information

MDPI

Article

Real-Time Detection and Recovery Method Against Ransomware Based on Simple Format Analysis

JaeYeol Kim

Department of Software, Kyungwoon University, Gumi 39160, Republic of Korea; pento@ikw.ac.kr

Abstract

Ransomware encrypts targeted files, making recovery difficult using conventional disinfection or deletion methods, unlike other types of malware. In particular, ransomware commonly encrypts important documents as a follow-up action, and existing antivirus programs are fundamentally incapable of preventing them. In this study, we analyzed 97 real-world ransomware behaviors and found that 95.88% of them involved encryption attempts. Consequently, we propose a real-time method for determining whether critical files have been compromised through encryption and for recovering them accordingly. The proposed Simple Format Analysis (SFA) detection technique consists of three methods: Simple Format Analysis–Fixed-structure-based (SFA-F), which analyzes the file format; Simple Format Analysis–Header-based (SFA-H), which focuses on file header information; and Simple Format Analysis—Fixed-structure-and-Header-based (SFA-F-H), a hybrid method that combines both. These techniques achieved detection accuracies ranging from 95.0% (SFA-F) to 97.9% (SFA-F-H), outperforming existing detection approaches. In addition, we introduce a novel real-time recovery approach known as real-time file restoration from damage, which integrates SFA detection with pre-input/output monitoring. We expect the proposed approach to significantly contribute to ransomware mitigation in cybersecurity environments.

Keywords: ransomware; real-time recovery; encryption; antivirus; behavioral analysis; file restoration; backup-free defense

check for updates

Academic Editor: Chingnung Yang

Received: 17 June 2025
Revised: 31 July 2025
Accepted: 22 August 2025
Published: 27 August 2025

Citation: Kim, J. Real-Time Detection and Recovery Method Against Ransomware Based on Simple Format Analysis. *Information* **2025**, *16*, 739. https://doi.org/10.3390/info16090739

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

Copyright: © 2025 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid advancement in network technologies, the malware volume has increased significantly. Globally, ransomware continues to exhibit an upward trend, with 5414 reported attacks as of 2024 [1]. According to Sophos, a UK-based cybersecurity company, approximately 59% of the large enterprises worldwide have experienced ransomware attacks, with an average ransom payment of 2 million USD [2].

In addition, ransomware primarily targets the Microsoft Windows operating system. This is because Windows remains the most widely used platform in both enterprise and personal environments, making it a primary target for ransomware [2,3]. Considering this, the experiments and analyses in this study were also conducted based on the Microsoft Windows operating system. Ransomware infections typically occur in two stages. The first is system infection, and the second involves the encryption or deletion of important files through access. The secondary damage phase, particularly file encryption, often results in nearly unrecoverable data or substantial financial losses. Therefore, countermeasures against ransomware must go beyond detection to enable real-time recovery when secondary damage occurs.

Previous ransomware detection studies have primarily focused on post-infection techniques such as signature-based malware detection [4]. Detection methods that use decoy files to identify directory traversals have also been proposed [5]. However, these post-infection approaches are limited in their ability to prevent file encryption or deletion caused by newly emerging ransomware. First determining whether encryption occurs in real time is essential to enable real-time recovery. Shannon proposed an entropy-based method to determine encryption using information theory [6], and more recent approaches analyze the dynamic API calls of ransomware to assess encryption behavior [7]. In terms of recovery, existing studies have included full document file backup strategies [8], along with techniques that monitor file input/output (I/O) behaviors to detect signs of damage [6]. Recently, the FFRecovery system was introduced, which extracts the original data blocks by leveraging the out-of-place update nature of flash memory [9].

Although most previous studies have focused on detection, this study emphasizes real-time recovery. For example, commonly used techniques include monitoring changes in file entropy, observing file system modifications, or analyzing execution behaviors. However, these approaches generally rely on post-event analysis and often require external backup systems, which limit their effectiveness against newly emerging ransomware threats.

In contrast, this study proposes an approach that enables recovery before damage occurs, emphasizing the importance of real-time intervention to restore files prior to being encrypted or overwritten. Specifically, we propose for the first time a real-time file restoration from damage (RFRD) method that monitors IRP_MJ_WRITE events occurring at the Windows kernel level before file overwrites. It uses simple format analysis (SFA) to assess potential damage and recover the target file before it is overwritten. The core of RFRD is the SFA algorithm, which analyzes the unique format and header structures of document files.

RFRD works in conjunction with the SFA algorithm, which analyzes the unique format and header structures of document files to detect structural anomalies and restore the original file before damage occurs. Unlike conventional approaches, the proposed method enables detection at the pre-encryption stage, allows recovery without the need for external backups, and is lightweight enough to be deployed at the kernel level. These characteristics make it particularly suitable for real-time response in resource-constrained systems or enterprise environments.

Experimental results show that SFA outperforms existing detection methods by 25.4–31.4% and significantly contributes to future real-time recovery technologies. In terms of processing performance, the CPU processing time was found to be between 1.0 ms and no more than 1.94 ms, and the average CPU usage ranged from 0.6% to 1.46%, indicating minimal impact on overall system performance.

The key contributions of this study can be summarized as follows:

- Real-time Ransomware Recovery: The method developed in this study introduces a novel approach that enables the immediate recovery of compromised files, even after a ransomware attack has completed its encryption process. Although conventional ransomware defense strategies emphasize periodic backups of critical files, RFRD provides an innovative solution that eliminates the need for scheduled backups by enabling real-time recovery.
- Preemptive Detection of Ransomware Damage: This study focuses on the preemptive detection of secondary ransomware behaviors, such as file encryption or deletion. The proposed SFA technique outperformed conventional detection methods by 25.4–31.4% in comparative experiments, demonstrating its superior detection performance.
- Response to Emerging Ransomware: Traditional antivirus software struggles to provide immediate protection against newly emerging ransomware. In contrast, the RFRD method holds significant value as it enables the real-time restoration of critical docu-

ments even before antivirus definitions are updated, thus offering proactive protection against novel threats.

The remainder of this paper is organized as follows: Section 2 reviews the related studies on ransomware detection and recovery. Section 3 presents the technical background of this study. Section 4 details the proposed dataset and modeling framework used in this study. Section 5 describes the experimental environment and presents results based on real-world ransomware samples. Finally, Section 6 discusses the limitations of the proposed approach and concludes this study.

2. Related Studies

2.1. Ransomware Detection

Antivirus solutions now include ransomware detection as a key feature to effectively identify known threats. Malware detection approaches are typically classified into static and dynamic analyses, with signature-based detection representing the key static technique. This method involves the extraction of key values from files using unique algorithms. However, its main limitation is the massive volume of new malware generated daily, rendering manual processing impractical.

The current capacity of analysts to manage signature analyses is overwhelming. According to the antivirus test report, over 450,000 to 600,000 new malware variants are reported each day, requiring thousands of analysts for real-time responses [4]. Thus, signature-based detection has a limited ability to address newly emerging ransomware threats and functions primarily as a post-incident response mechanism.

The decoy detection method deceives malware by exploiting common ransomware characteristics. Widely adopted in antivirus programs, it enhances detection rates with relatively minimal effort. This approach deploys decoy files at key directory locations, exploiting the tendency of ransomware to encrypt multiple files while traversing directories. When a ransomware process accesses a decoy during encryption, the activity is flagged and blocked, thereby preventing further damage. However, the detection rates may vary significantly depending on the initial execution path of the ransomware. One study applying the Beacon Code on MS Word documents reported 80% detection accuracy [5].

