ISSN: 2182-2069 / E-ISSN: 2182-2077

Ransomware Detection via Decoy Trap and File Traversal Entropy Checks

Seoyeon Kim[1], TaeGuen Kim[2], and Jae Yeol Jeong[3*]

[1]Department of Mobility Convergence Security, Soonchunhyang University, Seoul, South Korea. seoyeon56@sch.ac.kr, https://orcid.org/0009-0001-7213-8522

[2]Department of AI Cybersecurity, Korea University, Seojong Campus, Sejong, Korea. tg_kim@korea.ac.kr, https://orcid.org/0000-0002-6586-2037

[3*]Department of Information Security, Soonchunhyang University, Seoul, South Korea. blue7angels@sch.ac.kr, https://orcid.org/0000-0002-6743-6410

Received: August 07, 2025; Revised: September 13, 2025; Accepted: October 22, 2025; Published: November 28, 2025

Abstract

Ransomware is a type of malware that encrypts a victim's files and demands payment in exchange for the decryption key, and it has rapidly become one of the most pervasive global cyber threats. This has led to substantial financial losses and has emerged as a serious social problem worldwide. However, reliably detecting zero-day ransomware remains a challenging task. Most commercial antivirus solutions still rely primarily on signature-based detection, which suffers from inherent limitations, as generating accurate signatures requires time-consuming and in-depth analysis of suspicious programs. To overcome these limitations, this paper proposes a hybrid ransomware defense framework that combines previously studied decoy-based detection techniques with a dynamic entropy-based verification mechanism. The deception-based component builds on existing decoy file techniques by strategically deploying decoy files to increase the likelihood of exposing malicious behavior during the early stages of file encryption. In addition, this work newly designs and applies a dynamic entropy checking method that continuously monitors abnormal encryption patterns in file access and content modification, thereby mitigating potential detection blind spots caused by the inherent constraints of decoy-based approaches. In the experimental evaluation, we assess the effectiveness and robustness of the proposed framework, including scenarios involving random-access attacks that may be used to evade traditional decoy-based defenses, and report the corresponding results.

Keywords: Malicious Software Detection, Dynamic Analysis, Ransomware Detection, Hybrid Framework.

1 Introduction

Recently, ransomware campaigns targeting Windows-based and other mainstream platforms have continued to proliferate worldwide, with large organizations and critical infrastructure remaining prime targets. Recent telemetry shows millions of ransomware detections annually and a non-trivial fraction of users and enterprises directly affected; for example, Kaspersky reports several million ransomware detections in 2023–2024 with targeted attacks accounting for a significant share of incident-response cases, while Windows systems remain the primary focus of many major groups (Nyonyoh, 2025). On the financial side, Chainalysis

Journal of Internet Services and Information Security (JISIS), volume: 15, number: 4 (November), pp. 389-400.
DOI: *10.58346/JISIS.2025.I4.028*

*Corresponding author: Department of Information Security, Soonchunhyang University, Asan, South Korea.

estimates that known ransomware payments exceeded 1.25 billion USD in 2023 and remained on the order of hundreds of millions of dollars in 2024 despite increased enforcement and a decline in the proportion of victims who pay (Tamminen, 2025). Survey-based studies by Sophos similarly indicate that well over half of organizations have experienced ransomware incidents in recent years, with typical recovery costs (including downtime) and average ransom payments reaching into the millions of dollars (Sophos, 2024). These figures underscore that ransomware is no longer a marginal threat but a persistent global risk causing severe operational disruption and economic damage.

While ransomware shares many characteristics with general malware, it exhibits a distinctive operational pattern: an infected host performs a burst of intensive file I/O and encryption-like transformations over a large set of user files in a short time window to maximally lock the victim's data. Once this process completes, restoring files without the attacker's private keys is usually infeasible. Even when analysts fully reconstruct the behavior of a given strain through reverse engineering, the keys may be generated per victim and stored or derived server-side, leaving victims with few options other than backups or decryption tools released by investigators.

Conventional endpoint protection products still rely heavily on signature-based detection, which uses static artifacts (e.g., byte patterns, strings, or structural code features) extracted from known malware families. Such techniques are highly effective for already characterized variants but fundamentally struggle with zero-day ransomware and fast-evolving families, where reliable signatures are unavailable or can be obfuscated quickly. Moreover, full dynamic analysis on end hosts is rarely feasible due to performance and scalability constraints. As a result, there is a clear need for mechanisms explicitly tailored to the behavioral nature of ransomware, capable of reacting early in the encryption phase without imposing prohibitive overhead.

