransomware behavior. The dataset was purposely balanced since the initial experiments revealed that the models were biased when the number of normal sessions was much higher than the number of ransomware sessions.

3.1 Labeling Strategy

The training of the machine learning models was preceded by labeling the sessions in a rule-based manner. The regulations were founded on the identical ransomware patterns that were observed numerous times in prior research and threat reports, particularly recurring file renaming, extension alteration, intensive file access, and registry modification [5]. A session was considered ransomware when multiple suspicious behaviors were observed within the same three-minute session. As an illustration, when a single process altered numerous files, altered numerous extensions, and

created numerous actions within a very brief period, then this session was typically considered ransomware. The initial labeling process was done using the following simplified rule [19], [20]:

- events_count > 100;
- extensions_count > 3;
- rename_count > 20;
- events_per_sec > 2.

If these conditions appeared together, then:

- Label = 1.
- Otherwise:
- Label = 0.

These values were not selected directly from the beginning. Different thresholds were tested many times until the generated labels looked more realistic and closer to the ransomware behavior reported in MITRE ATT&CK, Atomic Red Team, and the core Wazuh paper [5].

3.2 Training and Validation

The dataset was split into training and testing after labeling. Around 80% of the sessions were used for training, while the remaining 20% were used for testing.

- Training set: 4,800 sessions;
- Testing set: 1,200 sessions.

Moreover, cross-validation was performed 10 times in training. This was significant since the initial results were very high and there was a fear that some of the models might be overfitting. Thus, the training data was split into ten sections. In every round, nine parts were used for training and one part was used for validation. In the last experiments, ten detection models were tested: six supervised models, three unsupervised methods, and one hybrid Random Forest + DBSCAN model. The labeled dataset was used to train the supervised models, and the unsupervised methods were tested based on their capacity to detect ransomware sessions that were not previously seen. In addition, the real-time performance of the framework was validated by measuring the detection delay from the first suspicious event until the alert generation. The delay measured was between 0.01 and 0.20 s.

3.3 Machine Learning Configuration

Six supervised algorithms, three unsupervised algorithms, and one hybrid model were evaluated in this work. The supervised models were [21]:

- Random Forest;
- Support Vector Machine (SVM);
- K-Nearest Neighbors (KNN);
- Gradient Boosting;
- Logistic Regression;
- Gaussian Naive Bayes;

The unsupervised models were:

- DBSCAN;
- K-Means;
- Isolation Forest.

Moreover, a hybrid Random Forest + DBSCAN model was proposed. The hybrid model is only triggered when the Random Forest and DBSCAN label the same session as malicious. The models were coded in Python with Scikit-learn. Pandas and NumPy were used during preprocessing and feature extraction, while SHAP was later used to explain the final predictions. Some of the most important parameters are listed below:

- Random Forest: 200 trees;
- SVM: RBF kernel with probability = True;
- KNN: 5 neighbors;
- Gradient Boosting: default parameters;
- Logistic Regression: max_iter = 5000;
- Gaussian Naive Bayes: default parameters;
- Isolation Forest: contamination = 0.1;
- K-Means: 2 clusters;
- DBSCAN: eps = 2.5 and min_samples = 5 after StandardScaler normalization.

The experiments were repeated with the DBSCAN values being tuned a few times. The system identified a lot of normal sessions as anomalies when eps was too small. Conversely, some suspicious sessions were not detected anymore when eps was larger. The best result was achieved when eps = 2.5 and min_samples = 5.

3.4 Evaluation Metrics

The performance of the proposed framework was assessed using four widely adopted evaluation metrics: accuracy, precision, recall, and F1-score. These metrics were selected because they are commonly employed to evaluate ransomware detection models and other cybersecurity classification systems [5], [22], [23].

Accuracy represents the overall proportion of correctly classified samples among all evaluated instances. Precision measures the proportion of correctly identified ransomware samples among all samples predicted as ransomware. Recall, also referred to as the true positive rate or sensitivity, indicates the proportion of actual ransomware attacks that are correctly detected by the proposed framework. The F1-score combines precision and recall into a single metric, providing a balanced

assessment of the detection performance, particularly when the dataset is imbalanced.