The I/O-based detection stems from the input/output activities of the monitoring file system. This method statistically analyzes the time required for file access requests by processes specific to a kernel. A recent study indicated that if one process accesses more than five files out of ten events within 0.01 s, it is likely to be ransomware [6]. Behavior-based detection automatically analyzes the behavioral patterns of malware to distinguish between malicious and benign activities. Gayathri proposed a method using support vector machines (SVMs) to detect ransomware based on its distinct behavioral characteristics [10].

2.2. Detection of Ransomware Tampering

Entropy-based detection is a widely adopted statistical method for identifying file encryptions. This technique is based on Shannon's entropy formula, which quantifies information content to determine whether a file has been encrypted [11]. Entropy-based detection has been used in various malware analysis methods. Kharraz applied an entropy analysis to ransomware detection by leveraging kernel-mode monitoring in Windows. They analyzed file I/O data to detect the presence of encryption and demonstrated improved detection performance for previously unknown ransomware. This study reported a detection rate of 72.2% for ransomware samples that had not yet been identified using existing antivirus programs [6].

Zakaria introduced the RENTAKA framework that monitors API calls and dynamically analyzes ransomware behavior in sandbox environments to identify encryption

attempts. This approach trains machine learning (ML) models on pre-encryption API patterns, achieving an accuracy of 97.05% with an SVM model [7]. Cache-assisted ransomware detection and recovery (CARDR) in SSDs, a recovery framework utilizing the SSD DRAM cache, analyzes I/O access patterns stored in the cache. It detects encryption patterns via repeated file overwriting using an SVM model, yielding a detection accuracy of 96.3% [12].

2.3. Ransomware Damage Recovery

Traditional recovery approaches rely on pre-backup, which is the most basic method for protecting important documents by backing them up beforehand. This method requires ample storage capacity and fails to recover files if an infection occurs before the next backup. Jadon proposed a method that periodically backs up data and uses convolutional neural network (CNN) and BiLSTM models to restore files in real time upon ransomware detection, achieving a recovery rate of 97.5% [13].

Recently, hardware-level recovery strategies have garnered attention [13]. FFRecovery leverages the out-of-place update characteristics of SSDs in which overwritten data are stored in a new location without immediate deletion. This allows the retrieval of the original content from the prior data block, achieving a 95.2% recovery rate for WannaCry [9]. CARDR enables real-time recovery of physical blocks using mapping information from the flash translation layer of the SSD [12].

2.4. Distinctions of Our Study

This section presents the unique aspects of the proposed RFRD method compared with existing solutions:

- Ransomware Detection Approach: Conventional detection mechanisms rely on post-discovery updates of signatures or behavioral rules, which are insufficient for preventing file modifications caused by newly emerging ransomware variants. Similarly, decoy-based methods cannot protect files until the decoy is activated by the ransomware. In contrast, the RFRD method shifts focus from traditional detection to real-time recovery of compromised files, employing the SFA technique to mitigate the damage as it occurs.
- File Modification Detection: Traditional entropy-based approaches apply statistical models and achieve a detection rate of 72.2% [6]. The RENTAKA framework uses ML to analyze API behavior with an accuracy of 97.5%. FFRecovery and CARDR rely on ML models that analyze SSD cache data. The proposed SFA technique is a real-time detection method based on file structure analysis. In extensive evaluation tests, it demonstrated an accuracy range of 95.0% to 97.9%, depending on file type, encryption method, and detection mode.
- Ransomware Recovery Method: Pre-backup techniques cannot recover data lost before the next scheduled backup. ML-based restoration methods, such as Jadon's, do not fully ensure data integrity. FFRecovery and CARDR depend on the assumption that SSD cache content remains intact. In contrast, the proposed RFRD method incorporates an integrated SFA detection engine, meaning its recovery performance is closely linked to the accuracy of the SFA algorithm. Experimental results demonstrated that, based on the high detection rate of SFA, the RFRD method achieved high recovery accuracy as well.

3. Background

This section provides the necessary background for understanding the proposed method, with a particular focus on the characteristics of ransomware and mechanisms for monitoring I/O operations in a Windows environment.

3.1. Ransomware

Ransomware is a type of malicious software that encrypts user files or blocks system access, demanding a ransom in exchange for restoration. Ransomware attacks typically occur in two stages. The first stage involves system infiltration through phishing emails, software vulnerabilities, or malicious downloads. The second stage causes secondary damage by encrypting or deleting critical files [2]. Among these, file encryption leads to severe and irreversible data loss. Traditional antivirus solutions are unable to cope with polymorphic or previously unknown (zero-day) ransomware variants [6]. Owing to the inherent structure of such attacks, a fundamental limitation known as detection latency persists, which refers to the time gap between the onset of a malicious behavior and its detection [7].

Consequently, recent approaches have proposed proactive defense mechanisms, including real-time file monitoring, dynamic backup creation, and kernel-level event tracking (e.g., Windows IRP_MJ_WRITE and IRP_MJ_CREATE events) [14]. These methods aim to minimize damage by detecting and responding to ransomware activity before file modification is completed.

3.2. Windows Kernel I/O Monitoring

In the Windows operating system, file access and modification operations are managed at the kernel level using I/O request packets (IRPs). These IRPs are handled by the I/O manager and forwarded to the device drivers using major functional codes, such as IRP_MJ_CREATE, IRP_MJ_READ, and IRP_MJ_WRITE. In particular, IRP_MJ_CREATE is triggered when a file is created or opened, whereas IRP_MJ_WRITE is triggered when data are written to a file or device [15].

Such kernel-level events provide critical opportunities for early detection of ransomware behavior. For instance, monitoring the IRP_MJ_CREATE event allows the interception of file access prior to modification, enabling proactive actions such as backup creation or access denial. In contrast, the IRP_MJ_WRITE event can be used to track actual write operations, thereby identifying potential encryption or unauthorized data tampering attempts.

The use of IRP_MJ_CREATE and IRP_MJ_WRITE for kernel-level monitoring, which is integrated into the proposed RFRD mechanism for ransomware detection and file recovery, is illustrated in Appendix A.1.

4. Materials and Methods

This section introduces the proposed RFRD method and presents comparative experimental results with existing approaches.

4.1. Dataset

Figure 1 illustrates the dataset used in our experiments, which comprises 97 real-world ransomware samples collected by a major Korean cybersecurity company between 2014 and 2016. These ransomware samples were categorized into ten distinct file behavior types, as summarized in Table 1. Table 1 summarizes a behavioral model based on the analysis of ransomware activity data. The model categorizes behaviors into four main categories—file, system, process, and network—and further classifies them into ten specific behavior types. Among these ten types, all are recoverable, except for F1 (encryption) and F3 (deletion), which result in irreversible damage.