Motivated by this, prior work has explored two primary directions: (1) deception-based methods that plant decoy files and trigger alarms when those files are accessed or modified in a ransomware-like manner, and (2) behavior-based or monitoring-based methods that track file system operations or content changes to identify anomalous encryption activity. Building on these lines of research rather than replacing them, we propose a new hybrid ransomware detection framework that tightly integrates established decoy-file techniques with a lightweight dynamic monitoring component.

In our design, the deception-based subsystem adopts and refines existing decoy deployment strategies to ensure that realistically placed and attractive decoy files are highly likely to be touched during the early stages of an attack. Complementing this, we introduce a directory entropy checker, a dynamic monitor-based mechanism intended specifically to support and reinforce decoy-based detection rather than to perform heavyweight generic ransomware behavior analysis. The directory entropy checker observes changes in aggregated entropy characteristics within selected directories and flags suspicious patterns that are consistent with bulk encryption, thereby compensating for blind spots where ransomware may partially evade or bypass decoy files. In this work, we reuse the deception-based method as established in our prior study without modification, and focus on designing the dynamic monitoring-based method and integrating it with the existing deception-based approach to construct a unified hybrid framework.

2 Ransomware Sample Analysis

We examined how real-world ransomware traverses' directories in practice. Since ransomware must repeatedly perform file operations to locate and encrypt candidate files, we hypothesized that its directory traversal would resemble systematic file system search strategies rather than random access.

(a) Wannacry (b) Cryptowall (c) Locky

Figure 1: Example of hopping distance changes of ransomware samples

In general, file system enumeration is commonly implemented using two fundamental graph search algorithms: depth-first search (DFS) and breadth-first search (BFS). DFS explores nodes "vertically" along a path before backtracking, whereas BFS explores "horizontally" across nodes at the same depth. For both algorithms, the actual visitation order can vary depending on how directories and files at the same level are prioritized. To capture representative patterns, we modeled four canonical traversal strategies:

- DFS with lexicographic (alphabetical) ordering (DFS-A)
- DFS with reverse-lexicographic ordering (DFS-R)
- BFS with lexicographic ordering (BFS-A)
- BFS with reverse-lexicographic ordering (BFS-R)

We then executed ransomware samples in a controlled environment and recorded their file-related behaviors by logging relevant API calls. From these logs, we extracted the sequences of file paths used as arguments and analyzed how closely these sequences matched the four modeled traversal strategies. Specifically, we constructed bigrams from each path sequence (pairs of consecutive file paths) and compared the bigram frequency distributions against those obtained from simulated DFS-A, DFS-R, BFS-A, and BFS-R traversals. Using the overlap coefficient defined in Equation (1), we quantified the similarity between observed and modeled bigram distributions, thereby inferring which traversal strategy each ransomware process most likely followed.

$$overlap(a,b) = \frac{|a \cap b|}{\min(|a|,|b|)} \quad (1)$$

$$hopdist(a,b) = h(a) + h(b) - 2 * c(a,b) \quad (2)$$

The results, summarized in Table 1, show that 94.38% of the ransomware processes exhibited bigram patterns most consistent with DFS-A, with a substantial margin over BFS-A, and effectively no processes aligned with DFS-R or BFS-R. To further validate that these behaviors reflect a structured traversal strategy rather than incidental similarity, we computed the "hopping distance" between consecutive file paths using Equation (2).

where a and b denote the previous and current paths, respectively; h( ) is a function that outputs the height of a given path. The height is the number of directories, except the root directory included in the path. The function c( ) is the length of the consecutive directory sequences present in both of the two given paths. For example, if paths a and b are "C:\\x\\y\\z" and "C:\\x\\y," then h(a) and h(b) will be three and two respectively, and c(a,b) will be two.

As illustrated in Figure 1, the hopping distances for the analyzed samples mostly fell between 0 and 10, with a standard deviation of approximately 1.32, reinforcing the observation that many ransomware families employ systematic, search-like traversal patterns consistent with our modeled cases.

These empirical findings form the basis for our decoy deployment strategy. Following established deception-based approaches from prior work, we do not simply cluster decoy files in a few obvious locations; instead, we distribute decoy files across the file system in a manner aligned with typical traversal routes inferred from the DFS-A–like behavior. This increases the likelihood that ransomware will interact with at least one decoy file in its early encryption phase, without easily distinguishing it as bait.