In these metrics, true positives (TP) and true negatives (TN) represent correctly classified ransomware and benign samples, respectively, whereas false positives (FP) and false negatives (FN) correspond to incorrectly classified samples. In the present study, recall was considered the most critical evaluation metric because failing to detect a ransomware attack may result in severe data loss and system compromise. Therefore, the proposed models were evaluated not only on the basis of accuracy but also according to their ability to maximize ransomware detection while minimizing missed attacks.

3.5 Experimental Environment

The suggested framework was developed with Python and a number of open-source machine learning and cybersecurity tools. Windows hosts were monitored with Sysmon to gather detailed system events, such as process creation, file access, registry modification, file renaming, and DNS activity. Simultaneously, Wazuh was used to consolidate, correlate, and process the generated alerts of both Windows and Ubuntu endpoints. In the experiments, over 45,000 alerts were gathered in five months. The alerts were then filtered and sorted into behavioral sessions based on the process identifier, parent process and a three-minute time window after collection. Every session was then converted into fourteen behavioral features, such as file creation, file renaming, extension changes, registry activity, directory dispersion, and event frequency. Pandas and NumPy were used to implement the behavioral analysis and feature extraction stages. Scikit-learn was used to develop the machine learning and anomaly detection models. The framework tested six supervised models (Random Forest, SVM, KNN, Gradient Boosting, Logistic Regression, and Gaussian Naive Bayes), three unsupervised models (DBSCAN, K-Means, and Isolation Forest), and a hybrid model of Random Forest + DBSCAN. DBSCAN was used with feature normalization with Standard Scaler to enhance the performance of anomaly detection. The optimal DBSCAN settings were achieved after a series of experiments with eps = 2.5 and min samples = 5. To interpret the model, SHAP was linked to the Random Forest model to determine the most significant features that contributed to each detection decision. The most prevalent influential characteristics in the experiments were extensions count, events per sec, rename count, unique files, and dominant ext ratio. The final framework also included a real-time alerting module. In case of ransomware behavior, the system would automatically create a Telegram notification with the confidence score, triggered models, suspicious files, MITRE ATT&CK mapping, precise detection time, and ransomware detection delay. The detection delay during the experiments was 0.01-0.20 s, which proves the real-time and early-stage detection of the proposed framework.

4 RESULTS AND DISCUSSION

Both supervised and unsupervised machine learning methods were used to test the proposed framework. This was not just to detect familiar ransomware behavior, but also to explore whether the framework can detect previously unknown attacks.

4.1 Supervised Learning Results

Table 1 shows the performance of the supervised learning models. Random Forest and Gradient Boosting were the most successful of all the methods tested with an accuracy of 98.67. Gradient Boosting offered a little better generalization, whereas Random Forest was especially helpful since it also supported SHAP-based explainability. Gaussian Naive Bayes and Logistic Regression also yielded good results with an accuracy of 97.42 and 96.67 respectively. The Support Vector Machine had an accuracy of 91.00, and K-Nearest Neighbors had the lowest performance of the supervised models. During the experiments, KNN was sensitive to the shape of the feature space and its performance decreased when the behavioral sessions became more complex.

Table 1: Evaluation results of supervised learning models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Random Forest	98.67	99.21	98.00	98.60
Gradient Boosting	98.67	99.33	97.83	98.57
Gaussian Naive Bayes	97.42	98.81	95.83	97.30
Logistic Regression	96.67	97.95	94.83	96.37
Support Vector Machine	91.00	95.12	86.00	90.33
K-Nearest Neighbors	89.08	93.44	82.67	87.73

Table 2: Evaluation of anomaly detection and hybrid models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Hybrid RF + DBSCAN	88.00	92.10	83.50	87.59
DBSCAN	85.58	88.74	80.33	84.32
Isolation Forest	77.92	81.44	69.17	74.80
K-Means	76.17	79.32	66.50	72.36

Table 3: Performance comparison with related work.