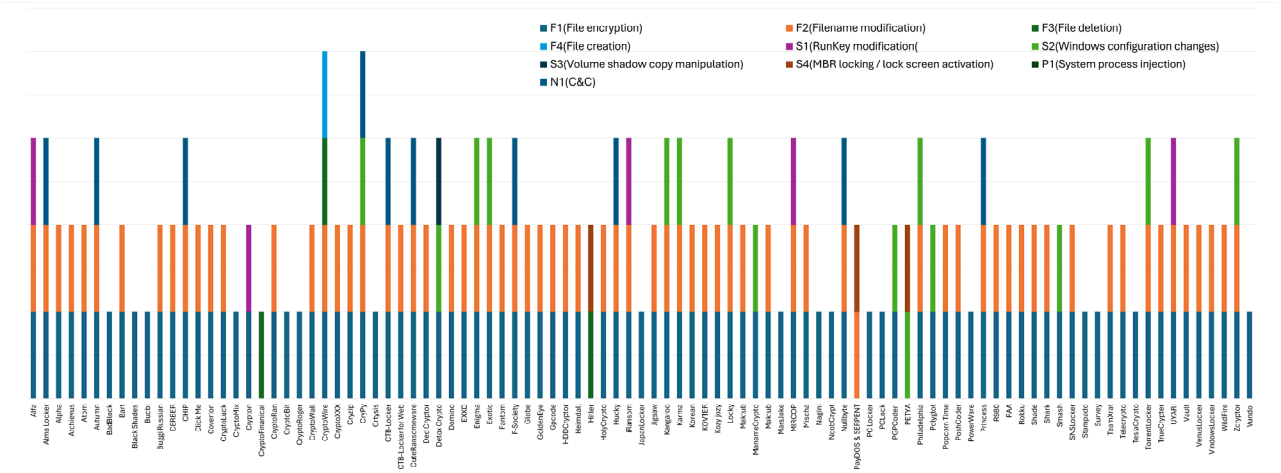

Figure 1. Classification of behavioral actions exhibited by ransomware families. Each dot indicates the presence of a specific behavior (F1–N1) performed by an individual ransomware family. This figure highlights the distribution and frequency of common and unique behavioral traits across 97 ransomware types.

Table 1. Modeling the distribution of ransomware samples across each behavior type (Figure 1).

Category	Behavior ID	Behavior Type	Distribution (%)
File (F)	F1	File encryption	* 95.88
	F2	Filename modification	72.61
	F3	File deletion	3.09
	F4	File creation	1.03
System (S)	S1	RunKey modification	5.15
	S2	Windows configuration changes	15.46
	S3	Volume shadow copy manipulation	1.03
	S4	MBR locking/lock screen activation	3.09
Process (P)	P1	System process injection	0.00
Network (N)	N1	Connection to command and control (C & C) server	9.70

(*) denotes the highest score.

Based on the classification results in Table 1, behavior type F1, which corresponds to ransomware file encryption, accounts for 95.88% of the observed cases. Therefore, this study focuses on the detection of F1 (irreversible file modification through encryption) and F3 (file deletion), both of which are considered unrecoverable.

4.2. Ransomware Distribution Analysis

This section provides an in-depth analysis of the ransomware behavior distributions presented in Table 1. Behavior IDs represent the distinct actions performed by each of the 97 ransomware samples. This study focuses on identifying behavioral IDs that are considered irrecoverable once executed. Specifically, we emphasize two critical behaviors: F1 (file encryption) and F3 (file deletion).

Among the analyzed samples, F1 was observed in 95.88% of the ransomware families, indicating that file encryption was the most prevalent and damaging behavior. We further examined the exception—behavior types that did not exhibit F1 behavior. These are summarized in Table 2, which lists the four ransomware families (4.12%) that did not perform file encryption during infection. Through this analysis, we identified F1 and F3 as the two primary behaviors responsible for the irreversible damage. This finding provides a

solid rationale for designing the proposed RFRD method, which specifically targets these critical actions.

Table 2. Behavioral analysis of non-encrypting ransomware.

Ransomware	Behavior ID	Behavior Type	Recovery Possibility	Description
CryptoFinancial	F3	File deletion	Not recoverable	Deceives the user by deleting files and presenting them as encryption.
Hitler	F3	File deletion	Not recoverable	Deceives the user by deleting files and presenting them as encryption.
PayDos and SERPENT	F2	Filename modification	Recoverable	Changes only one character in the file extension
Petya	S2	Windows configuration changes	Data can be recovered	Encrypts the master boot record (MBR) boot sector

4.3. Real-Time File Restoration from Damage (RFRD) Method

This section presents the overall workflow and detailed algorithm of the proposed RFRD method, which enables real-time detection and recovery from ransomware-induced damage (Figure 2).

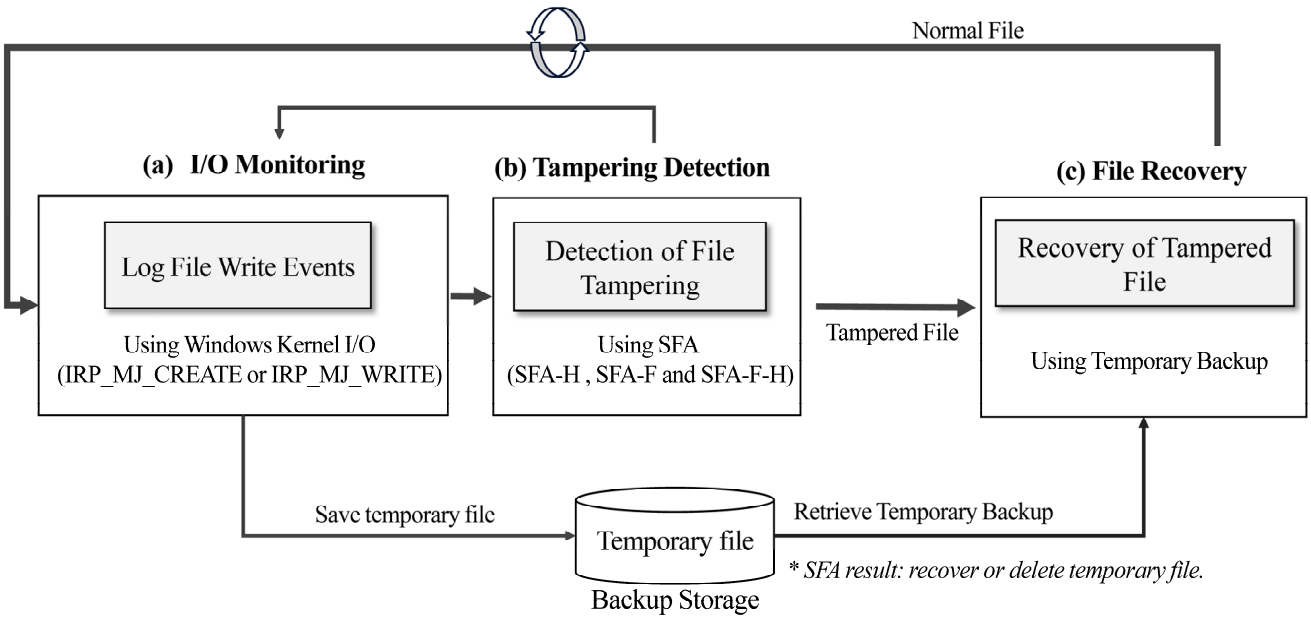

Figure 2. RFRD workflow of the proposed ransomware detection and file recovery mechanism. The system comprises (**a**) logging file write I/O operations using IRP_MJ_WRITE, (**b**) detecting file tampering via SFA, and (**c**) recovering tampered files using temporary backup copies saved prior to the modification. The asterisk (*) indicates that, based on the SFA result, the system either recovers the tampered file or deletes the temporary file.

The proposed RFRD method comprises three main stages. (a) I/O Monitoring: This stage monitors file I/O operations on the Windows system when ransomware activity is detected. (b) Detection of file tampering: In this phase, our custom-developed SFA is applied to identify files that have been tampered with as a result of ransomware behavior. (c) Recovery of tampered files: Finally, compromised files are restored using temporary backup copies created prior to modification. The following sections describe the algorithms for each step of the RFRD procedure.

4.3.1. I/O Monitoring

At this stage, real-time monitoring is applied to the target files affected by the ransomware. When a file I/O operation involving predefined document-type extensions is detected, a temporary backup is created immediately to preserve the original content.