Additionally, a key claim of this work is that, as shown in our ransomware analysis, not all samples strictly follow the four traversal strategies we described, and that if a ransomware adopts a randomized file access pattern, the effectiveness of purely decoy-based detection can be significantly degraded.

Therefore, we incorporate a complementary dynamic monitoring mechanism to address more evasive scenarios, such as ransomware that deliberately accesses files in a pseudo-random fashion to circumvent decoy-based defenses, thereby strengthening overall robustness beyond what deception-based techniques alone can guarantee.

Table 1: Ransomware bigram frequency results

	DFS-A	**BFS-A**	**DFS-R**	**BFS-R**
The portion of the matched processes (%)	94.375	1.25	3.75	0.625
The average similarity with the processes	35.7	29.4	14.54	12.24

3 Related Works

Prior work on ransomware can be broadly grouped into three areas: behavioral analysis, protection techniques based on system/file monitoring or deception, and machine learning–based detection.

First, several studies have concentrated on understanding how ransomware families operate on compromised systems. (Gonzalez & Hayajneh, 2017) investigated the behavior of prevalent ransomware strains and reviewed existing countermeasures. (Gazet, 2010) examined early ransomware samples using reverse engineering to characterize their functionality and cryptographic routines. (Pathak & Nanded, 2016) analyzed multiple ransomware families, describing their behavioral traits and outlining basic preventive measures such as regular backups and software updates. (Cabaj et al., 2015) provided an in-depth analysis of CryptoWall, focusing on its network communications; their study showed that CryptoWall leverages HTTP-based channels and employs concealment techniques including Tor and deceptive DNS usage. Collectively, these works laid the foundation for understanding ransomware behavior at the code, system, and network levels.

Building on such insights, a number of protection mechanisms have been proposed that monitor system changes or file content characteristics. (Jung & Won, 2018) and (Lee et al., 2019) leveraged entropy measurements of file content to distinguish encrypted data from normal files. (Scaife et al., 2016) introduced a behavior-based approach that tracks indicators such as file type modifications, sdhash variations, and file entropy to flag ransomware-like activities. (Kharaz et al., 2016) developed UNVEIL, a dynamic analysis system that observes I/O behavior, checks the entropy of buffer data, and inspects suspicious access patterns, while also comparing screenshots to identify ransomware-induced screen changes. (Continella et al., 2016) proposed ShieldFS, a self-healing filesystem layer for Windows that monitors features such as file read/write/rename frequency and write entropy, caches original file versions on a shadow volume, and restores them when ransomware activity is detected. (Al-Rimy et al., 2020) modeled ransomware behavior using annotated TF-IDF features derived from API invocation frequencies observed in sandboxed execution. Although effective in controlled or post-compromise scenarios, these approaches generally infer an attack only after substantial file system modifications or repeated encryption attempts have occurred, which

necessitates non-trivial observation time. By contrast, our framework aims to support near real-time detection on end hosts by leveraging strategically deployed decoy files and lightweight entropy-based monitoring, thereby reacting earlier in the attack lifecycle.

In parallel, several deception-oriented approaches have been explored. (Chakkaravarthy et al., 2020) targeted IoT environments, deploying decoy files and monitoring process-level resource usage such as CPU load, memory consumption, and active time when those decoys are manipulated. (Feng et al., 2017) proposed a decoy-based framework that observes access to decoy files as a trigger for ransomware detection. (Moore, 2016) implemented a ransomware honeypot and empirically showed that many samples traverse files in alphabetic order, aligning with later findings on structured traversal. (Kok et al., 2019) surveyed ransomware behavior and existing defenses, emphasizing that honeypot-based traps and decoy files are not guaranteed to be accessed by all ransomware variants. A key limitation across these deception-based schemes is the lack of robustness against adversaries that adopt randomized or non-standard traversal strategies; such attackers may simply avoid decoy locations, intentionally or incidentally. Our work addresses this gap by employing a decoy deployment strategy that distributes decoy files systematically across the file system structure, improving the likelihood of contact even under more diverse traversal patterns and strengthening resilience against random-access style evasion.