Study	Dataset	Best Model	Accuracy (%)
Core Wazuh-based study [5]	15,427 alerts from 12 endpoints	Random Forest	97.20
Early ransomware detection study [23]	Behavioral file activity dataset	Random Forest	96.30
Explainable ransomware detection [2]	Dynamic behavioral dataset	Deep Learning	97.80
Proposed framework	More than 45,000 alerts and 6,000 sessions	Random Forest / Gradient Boosting	98.67

Table 4: Functional comparison of ransomware detection frameworks.

Aspect	Previous Study [5]	Proposed Framework
Number of Alerts	15,427	More than 45,000
Number of Detection Models	2	10
Explainability	Not included	SHAP
Zero-Day Detection	Limited	DBSCAN + Isolation Forest
Hybrid Model	RF + DBSCAN	RF + DBSCAN as tenth model
Real-Time Alerting	Not included	Telegram
Detection Delay	Not reported	0.01-0.20 s
MITRE ATT&CK Mapping	Limited	Included
Best Accuracy	97.20%	98.67%

4.2 Unsupervised and Hybrid Learning Results

Three anomaly detection methods were tested to detect ransomware behavior that had not been seen before or was a zero-day attack (Table 2). These methods do not learn directly on labeled sessions as the supervised models do. Rather, they seek to find sessions that are highly differentiated to normal behavior. DBSCAN was the most successful of the unsupervised techniques, having been tuned to the best result. The last setup was eps = 2.5 and min samples = 5 with feature normalization by Standard Scaler. DBSCAN had an accuracy of 85.58, precision of 88.74, recall of 80.33, and F1-score of 84.32, which was much better than the results of K-Means and Isolation Forest. K-Means had an accuracy of 76.17, and Isolation Forest had 77.92. These methods were helpful in detecting anomalies, but they were not as accurate as the supervised classifiers. Moreover, a hybrid Random Forest + DBSCAN model was tested. The hybrid model is only triggered when both Random Forest and DBSCAN label the same session as malicious. The hybrid approach achieved an accuracy of 88.00%, precision of 92.10%, recall of 83.50%, and F1-score of 87.59%. Thus, the hybrid model was an extra layer of confidence that did not substitute the other detection models.

4.3 Comparison with Previous Studies

The results obtained were compared to some of the past ransomware detection studies. The fundamental Wazuh-based research [5] combined machine learning with Wazuh and stated that the accuracy of the Random Forest was 97.2% with about 15,427 alerts gathered on 12 endpoints (Table 3). The proposed framework achieved better performance while using a larger and more diverse dataset. Moreover, the proposed framework, in contrast to the previous ones, also offers SHAP-based explainability, zero-day detection, real-time notification, and the measurement of the ransomware detection delay [22], [24].

The proposed framework also provides several practical capabilities that are not fully addressed in

the previous Wazuh-based study, as shown in Table 4.

4.4 SHAP-Based Interpretation

SHAP was used on the Random Forest model to gain a better insight into why the system categorizes a session as ransomware. The most influential features in most ransomware sessions were:

- extensions_count;
- events_per_sec;
- rename_count;
- unique_files;
- dominant_ext_ratio.

This finding is understandable since ransomware typically alters numerous files, alters extensions, and does numerous activities within a limited time. As an illustration, a single suspicious session had over 60 modified files, over 30 rename operations, and a large number of extensions changes. These features were given high importance values by SHAP, and this implies that the model used these indicators as the primary ones in coming up with the final prediction.

4.5 Real-Time Detection Delay

The proposed framework was also tested during the experiments based on the time needed to identify ransomware following the initial suspicious event. The delay in detection was calculated as the time between the first file creation or file renaming event and the time when the Telegram alert was received. The experiments showed that the framework was able to detect ransomware within 0.01-0.20 s. The alert was raised in the majority of cases when the first file creation and extension-changing operations were performed, when a significant number of files were not encrypted yet. These findings indicate that the suggested framework can be used to detect ransomware in real-time and at an early stage.