The Windows operating system allows the detection of file creation events via the IRP_MJ_CREATE signal, which is triggered in the kernel prior to the completion of write operations [14]. Based on this observation, the proposed method initiates a temporary backup at the point of file creation or access. Furthermore, temporary backups are created only when kernel-level IRP_MJ_WRITE events occur, specifically during file creation, deletion, or modification. The monitoring process was selectively applied to files with predefined extensions, such as .docx, .pdf, and .xlsx, to address the potential performance degradation caused by excessive I/O logging. Appendix A.1 presents the implementation details of this mechanism.

4.3.2. Detection of File Tampering

This phase determines whether a file targeted by the ransomware has been encrypted. The core algorithm in this phase employs the SFA method to assess the structural integrity of the file format.

SFA includes two techniques: SFA-H and SFA-F. In addition, a hybrid approach, SFA-F-H, which combines the strengths of both methods, is also proposed. SFA-H identifies the file type by comparing predefined signature patterns located at the beginning of the file. If the file header is encrypted, rendering the signature unrecognizable, the file is classified as compromised. In contrast, SFA-F addresses the limitations of SFA-H by analyzing the structural format of the entire file, even if only the middle section has been encrypted, while preserving the header. This enables accurate detection, even when partial encryption techniques are used. However, this method requires deeper analysis of the internal document format. In addition, SFA-F-H is a hybrid approach that combines the strengths of both SFA-H and SFA-F, providing high detection performance capable of handling various types of encryption across different document formats and file extensions.

As shown in Figure 3, file types such as .docx, .pptx, and .xlsx used in MS Office share a common header format. Thus, header validation provides a reliable means of distinguishing between normal and tampered files. A typical validation method is the CRC32 checksum verification embedded in the header. However, this requires the computation of a full file hash, which is time-consuming and unsuitable for real-time detection. Consequently, we propose a lightweight signature validation technique.

MS Office file headers include three mandatory signature offsets. The SFA-F detection process verifies the presence and correctness of these offsets, as summarized in Table 3. Specifically:

- This confirms the presence of the three predefined signature offsets. If any are missing, the file is classified as compromised.
- The reader scans the last 64 KB of the file for the end of central directory (EOCD) signature (0x06054B50) to locate the EOCD. It then reads the 4-byte offset field at bytes 16–19 of the EOCD, which points to the start of the central directory. If the data at that offset do not begin with the central directory entry signature (0x02014B50), the file may be flagged as having been tampered with.

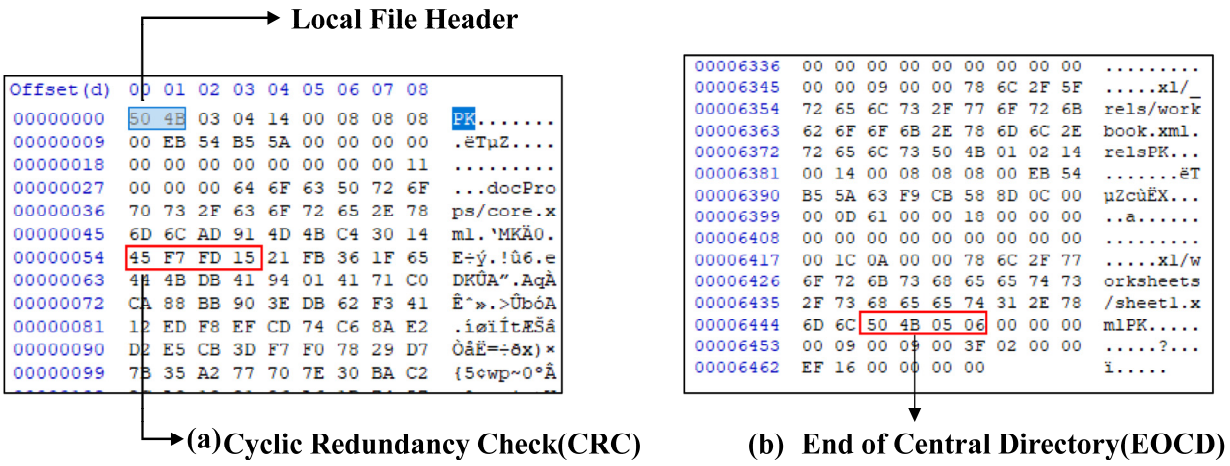

Figure 3. Structural layout of a typical Microsoft Office file, including required document-based header segments used for signature validation in the SFA method: (**a**) verification of structural integrity using the CRC field; (**b**) analysis of structural completeness through the End of Central Directory (EOCD) structure. The blue text represents the offset, the blue highlighting indicates header characters, and the red boxes denote the CRC and EOCD fields.

Table 3. MS Office header signature offsets.

Header Format	Size (Bytes)	Signature Offset	Description
Local file header	2	0x04034b50 (50 4B 03 04)	Basic file information
Central directory (CD)	4	0x02014B50 (50 4B 01 02)	Metadata structure of the file
EOCD	4	0x06054B50 (50 4B 05 06)	Pointer to the central directory start

4.3.3. Recovery of Tampered File

At this stage, the system leverages the analysis results obtained from phase (b) in Figure 1 to determine whether a file has been compromised by ransomware-induced encryption or deletion. If the file is identified as being tampered with, it is immediately restored using the temporary backup created during Phase (a). This process enables the proposed RFRD method to perform real-time file recovery, thereby minimizing data loss and ensuring system continuity, even under active ransomware attacks.

5. Performance Evaluation

5.1. Performance Evaluation of Ransomware Detection Using Field-Collected Real-World Data

5.1.1. Hardware and Software Environment

The experimental environment was configured using Microsoft Windows 7 x64 Service Pack 1 as the operating system. The hardware specifications included an Intel Core i5 processor running at 2.4 GHz and 1 GB of RAM.

5.1.2. Test Dataset

To evaluate the effectiveness of the proposed method, six real-world ransomware families—Cerber, TeselaCrypto, SegaCrypto, Locky, Ransom32, and CTBLocker—were used, as summarized in Table 4. Each ransomware variant was selected based on its unique encryption behavior and propagation method, providing a representative range of attack patterns.

Table 4. Description of ransomware samples used for test.

Ransomware	Behavior Description
Cerber	Changes file extensions to .hta or random strings during encryption. Skips 512 or 640 bytes at the beginning before encrypting.
TeselaCrypto	Distributed via vulnerable web pages and email attachments. Changes extensions to .ecc and .micr.
SegaCrypto	Delivered through emails or Word documents. Encrypts files and appends the .sega extension.
Locky	Renames files with the .locky extension and encrypts using RSA-2014 or AES-128.
Ransom32	JavaScript-based ransomware with obfuscation techniques.

The tampering test set comprised 360 document samples distributed across the four directory structures. Table 5 lists the directory compositions used in the validation experiments.

Table 5. Directory structure and quantity of document samples used for test.

Name	Directory Path	Number of Files
Data 1	C:\\	90
Data 2	C:\\library\doc	90
Data 3	C:\\library\photo	90
Data 4	C:\\Users	90
Sum	4 Directory	360

5.1.3. Evaluation Results

This section presents the experimental results that validate the effectiveness of the proposed RFRD method under real ransomware tampering scenarios.

Comparison of Detection Performance: SFA-H vs. SFA-F

We evaluated the detection performance of two variations of the SFA technique: SFA-H (header-based) and SFA-F (form-based). The test involved executing five representative ransomware families selected from Table 4 and infecting 360 document files structured across four directories, as summarized in Table 5. Table 6 summarizes the results of the SFA-H method. The average recovery rate across all file types was 83.7%. Notably, the detection performance for files encrypted by Cerber ransomware was significantly lower, at only 2.2%, owing to the unique behavior of skipping the initial header region during encryption.