Alongside behavioral and deception-based defenses, numerous machine learning–driven methods for ransomware detection have been proposed. (Takeuchi et al., 2018) extracted n-grams from API call sequences and trained SVM classifiers to discriminate ransomware behavior. (Poudyal et al., 2018) utilized assembly-level features with decision tree and random forest models. (Maniath et al., 2017) adopted an LSTM-based architecture that ingests API call sequences to classify samples as benign or ransomware. (Cusack et al., 2018) (implicitly referenced) relied on network flow features such as protocol type, IP information, flags, and TTL values, for classification. (Poudyal et al., 2019) further extended static analysis by using n-gram features of DLL usage, function calls, and assembly instructions as input to machine learning classifiers. (Almashhadani et al., 2019) modeled Locky ransomware behavior using network features from TCP, HTTP, and DNS traffic. (Chen et al., 2017) analyzed API call graphs combined with several learning algorithms, and subsequent work (Sharmeen et al., 2020).

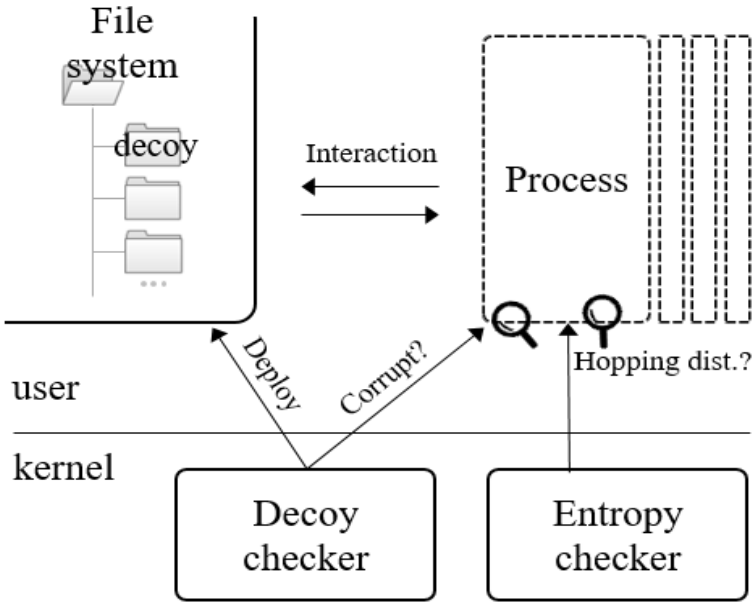

Figure 2: Our proposed framework

Explored deep learning techniques using API traces gathered via sandbox execution. **(**Bae et al., 2020) proposed a TF-IDF–weighted n-gram approach over API sequences, applying multiple classifiers such as Random Forest and SVM.

4 Proposed Method

As illustrated in Figure 2, our proposed ransomware protection framework is composed of two tightly integrated modules: a deception-based protection module ("decoy checker") and a dynamic monitor-based detection module ("entropy checker"). The decoy checker adopts and refines decoy file deployment strategies previously explored in earlier studies, placing decoy files strategically and monitoring whether they are modified in a ransomware-like manner. Building on top of this established deception-based approach, the entropy checker performs lightweight dynamic analysis by continuously computing entropy-based indicators that reflect irregularities and diversity in a process's file system traversal and content modification behavior. By combining prior decoy-based techniques with this complementary dynamic entropy monitoring, the framework operates as a hybrid protection mechanism, providing more robust and early-stage detection capabilities than either component could achieve in isolation. Detailed design and operational procedures of each module are described in the following subsections.

Deception-based Protection

Our deception-based protection module adopts a previously developed decoy deployment strategy that evenly spreads realistic decoy files across the file system rather than clustering them in a few locations. In that prior work, the directory structure is modeled as a graph, partitioned into sub-file systems using spectral clustering with a hopping-distance-based similarity metric, and key directories along typical ransomware traversal paths (e.g., DFS/BFS-prioritized paths) are selected as decoy placement points. Decoy files are then generated to closely resemble genuine user files in type, size, and naming, and are configured as superhidden to remain inconspicuous to normal users while still accessible to ransomware. Each deployed decoy file is continuously monitored, and any ransomware-like manipulation serves as an early and reliable indicator of malicious activity. In this study, we briefly incorporate this established deception-based mechanism as one component of our hybrid protection framework.