4.6 Discussion

The experiments demonstrated that high accuracy is not enough. Other models had reasonable accuracy values but still performed poorly in detecting suspicious sessions. Isolation Forest and K-Means were also helpful in detecting anomalies, but their performance was still lower than that of DBSCAN. Gradient Boosting and Random Forest yielded the most overall results since they could learn complicated relationships among the behavioral features. Random Forest was also significant as it facilitated SHAP explainability. The experiments also indicated that the integration of supervised learning and anomaly detection enhanced the strength of the framework. The supervised models worked well with known ransomware behavior, whereas DBSCAN and Isolation Forest enhanced the capability to detect previously unknown attacks. The hybrid Random Forest + DBSCAN model was not used as a replacement for the other models. Rather, it served as a tenth extra model that boosted confidence when both Random Forest and DBSCAN concurred on suspicious activity. Lastly, the Telegram notification system was handy in testing. The analyst was immediately informed of the confidence score, triggered models, suspicious files, and the most influential features whenever the ransomware behavior was detected. This minimized the time taken to detect the attack and enhanced the practical usefulness of the framework.

5 CONCLUSIONS

This study presented a real-time and explainable ransomware detection framework based on behavioral telemetry collected from Windows and Ubuntu endpoints. The proposed framework integrates Sysmon, Wazuh, machine learning, anomaly detection, SHAP explainability, MITRE ATT&CK mapping, and real-time Telegram notifications into a unified detection pipeline [25]. Unlike conventional signature-based approaches, the framework relies on behavioral indicators to detect ransomware, enabling the identification of previously unseen attacks.

More than 45,000 security alerts collected over five months were processed and grouped into approximately 6,000 behavioral sessions. Based on these sessions, 15 behavioral features were extracted from 52 ransomware behaviors identified in the MITRE ATT&CK framework, Atomic Red Team, Microsoft threat intelligence reports, and previous studies. Among the evaluated classifiers, Random Forest and Gradient Boosting achieved the highest detection performance, reaching an accuracy of 98.67%. Gaussian Naive Bayes, Logistic Regression, and DBSCAN achieved a maximum accuracy of 85.58% after parameter optimization. In addition, the proposed hybrid Random Forest–DBSCAN model achieved an accuracy of 88.00% while providing greater confidence when both supervised and anomaly detection components agreed on suspicious behavior.

The proposed framework also demonstrated effective early-stage ransomware detection. Malicious activity was identified within 0.01–0.20 s

after execution, frequently during the initial file creation and file renaming stages before large-scale file encryption occurred. Following detection, the system automatically generated Telegram notifications containing the confidence score, triggered detection models, suspicious files, detection delay, and the corresponding MITRE ATT&CK techniques. Furthermore, the integration of SHAP with the Random Forest classifier improved the interpretability of the detection process by identifying the most influential features contributing to each prediction, thereby increasing analyst confidence and facilitating security investigations.

Overall, the experimental results demonstrate that the proposed framework provides an accurate, interpretable, and efficient solution for real-time ransomware detection and represents a promising approach for strengthening endpoint protection in modern computing environments.

6 FUTURE WORK

Although the proposed framework achieved promising results, several limitations remain. The experimental evaluation was primarily conducted in a controlled environment and depended on the availability of Sysmon and Wazuh telemetry. In addition, the unsupervised detection models exhibited lower performance than the supervised classifiers, indicating that further improvements are needed for reliable zero-day ransomware detection.

Future research will investigate the application of deep learning techniques and compare their performance with the machine learning models evaluated in this study. The framework will also be validated using larger and more diverse datasets collected from enterprise environments and additional operating systems to further assess its robustness and generalization capability. Finally, integrating automated incident-response mechanisms, such as endpoint isolation, malicious process termination, and automatic generation of mitigation rules following ransomware detection, represents a promising direction for enhancing the practical deployment of the proposed framework [26], [27].

REFERENCES

[1] A. Al-rikabi, M. A. Mohammed, and H. M. Salih, “GPT-4 to Mitigate Ransomware: A Survey of Explainable and Intelligent Detection Techniques,” 2025.