Table 6. Ransomware infection detection performance by file type using SFA-H (%).

Ransomware	DOCX	PPTX	XLSX	Average
Cerber	3.3	0.0	3.3	2.2
TeselaCrypto	100.0	100.0	100.0	100.0
SegaCrypto	100.0	100.0	100.0	100.0
Locky	100.0	100.0	100.0	100.0
Ransom32	100.0	100.0	100.0	100.0
CTBLocker	100.0	100.0	100.0	100.0
Average	83.9	83.3	83.9	83.7

Table 7 summarizes the detection performance of the SFA-F method, which outperformed that of SFA-H, with a perfect detection rate of 100% across all ransomware types and document formats. This confirms the superior reliability of the format-level structural validation.

Table 7. Ransomware infection detection performance by file type using SFA-F (%).

Ransomware	DOCX	PPTX	XLSX	Average
Cerber	100.0	100.0	100.0	100.0
TeselaCrypto	100.0	100.0	100.0	100.0
SegaCrypto	100.0	100.0	100.0	100.0
Locky	100.0	100.0	100.0	100.0
Ransom32	100.0	100.0	100.0	100.0
CTBLocker	100.0	100.0	100.0	100.0
Average	100.0	100.0	100.0	100.0

In summary, the SFA-F method demonstrated superior detection performance, exceeding that of the SFA-H method by 16.3%. However, this requires an in-depth format analysis, which may incur additional computational overhead. In contrast, SFA-H is advantageous for lightweight, real-time detection owing to its simplicity and speed.

Comparison with Other Methods

This section presents a comparative experimental analysis of the proposed SFA (SFA-H and SFA-F) method and other ransomware detection techniques, namely, the entropy-based detection [6] and decoy-based detection [5] methods. Figure 4 illustrates a performance comparison among different ransomware detection models aimed at real-time recovery. The proposed SFA-F method significantly outperformed the other methods, with detection rates exceeding them by 25–38%, demonstrating its superior capability.

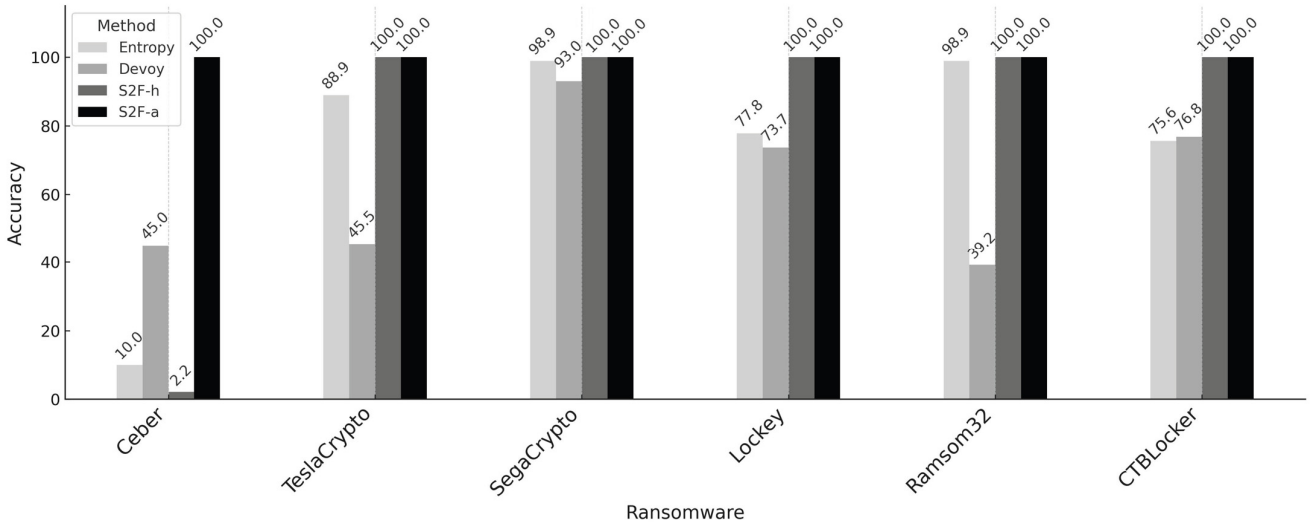

Figure 4. Comparison of detection performance among different ransomware detection models for real-time recovery.

5.2. In-Depth Analysis of Detection Scalability and Model Performance Using Publicly Available Ransomware Samples

5.2.1. Hardware and Software Environment

This study was conducted in a VMware 11 environment, with 8 GB of memory and two allocated CPU cores. The operating system used was Windows 11, and for analysis, a large language model (LLM), ChatGPT-4o (hereafter referred to as ChatGPT), was employed.

5.2.2. Extended Test Dataset

Table 8 presents five real-world ransomware samples—Cerber, Hive, Thanos, XData, and WannaCry—collected from the open-source ransomware repository theZoo [16] be-

tween 2017 and 2022. These samples were used to conduct the in-depth evaluation tests in this study.

Table 8. Description of ransomware samples used for the extended test set.

Ransomware	Behavior Description
Cerber	Change file extensions to .hta or random strings during encryption. This ransomware sample was registered on the theZoo [16] repository in 2021.
Hive	Active since mid-2021, Hive employs a double extortion strategy and a Ransomware-as-a-Service (RaaS) model. It not only encrypts files but also exfiltrates data to pressure victims.
Thanos	Thanos is a customizable RaaS-based ransomware that began circulating on the dark web in early 2020.
XData	XData rapidly spread in Ukraine in May 2017. It is known for its fast propagation, primarily exploiting SMB vulnerabilities.
WannaCry	WannaCry is a notorious worm-type ransomware that spread globally in 2017 by exploiting a vulnerability in the SMB protocol.

In addition to the DOCX, PPTX, and XLSX files used in Section 5.1, additional document types such as PDF, HWPS, ZIP, and JPG formats were included to enhance the diversity of the experimental dataset. The key characteristics of each file type are summarized in Table 9.

Table 9. Description of test file types.

File Type	Description
DOCX	Used in Section 5.1; includes an End of Central Directory (EOCD) structure.
PPTX	Used in Section 5.1; includes an End of Central Directory (EOCD) structure.
XLSX	Used in Section 5.1; includes an End of Central Directory (EOCD) structure.
PDF	Added in Section 5.2; the format analysis code is provided in Appendix A.4.
HWPS	Added in Section 5.2; includes an EOCD structure. This is a word processing format widely used in South Korea.
ZIP	Added in Section 5.2; a compressed file format that includes an EOCD structure.
jpg	Added in Section 5.2; the format analysis code is provided in Appendix A.3.

5.2.3. Experimental Method

The experiments in this section were conducted using the same procedure described in Section 5.1. However, the key difference lies in the use of a large language model (LLM), specifically ChatGPT-4o, which was queried in real-time using predefined prompt commands. The prompt used to interact with the LLM was as follows:

> *"Please analyze the file I will now provide using various methods, and summarize the results in CSV format, including the file name, file size, upload time, entropy, encryption status, and analysis time."*

Four key performance metrics were used in this experiment and are defined as follows:

- True Positive (*TP*): The model correctly predicts a sample as positive when it is positive.
- True Negative (*TN*): The model correctly predicts a sample as negative when it is negative.
- False Positive (*FP*): The model incorrectly predicts a sample as positive when it is negative.

- False Negative (*FN*): The model incorrectly predicts a sample as negative when it is positive.