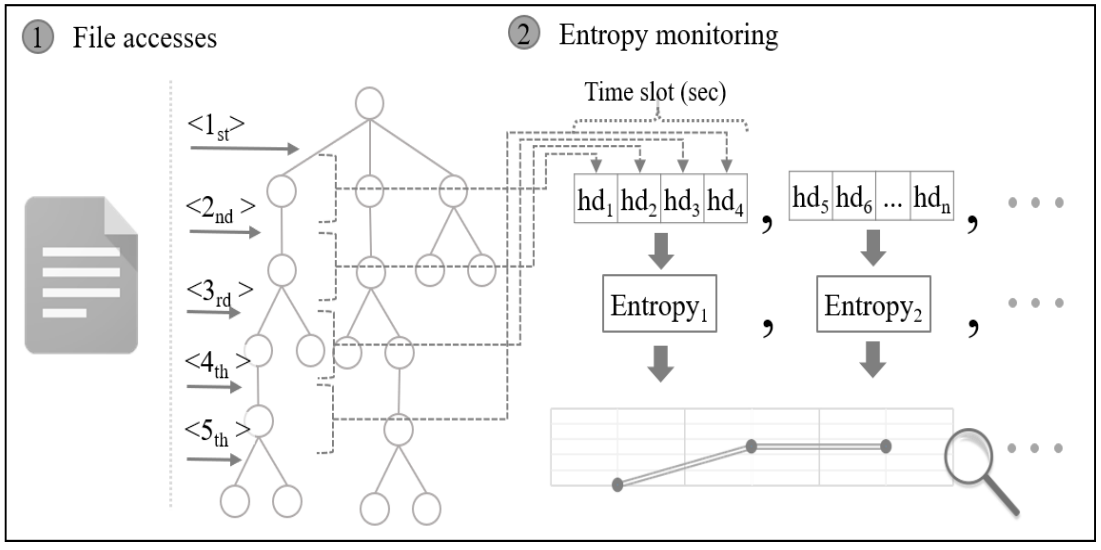

Figure 3: Process flow of dynamic monitoring-based protection

Dynamic Monitoring-based Protection

The entropy checker provides a complementary dynamic monitoring-based protection mechanism designed to detect ransomware that attempts to evade decoy-based defenses by accessing files in a pseudo-random manner. While the deception-based method assumes that ransomware follows typical directory traversal patterns (such as DFS- or BFS-like exploration), a more evasive adversary may intentionally randomize its access sequence to reduce the probability of touching strategically placed decoy files in the early stages of an attack. To handle this threat model, our framework evaluates how "structured" or "random" a process's file access behavior is, using an entropy metric derived from file system geometry.

Figure 3 summarizes the core procedure. In the first step, each file operation issued by a process is mapped onto the directory tree, and successive file paths are observed along the file system hierarchy. Rather than treating file names as independent items, we interpret each path in terms of its position in the tree and the transition (hopping distance) from the previously accessed path. This captures how far and in what manner the process moves through the file system (e.g., staying within the same directory, moving one level up or down, or jumping across distant branches) In the second step, these hopping distances are collected over fixed-length time slots (e.g., several seconds). For each time slot, the framework forms a set of hopping distances, denoted as hdset, where each element hd_i is computed using the previously defined hopping distance function based on Equation (2). From this set, the entropy value is calculated according to Equation (3), which quantifies the diversity and distribution of the hopping distances within that interval.

$$Entropy(hdset) = -\sum_{i=1}^{k} P(hd_i) \cdot log_2(P(hd_i)) \quad (3)$$

Operationally, this process unfolds as follows. For every monitored process, its consecutive file accesses are segmented into time windows. Within each window, the hopping distances are computed and aggregated, and an entropy score is derived to represent the degree of randomness in the file access pattern. These entropy scores are then tracked over time, as depicted in the right side of Figure 3: each block of hopping distances produces an $Entropy_1$, $Entropy_2$, …, and the resulting sequence of entropy values is examined for abnormal trends. A benign application or typical system utility often exhibits relatively structured and repetitive traversal patterns, leading to more stable or predictable entropy profiles. In contrast, ransomware attempting to randomize its traversal or aggressively sweep diverse regions of the file system tends to generate higher and more irregular entropy values, reflecting frequent long-distance jumps or inconsistent access locality.

By emphasizing both the spatial (tree position) and temporal (time-slot-based) characteristics of file access behavior, the entropy checker can effectively highlight suspicious randomness that decoy files alone might miss. Even if the decoy files are not immediately accessed, a process that persistently produces entropy patterns indicative of wide, irregular, and intensive traversal can be flagged for further inspection or mitigation. In this way, the entropy checker reinforces the deception-based module and contributes to a more robust hybrid defense against both conventional and evasive ransomware strategies.