[2] H. Ibrahim, K. Hassan, and M. Abdullah, “Towards an Explainable Deep Learning Framework for Ransomware Detection,” 2024.

[3] H. Al-Samarraie, S. Ahmed, and L. Chen, “RansoGuard: A Real-Time Behavioral Framework for Ransomware Detection,” 2025.

[4] Y. Zhang, R. Chen, and T. Lee, “Early Ransomware Detection Using Behavioral Features,” 2024.

[5] S. A. Chamkar, M. Zaydi, Y. Maleh, and N. Gherabi, “Improving Threat Detection in Wazuh Using Machine Learning Techniques,” Journal of Cybersecurity and Privacy, vol. 5, no. 2, p. 34, 2025.

[6] A. Ibrahim and M. Kareem, “A Comprehensive Review of Ransomware Detection Techniques,” 2025.

[7] M. Roberts and P. Singh, “Decrypting Files Encrypted by Modern Ransomware: Challenges and Opportunities,” 2025.

[8] R. Khan and M. Lewis, “Double Extortion Ransomware: Analysis and Defense Strategies,” 2025.

[9] N. Mahmood and A. Tariq, “Literature Review of Machine Learning-Based Ransomware Detection,” 2021.

[10] F. Ahmed and B. Saleh, “Mitigating Ransomware Attacks Using Machine Learning,” 2025.

[11] K. Lee and M. Ahmed, “An Intelligent Ransomware Attack Detection Framework,” 2025.

[12] J. Peterson and H. Wang, “API Call-Based Detection of Ransomware Behavior,” 2025.

[13] S. Kumar and R. Patel, “Vision Transformer Technique for Behavioral Ransomware Detection,” 2025.

[14] M. Kareem and S. Ahmed, “A Novel Technique for Ransomware Detection Based on Behavioral Telemetry,” 2025.

[15] J. Smith and K. Zhao, “Generative AI for Intelligent Malware and Ransomware Mitigation Systems,” 2025.

[16] L. Chen and P. Kumar, “Mitigating Zero-Day Ransomware Using Behavioral Analysis,” 2025.

[17] R. Gupta and M. Ibrahim, “A Zero-Day Ransomware Detection Framework Based on Anomaly Analysis,” 2025.

[18] M. Ali and S. Hussein, “Explainable Deep Learning-Based Ransomware Detection Using Dynamic Behavioral Features,” 2025.

[19] A. Ahmed, T. Hasan, F. A. Abdullatif, M. S. T., and M. S. M. Rahim, “A Digital Signature System Based on Real Time Face Recognition,” in 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 2019, pp. 298-302, [Online]. Available: https://doi.org/10.1109/ICSEngT.2019.8906410.

[20] A. Alhawi, J. Baldwin, and A. Dehghantanha, “Leveraging Machine Learning Techniques for Ransomware Detection,” Future Generation Computer Systems, vol. 87, pp. 197-212, 2018.

[21] T. Wilson and P. Zhao, “A Swarm Intelligence-Enhanced Ransomware Detection Model,” 2026.

[22] M. Hassan, T. Nguyen, and J. Park, “Ransomware-as-a-Service: Recent Trends and Detection Challenges,” 2025.

[23] S. Ali, M. Rahman, and J. Kim, “Enhanced Ransomware Attack Detection Using Machine Learning,” 2025.

[24] M. Al-Hadithi and A. Kareem, “A Behavioral and Explainable Real-Time Framework for Ransomware Detection Using System Telemetry,” unpublished manuscript, 2026.

[25] D. Kareem and F. Hasan, “Early Detection and Defense Countermeasure Inference for Ransomware,” 2025.

[26] R. Ahmed and L. Saeed, “Ransomware Detection Using Deep Learning,” 2025.

[27] R. Vinayakumar et al., “Deep Learning Approach for Intelligent Intrusion Detection System,” IEEE Access, vol. 7, pp. 41525-41550, 2019.