The detection accuracy of our proposed method was evaluated using these four indicators. Specifically, Equation (1) defines accuracy, Equation (2) defines precision, Equation (3) defines recall, and Equation (4) defines the F1 score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$f1\ score = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

5.2.4. Extended Evaluation Results

Comparative Evaluation of Ransomware Detection Methods Using Structured File Formats

Table 10 compares the detection performance of various methods on five types of ransomware—Cerber, Hive, Thanos, XData, and WannaCry—using structured file formats such as DOCX, PPTX, XLSX, HWPX, PDF, and ZIP. These formats are commonly used in real-world office environments and are frequent targets of ransomware attacks, making them suitable for analysis. In this experiment, five detection methods were evaluated: entropy-based methods with thresholds of 5.5 and 7.5, a semantic analysis-based method using ChatGPT-4o, and two structure-based methods, namely, SFA-H and SFA-F. The evaluation metrics included precision, recall, accuracy, and F1 score.

Table 10. Comparative analysis of ransomware detection methods on Cerber, Hive, Thanos, XData, and WannaCry using document file formats (DOCX, PPTX, XLSX, HWPX, PDF, and ZIP).

Method	Ransomware	Precision	Recall	Accuracy	F1 Score	Avg. F1 Score
Entropy (5.5)	Cerber	1.000	0.530	0.530	0.690	0.646
	Hive	1.000	0.530	0.530	0.690	
	Thanos	0.600	0.400	0.320	0.480	
	XData	1.000	0.530	0.530	0.690	
	WannaCry	0.980	0.520	0.520	0.680	
Entropy (7.5)	Cerber	1.000	0.530	0.560	0.690	0.680
	Hive	0.980	0.520	0.550	0.680	
	Thanos	0.931	0.511	0.520	0.660	
	XData	0.990	0.530	0.550	0.690	
	WannaCry	0.980	0.520	0.550	0.680	
ChatGPT	Cerber	0.920	0.630	0.700	0.750	0.706
	Hive	0.980	0.650	0.730	0.780	
	Thanos	0.940	0.640	0.710	0.760	
	XData	0.820	0.610	0.650	0.700	
	WannaCry	0.560	0.510	0.520	0.540	
SFA-H	Cerber	0.170	1.000	0.590	0.290	0.858
	Hive	1.000	1.000	1.000	1.000	
	Thanos	1.000	1.000	1.000	1.000	
	XData	1.000	1.000	1.000	1.000	
	WannaCry	1.000	1.000	1.000	1.000	
SFA-F	Cerber	1.000	1.000	1.000	1.000	* 0.960
	Hive	1.000	1.000	1.000	1.000	
	Thanos	0.700	1.000	0.850	0.820	
	XData	0.960	1.000	0.980	0.980	
	WannaCry	1.000	1.000	1.000	1.000	

* denotes the highest score.

The entropy-based methods achieved generally high precision but showed relatively low recall and accuracy. In contrast, the ChatGPT-based method demonstrated more balanced performance, outperforming entropy-based techniques in both accuracy and F1 score. This indicates that semantic or structural understanding of documents contributes meaningfully to ransomware detection.

The SFA-F method demonstrated the highest overall performance, achieving precision, recall, accuracy, and F1 score values of 0.960 or higher across all ransomware types. This superior performance can be attributed to the application of structural format-based detection rules, which provided consistent and reliable results. In particular, SFA-F outperformed other methods in terms of detection stability and generalizability. In contrast, the SFA-H method showed lower F1 scores, especially for ransomware such as Cerber, which performs partial encryption. This result highlights the limitations of header-based detection when structural integrity is partially preserved and underscores the effectiveness of SFA-F in dealing with sophisticated ransomware attacks.

Performance Evaluation of SFA-F Hybrid Detection Methods for Unsupported File Extensions

This section evaluates the detection performance of SFA-F against document file types that are not natively supported. To assess the flexibility of hybrid detection models, JPG files, which are not included in the current SFA-F rule set, were selected for experimentation. This scenario simulates cases in which ransomware manipulates file extensions or targets newly emerging file formats for encryption.

To this end, three hybrid detection models were constructed and compared:

- SFA-F-H: A hybrid method combining SFA-F and SFA-H.
- Simple Format Analysis—Fixed-structure and Entropy-based (SFA-F-Entropy): A static feature-based method that calculates entropy values (with a threshold of 7.5, as shown to be effective in Table 10).
- Simple Format Analysis—Fixed-structure and GPT (SFA-F-ChatGPT): A semantic analysis approach that leverages a large language model (LLM) to interpret content patterns within the file.

Table 11 summarizes the experimental results. The SFA-F-H and SFA-F-ChatGPT models showed high recall rates, indicating their effectiveness in detecting encrypted files without omission. However, while SFA-F-Entropy (7.5) maintained high precision, it suffered from low recall, suggesting a higher rate of missed detections. These results imply that for unknown or unsupported file extensions, structure-based or semantic-based approaches may offer more robust detection capabilities.

Table 11. Performance comparison of SFA-F hybrid detection methods for file extensions, including the originally supported formats (DOCX, PPTX, XLSX, HWPX, PDF, ZIP) and the newly added unsupported format (JPG).

Method	Ransomware	Accuracy	Precision	Recall	F1 Score	Avg. F1 Score
SFA-F-H	Cerber	0.928	0.857	0.998	0.922	* 0.979
	Hive	0.999	1.000	0.998	0.999	
	Thanos	0.989	0.979	0.998	0.988	
	XData	0.989	0.979	0.998	0.988	
	WannaCry	0.999	1.000	0.998	0.999	

Table 11. *Cont.*

Method	Ransomware	Accuracy	Precision	Recall	F1 Score	Avg. F1 Score
SFA-F-Entropy (7.5)	Cerber	0.954	1.000	0.915	0.956	0.950
	Hive	0.952	0.997	0.915	0.954	
	Thanos	0.940	0.973	0.913	0.942	
	XData	0.943	0.979	0.913	0.945	
	WannaCry	0.954	1.000	0.915	0.956	
SFA-F-ChatGPT	Cerber	0.922	0.929	0.915	0.922	0.946
	Hive	0.950	0.986	0.920	0.952	
	Thanos	0.957	1.000	0.921	0.959	
	XData	0.950	0.986	0.920	0.952	
	WannaCry	0.943	0.971	0.919	0.944	

* denotes the highest score.

Performance Comparison of Detection Models in Terms of Processing Speed and Resource Utilization

As shown in Table 12, SFA-H achieved a very short average processing time of 1.0 ms and maintained a CPU usage of less than 0.6%, confirming its suitability as a lightweight method for real-time detection environments. SFA-F, while consuming slightly more resources, still operated efficiently with CPU processing times not exceeding 1.94 ms and an average CPU usage of 1.46%, demonstrating an excellent balance between detection precision and computational efficiency. In contrast, the entropy-based detection method showed a significantly higher CPU usage, reaching up to 6.9% on average, and longer CPU processing times of up to 4.6 ms, indicating a relatively higher consumption of system resources.

Table 12. Resource consumption comparison of detection methods across five ransomware types.