5 Evaluation

In this section, the performance of the dynamic monitoring-based detection method is evaluated. The performance result of the deception-based detection method is included in. In our experiments, a total of 94 ransomware samples, were utilized in the experiments. Additionally, we collected a set of 100 benign samples that performed file-related operations, including well-known compression or encryption applications like 7-zip and WinRAR. In addition, for the experiments, we simulated ransomware capable of conducting random-access attacks. The simulated ransomware utilizes PowerShell, a command-line shell, to obtain a comprehensive list of all files within the file system. Subsequently, it randomly selects a file path to access.

Upon selecting a file, the ransomware proceeds to encrypt it and then deletes the original file. This cycle of file selection, encryption, and deletion continues iteratively until all files are encrypted. The ransomware was executed repeatedly, and the number of encrypted files was recorded for analysis.

We had two experiments for the evaluation of the dynamic monitoring-based protection. Firstly, we checked how the entropy values are varied by the different experimental data sets. The ransomware samples and the benign programs in real-world were used in the experiment, and the ransomware that utilizes the random-access attack was simulated and tested. In addition, we conducted an experiment to confirm that the

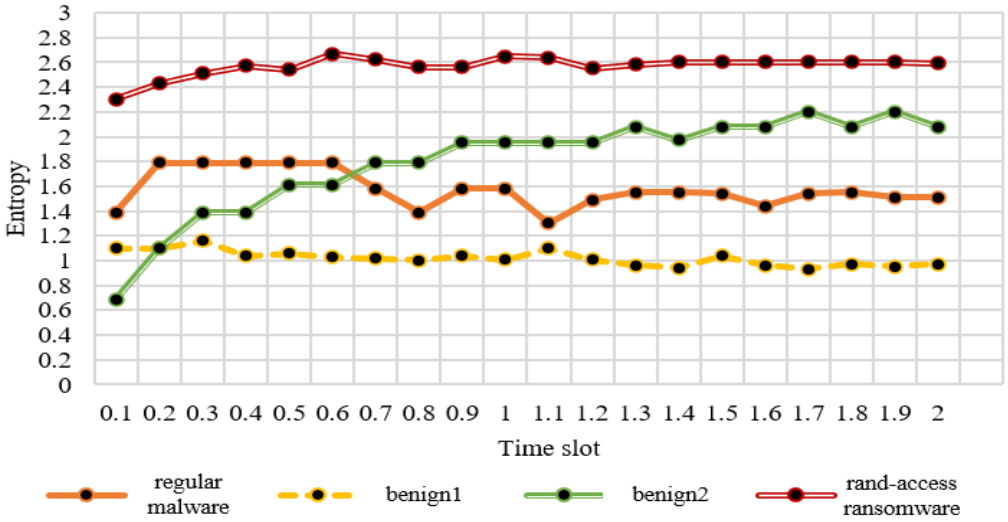

Figure 4: Maximum entropy as a function of the time slot of the sliding window. benign1 is a set of compression applications. benign2 is a set of encryption programs, and the random-access ransomware is the simulated ransomware program)

module can detect the random-access attack, and we showed how the detection accuracy varies with the different parameters that can affect the detection result.

Entropy Differences with the Random-access Activity

As a result of the manual analysis using ransomware samples, it was observed that ransomware share similar file access patterns. Considering this tendency, our previously proposed method protects the files by deploying decoy files based on the file access pattern of the ransomware in a partition. Previous experimental results demonstrate that the robustness of our method against a random file access attack is improved by the proposed decoy deployment strategy, but naturally, an attack cannot be totally avoided because it is infeasible to deploy decoys in all directories in the file system, which is an implicit limitation of deception-based protection.

For this reason, our proposed method additionally applies dynamic monitoring-based protection to compensate for the limitation by monitoring file access entropy. The purpose of dynamic monitoring-based protection is to find a process that accesses the files irregularly. It was not designed to detect malicious file-related behaviors. Dynamic monitoring-based protection was evaluated for the purpose it was designed. The entropy was measured for ransomware file accesses of various types of samples. Additionally, the ransomware that access files randomly was executed 100 times. The simulated ransomware, accessed files randomly in the test because the dynamic monitoring-based method was designed to detect random access.