Method	Ransomware	File Size (KB)	Processing Time (ms)	I/O Wait Time (ms)	Backup Time (ms)	CPU (%)
SFA-F	Cerber	2324.8	2.9	0.5	23.3	1.8
	Hive	2266.8	1.6	0.5	7.2	1.4
	Thanos	1795.5	2.5	0.1	7	1.3
	XData	2266.3	1.6	0.3	3.7	1.7
	WannaCry	2324.6	1.1	0.4	3.1	1.1
	Average	2195.6	1.94	0.36	8.86	1.46
SFA-H	Cerber	2257.0	0.8	0.2	2.8	0.8
	Hive	2332.0	1.6	0.4	5.0	0.7
	Thanos	1795.5	0.3	0.1	5.4	0.3
	XData	2266.3	1.5	0.4	5.5	0.4
	WannaCry	2324.6	0.9	0.2	5.7	0.6
	Average	2195.1	1.0	0.3	4.9	0.6
Entropy (7.5)	Cerber	2324.8	5.3	0.4	3.4	7.8
	Hive	2324.8	5.3	0.4	3.4	7.8
	Thanos	1795.5	1.4	0.0	0.3	5.8
	XData	2266.3	5.0	0.3	4.3	7.3
	WannaCry	2324.6	6.0	0.5	3.4	6.0
	Average	2207.2	4.6	0.3	2.9	2.9

Across all detection methods, the I/O wait time remained below 0.4 ms, and the average backup time was under 10 ms, suggesting that the overall impact on system performance was minimal. In summary, the proposed SFA-F and SFA-H methods are confirmed to be lightweight and efficient, making them highly suitable for integration

into real-time monitoring and security systems, with minimal resource overhead and high detection performance.

6. Discussion and Limitation

This section discusses the limitations of the proposed method and outlines the directions for future research. The RFRD and SFA techniques presented in this study represent a novel approach that enables the real-time recovery of files compromised by ransomware. However, this study has several limitations that warrant further investigation.

First, although the SFA-H method offers advantages in terms of speed and simplicity, it may fail to detect cases where only parts of a document are encrypted, leaving the header intact. Consequently, we developed the SFA-F method, which analyzes the structural format of a file. Although this method demonstrated a strong detection performance, it requires prior knowledge of the document's internal structure, potentially increasing the implementation complexity.

Additionally, the proposed method yields false positives when encountering legitimate encryption mechanisms such as digital rights management. Nevertheless, we believe that this issue can be mitigated through application-level approaches, such as implementing a whitelist to allow trusted encryption processes.

The SFA detection method relies on static file structure verification, but false positives may occur when legitimate processes temporarily exhibit abnormal behaviors. To address this issue, integrating process-level behavioral analysis techniques, such as behavior scoring and whitelisting of trusted applications, could further improve the accuracy and reliability of the detection system. Advanced evasion techniques used by modern ransomware, such as delayed execution and fragmented encryption, also present significant limitations. While the SFA method is capable of detecting structural anomalies, its detection performance may vary depending on the granularity of I/O monitoring. Therefore, future research will focus on developing techniques that can automatically adjust the scope and frequency of I/O monitoring by analyzing system behavior patterns, thereby providing a more effective response to evasive ransomware variants.

Finally, while the proposed method emphasizes kernel-level detection and recovery mechanisms, it currently lacks a user interface or dedicated logging system. These components play a crucial role in enhancing system transparency and ensuring traceability in the event of an incident, which is particularly important in enterprise environments. As a result, future study will explore the development of visualization modules or logging systems to enable verification and auditing of ransomware-related events.

7. Conclusions

This study proposed a novel method, RFRD, to address the limitations of conventional ransomware defense mechanisms by enabling real-time detection and recovery of encrypted files. Unlike traditional approaches that rely on prescheduled backups or post-infection signature analysis, the RFRD system operates proactively at the kernel level by monitoring I/O events and leveraging an SFA to assess file integrity prior to modification.

The SFA technique was introduced in two forms: SFA-H and SFA-F. SFA-H allows lightweight detection based on header validation, whereas SFA-F offers a robust structural analysis for scenarios involving partial encryption.

Through experiments involving 97 real-world ransomware samples, SFA-F achieved an average F1 score of 0.960, outperforming entropy-based methods (average F1 score: ~0.55–0.65) and ChatGPT-based analysis (average F1 score: ~0.71). To evaluate detection flexibility for unsupported extensions such as JPG, hybrid models (SFA-F-H, SFA-F-GPT, and SFA-F-Entropy) were constructed. Among these, SFA-F-H demonstrated the highest

average F1 score of 0.979, confirming its strong adaptability to diverse file types. In terms of performance, SFA-F and SFA-H exhibited excellent resource efficiency, with average CPU usage of 1.46% and 0.6%, and processing times under 1.94 ms, validating their suitability for integration into real-time security systems.

In summary, this study contributes significantly to the field of ransomware defense by offering a real-time, backup-free recovery method that complements existing detection strategies and addresses the increasing threat of advanced ransomware variants.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A.

Appendix A.1. Windows Kernel I/O Event Hooking Using IRP

This appendix presents a simplified implementation of IRP event hooking in a Windows kernel-mode driver. The code demonstrates how IRP_MJ_CREATE and IRP_MJ_WRITE events can be monitored to trigger temporary backups of sensitive documents before modification occurs (Algorithm A1).

Algorithm A1: Windows Kernel I/O Processing Using IRP

```
NTSTATUS DriverDispatch(
PDEVICE_OBJECT DeviceObject,
PIRP Irp
)
{
PIO_STACK_LOCATION irpSp = IoGetCurrentIrpStackLocation(Irp);
NTSTATUS status = STATUS_SUCCESS;
switch (irpSp->MajorFunction) {
case IRP_MJ_CREATE:
// Backup the file or log its access path
BackupFileIfNeeded(irpSp->FileObject->FileName);
break;

case IRP_MJ_WRITE:
// Create a temporary backup before modifying the file
SaveTemporaryCopy(irpSp->FileObject);
break;
default:
break;
}
Irp->IoStatus.Status = status;
Irp->IoStatus.Information = 0;
IoCompleteRequest(Irp, IO_NO_INCREMENT);
return status;
}
```

Appendix A.2. Format-Based Structural Validation Using EOCD and CD Entry

This appendix presents the SFA-F method, which checks the integrity of a document file by locating the End of Central Directory (EOCD) record and verifying the presence of the Central Directory (CD) Entry (Algorithm A2).

Algorithm A2: Format-Based Structural Validation for Encrypted File Detection (SFA-F)

```
#define DEF_EOCD_SIG "\x50\x4B\x05\x06"     // EOCD signature: PK\x05\x06
#define DEF_CD_SIG    "\x50\x4B\x01\x02"    // Central Directory signature: PK\x01\x02
#define DEF_EOCD_SIZE 22
#define DEF_MAX_SCAN_SIZE 65557             // 64KB + EOCD_SIZE

int SFA_Format(const char *ppath) {
FILE *fp = fopen(ppath, "rb");
if (!fp) {
printf("Cannot open file.\n");
return −1;
}
// (1) Get total file size
fseek(fp, 0, SEEK_END);
long n_file_size = ftell(fp);

// (2) Limit the scanning size to the last 64KB
long n_skn_size = (n_file_size < DEF_MAX_SCAN_SIZE) ? n_file_size: MAX_SCAN_SIZE;
fseek(fp, −n_skn_size, SEEK_END);

// (3) Read the last scan size bytes
unsigned char *szbuffer = (unsigned char *)malloc(n_skn_size);
if (!szbuffer)
{
printf("Memory allocation failed.\n");
fclose(fp);
return −1;
}
fread(szbuffer, 1, n_skn_size, fp);

// (4) Search EOCD signature (0x06054B50) from the end
long n_eocd_index = −1;
for (long i = n_skn_size − EDF_EOCD_SIZE; i >= 0; i--)
{
if (memcmp(szbuffer + i, DEF_EOCD_SIG, 4) == 0)
{
n_eocd_index = i;
break;
}
}
if (n_eocd_index == −1)
{
free(szbuffer);
fclose(fp);
return −1;
}
// (5) Read Central Directory offset from EOCD (bytes 16–19, little-endian)
unsigned char *p_eocd = szbuffer + n_eocd_index;
long cd_offset = p_eocd [16] |
(p_eocd [17] << 8) |
(p_eocd [18] << 16) |
(p_eocd [19] << 24);

// (6) Move to Central Directory offset and verify its signature
fseek(fp, cd_offset, SEEK_SET);
unsigned char lst_cd_sig[4];
fread(lst_cd_sig, 1, 4, fp);
if (memcmp(lst_cd_sig, DEF_CD_SIG, 4) ! = 0)
{
// (5–1) It is tampered status
printf("Central Directory signature mismatch at offset %ld. File may be tampered.\n", cd_offset);
}
else
{
// (5–1) It is Normal status
printf("EOCD and Central Directory are valid. CD starts at offset %ld.\n", cd_offset);
}
free(szbuffer);
fclose(fp);
return 0;
}
```

Appendix A.3. Structural Validation of JPEG Files Based on SOI/EOI Signatures

To validate the integrity of JPEG files, we implement a header and footer marker check using standard Start of Image (SOI) and End of Image (EOI) bytes defined in the JPEG specification (Algorithm A3).