Figure 4 displays the maximum entropy for each sample type, which varies as a function of the time slot of the sliding window. As is observed in the graph, the maximum entropy in the random-access ransomware set is much higher, exhibiting a clear difference, when compared against the other sets. The average value of the maximum entropies in all cases was approximately 2.57. The maximum entropies of the compression/encryption program were in the range of 0.93 to 1.16, and the maximum entropies of the existing malware were in the range of 1.3 to 1.79. For samples that travel directories in a regular manner, the values were relatively low. In the case of benign samples, except for the compression / encryption programs, the entropy measured in a short time slot period was very low, but the value increased steadily until the time slot was set to 1 s. The entropy for the time slot of 0.1 s was 0.69, and the value for a time slot of 1 s was 1.95. Once the time slot exceeded 1 s, the value increased slightly, and saturated at approximately 2.0 for a time slot of 1.3 s. Based on this experimental result, it was confirmed that a detection threshold can be specified to detect random-access ransomware.

Detection Accuracy with Different Parameter Settings

Finally, an experiment was conducted to determine the performance of the dynamic monitoring-based protection method with different parameter settings. The parameters that can affect performance are time slot and detection threshold. In this experiment, only random-access ransomware and benign samples were used. The accuracies for the different parameters are listed in Table 2. The values within the bold line area represent the accuracy when the false positive rate is zero.

The protection mechanism that blocks the normal behavior of the benign program by mistake may be a significant problem for users in terms of usability. From this perspective, it is necessary to select the parameters included within the bold line area to ensure that the protection method does not produce false positives. The zero-false positive cases show that the threshold tends to increase as the time slot increases. Therefore, it is necessary to first select an appropriate time slot for the host system to accommodate and set the threshold value for the maximum detection rate accordingly. In our research, the best parameter for dynamic monitoring-based protection is (time slot = 1 s, threshold = 2.0, or 2.1).

This parameter setting is selected among all settings to produce an accuracy of 0.99 because the time slot value is the largest. The time slot value determines the frequency of the entropy calculation in the detection; therefore, the longest time slot is preferred to reduce the overall overhead for dynamic monitoring.

6 Conclusion

This work proposes a practical hybrid ransomware defense framework that combines an established deception-based decoy file deployment strategy with a lightweight dynamic entropy-based monitor to enable early, real-time detection on end hosts. Recognizing that advanced ransomware may evade such traps via pseudo-random file access, they introduce a directory entropy checker that models file access behavior in terms of “hopping distances” on the directory tree and computes entropy over sliding time windows to quantify how structured or random each process’s traversal is. Using 94 real ransomware samples, 100 benign programs, and a simulated random-access ransomware, they demonstrate that random-access attacks exhibit significantly higher entropy and that, with appropriate parameter settings (e.g., 1 s window, entropy threshold around 2.0 to 2.1), the method achieves high detection accuracy (~0.99) with zero false positives and negligible overhead. Overall, the framework strengthens decoy-centric protection against both conventional and evasive ransomware by tightly integrating deception and geometry-aware dynamic monitoring.

Table 2: The heat map for accuracy (gray scale representation: white means the largest value and the darkest gray means the smallest value, The accuracies in the bold line area are the values with 0 false positive rate.)