Algorithm A3: Structural Validation of JPEG Files Based on SOI/EOI Signatures

```
typedef unsigned char BYTE;
int check_jpg_header(const char* path) {
FILE* fp = fopen(path, "rb");
if (!fp) {
perror("Failed to open file");
return −1;
}
BYTE head[4] = {0};
fread(head, 1, 4, fp);

// Check Start of Image (SOI) marker and APPn segment range
// JPEG must begin with: 0xFF 0xD8 0xFF 0xE0~0xEF
if (!(head[0] == 0xFF && head[1] == 0xD8 && head[2] == 0xFF && (head[3] & 0xF0) == 0xE0)) {
fclose(fp);
return −1; // Not a valid JPEG SOI + APPn segment
}

// Seek to last 2 bytes to verify End of Image (EOI) marker
// JPEG must end with: 0xFF 0xD9
fseek(fp, −2, SEEK_END);
BYTE tail[2] = { 0 };
fread(tail, 1, 2, fp);
fclose(fp);
if (tail[0] == 0xFF && tail[1] == 0xD9)
return 0; // Valid JPEG file
else
return −1; // Invalid or missing EOI marker
}
```

Appendix A.4. Structural Validation of PDF Files

A PDF file is validated by first checking the version header (%PDF-) and then examining the tail region to confirm the presence of the startxref and %%EOF markers. The file is regarded as valid only when both markers are detected (Algorithm A4).

Algorithm A4: Structural Validation of PDF Files

```
int check_pdf_file (const char* path) {
float version = get_pdf_version(path);
if (version < 0.0f) {
printf("PDF header not found\n");
return 1;
}

FILE* fp = fopen(path, "rb");
if (!fp) return −1;
fseek(fp, 0, SEEK_END);
long filesize = ftell(fp);
long seek_pos = (filesize > MAX_TAIL_SIZE) ? filesize - MAX_TAIL_SIZE : 0;
fseek(fp, seek_pos, SEEK_SET);
unsigned char* tail = (unsigned char*)mal loc(MAX_TAIL_SIZE);
if (!tail) {
fclose(fp); return −1;
}
size_t read = fread(tail, 1, MAX_TAIL_SIZE, fp);
fclose(fp);

int has_eof = mem_contains(tail, read, "%%EOF");
int has_startxref = mem_contains(tail, read, "startxref");
int result = 1;
if (has_eof && has_startxref)
result = 0;
free(tail);
return result;
}
```

References

1. Cyberint. Ransomware Annual Report 2024. Available online: https://cyberint.com/blog/research/ransomware-annual-report-2024/ (accessed on 11 June 2025).
2. Sophos. *The State of Ransomware 2024*; Sophos Ltd.: Oxford, UK, 2024; Available online: https://www.sophos.com/en-us/content/state-of-ransomware (accessed on 7 June 2025).
3. Aung, Y.L.; Khoo, Y.L.; Zheng, D.Y.; Swee Duo, B.; Chattopadhyay, S.; Zhou, J.; Lu, L.; Goh, W. HoneyWin: High-Interaction Windows Honeypot in Enterprise Environment. *arXiv* **2025**. [CrossRef]
4. AV-TEST Institute. Every Day, the AV-TEST Institute Registers over 450,000 New Malicious Programs (Malware) and Potentially Unwanted Applications (PUA). Available online: https://www.av-test.org/en/statistics/malware/ (accessed on 11 June 2025).
5. Bowen, B.M.; Hershkop, S.; Keromytis, A.D.; Stolfo, S.J. Baiting Inside Attackers Using Decoy Documents. In *Security and Privacy in Communication Networks: Revised Selected Papers*; Chen, Y., Dimitriou, T.D., Zhou, J., Eds.; Lecture Notes in Computer Sciences; Springer: Berlin/Heidelberg, Germany, 2009; Volume 19, pp. 51–70. [CrossRef]
6. Kharraz, A.; Arshad, S.; Mulliner, C.; Robertson, W.; Kirda, E.A. Large-Scale, Automated Approach to Detecting Ransomware. In Proceedings of the 25th USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016; pp. 757–772. Available online: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz (accessed on 31 July 2025).
7. Zakaria, W.Z.A.; Abdollah, M.F.; Mohd, O.; Yassin, S.M.W.M.; Ariffin, A. RENTAKA: A Novel Machine Learning Framework for Crypto-Ransomware Pre-Encryption Detection. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 378–385. [CrossRef]
8. Mehra, T. The Role of Encryption in Securing Backup Data Against Ransomware Threats. *Int. J. Sci. Res. Arch.* **2024**, *13*, 1971–1974. [CrossRef]
9. Dafoe, J.; Chen, N.; Chen, B.; Wang, Z. Enabling Per-File Data Recovery from Ransomware Attacks via File System Forensics and Flash Translation Layer Data Extraction. *Cybersecurity* **2024**, *7*, 75. [CrossRef]
10. Gayathri, G.; Arivoli, R.S. A Comprehensive Behavioural Study of Ransomware and Its Impact. In Proceedings of the International Conference on Innovative Computing & Communication (ICICC 2024), New Delhi, India, 16–17 February 2024; Available online: https://ssrn.com/abstract=5022928 (accessed on 11 June 2025).
11. Thomas, M.C.; Joy, A.T. *Elements of Information Theory*; Wiley Interscience: Hoboken, NJ, USA, 2006.
12. Hassan, M.W.; Goel, N.; Kalyan, T.V. CARDR: DRAM Cache Assisted Ransomware Detection and Recovery in SSDs. In Proceedings of the ACM International Symposium on Memory Systems (MEMSYS 2024), Washington, DC, USA, 30 September–3 October 2024. [CrossRef]
13. Jadon, R.; Srinivasan, K.; Chauhan, G.S.; Budda, R.; Gollapalli, V.S.T.; Prema, R. Enhanced Ransomware Detection and Prevention Using CNN-BiLSTM for Deep Behavioural Analysis. *Int. J. Recent Adv. Multidiscip. Res.* **2025**, *12*, 10900–10904. Available online: https://ijramr.com/issue/enhanced-ransomware-detection-and-prevention-using-cnn-bilstm-deep-behavioural-analysis (accessed on 31 July 2025).
14. Higuchi, K.; Kobayashi, R. ROFBSα: Real-Time Backup System Decoupled from ML-Based Ransomware Detection. *arXiv* **2025**. [CrossRef]
15. Microsoft Docs. IRP Major Function Codes. Available online: https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/irp-major-function-codes (accessed on 8 June 2025).
16. ytisf. theZoo: A Live Repository of Malware Samples. Available online: https://github.com/ytisf/theZoo (accessed on 28 July 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.