		Time slot (sec)																			
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**1.7**	**1.8**	**1.9**	**2**
Detection threshold	**1**	0.99	0.98	0.62	0.57	0.55	0.55	0.54	0.54	0.54	0.54	0.53	0.53	0.53	0.53	0.52	0.53	0.53	0.53	0.53	0.53
	1.1	0.99	0.99	0.99	0.86	0.77	0.68	0.66	0.62	0.61	0.6	0.59	0.6	0.58	0.56	0.56	0.56	0.56	0.55	0.55	0.55
	1.2	0.99	0.99	0.99	0.86	0.77	0.68	0.66	0.62	0.61	0.6	0.6	0.6	0.58	0.56	0.56	0.57	0.56	0.56	0.55	0.55
	1.3	0.99	0.99	0.99	0.86	0.77	0.68	0.66	0.63	0.61	0.6	0.6	0.62	0.6	0.57	0.58	0.58	0.57	0.58	0.57	0.56
	1.4	0.99	0.99	0.99	0.99	0.97	0.92	0.87	0.85	0.8	0.78	0.77	0.77	0.75	0.72	0.71	0.69	0.69	0.68	0.66	0.66
	1.5	0.99	0.99	0.99	0.99	0.97	0.92	0.87	0.85	0.8	0.79	0.77	0.78	0.77	0.74	0.72	0.69	0.69	0.7	0.67	0.68
	1.6	0.99	0.99	0.99	0.99	0.97	0.92	0.89	0.87	0.84	0.83	0.79	0.83	0.82	0.79	0.77	0.81	0.79	0.77	0.75	0.74
	1.7	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.96	0.95	0.9	0.89	0.89	0.87	0.85	0.86	0.86	0.86	0.83	0.83	0.81
	1.8	0.97	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.97	0.97	0.95	0.96	0.93	0.93	0.91	0.91	0.91	0.9
	1.9	0.95	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.97	0.97	0.95	0.96	0.95	0.95	0.93	0.92	0.93	0.92
	2	0.92	0.97	0.98	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.98	0.97	0.96	0.96	0.95
	2.1	0.64	0.95	0.97	0.98	0.98	0.98	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.99
	2.2	0.54	0.92	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.99	0.99	0.99	0.99	0.99	0.99
	2.3	0.52	0.75	0.93	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.99	0.99	0.97	0.99	0.99
	2.4	0.51	0.52	0.72	0.79	0.82	0.85	0.9	0.92	0.93	0.93	0.93	0.96	0.96	0.94	0.96	0.94	0.91	0.94	0.97	0.96
	2.5	0.51	0.51	0.52	0.54	0.56	0.64	0.66	0.66	0.64	0.7	0.62	0.67	0.69	0.71	0.73	0.68	0.66	0.68	0.68	0.66
	2.6	0.51	0.51	0.51	0.51	0.51	0.54	0.52	0.51	0.51	0.56	0.52	0.5	0.5	0.51	0.51	0.51	0.51	0.5	0.5	0.5
	2.7	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	2.8	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

References

[1] Almashhadani, A. O., Kaiiali, M., Sezer, S., & O'Kane, P. (2019). A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE access*, *7*, 47053-47067. https://doi.org/10.1109/ACCESS.2019.2907485

[2] Al-Rimy, B. A. S., Maarof, M. A., Alazab, M., Alsolami, F., Shaid, S. Z. M., Ghaleb, F. A., ... & Ali, A. M. (2020). A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. *IEEE Access*, *8*, 140586-140598. https://doi.org/10.1109/ACCESS.2020.3012674

[3] Bae, S. I., Lee, G. B., & Im, E. G. (2020). Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience*, *32*(18), e5422. https://doi.org/10.1002/cpe.5422

[4] Cabaj, K., Gawkowski, P., Grochowski, K., & Osojca, D. (2015). Network activity analysis of CryptoWall ransomware. *Przeglad Elektrotechniczny*, *91*(11), 201-204. https://doi.org/10.15199/48.2015.11.48

[5] Chakkaravarthy, S. S., Sangeetha, D., Cruz, M. V., Vaidehi, V., & Raman, B. (2020). Design of intrusion detection honeypot using social leopard algorithm to detect IoT ransomware attacks. *IEEE Access*, *8*, 169944-169956. https://doi.org/10.1109/ACCESS.2020.3023764

[6] Chen, Z. G., Kang, H. S., Yin, S. N., & Kim, S. R. (2017, September). Automatic ransomware detection and analysis based on dynamic API calls flow graph. In *Proceedings of the international conference on research in adaptive and convergent systems* (pp. 196-201). https://doi.org/10.1145/3129676.3129704

[7] Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., & Maggi, F. (2016, December). Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the*

32nd annual conference on computer security applications (pp. 336-347). https://doi.org/10.1145/2991079.2991110

[8] Cusack, G., Michel, O., & Keller, E. (2018, March). Machine learning-based detection of ransomware using SDN. In *Proceedings of the 2018 ACM international workshop on security in software defined networks & network function virtualization* (pp. 1-6). https://doi.org/10.1145/3180465.3180467

[9] Feng, Y., Liu, C., & Liu, B. (2017, May). Poster: A new approach to detecting ransomware with deception. In *38th IEEE symposium on security and privacy*.

[10] Gazet, A. (2010). Comparative analysis of various ransomware virii. *Journal in computer virology*, *6*(1), 77-90.

[11] Gonzalez, D., & Hayajneh, T. (2017, October). Detection and prevention of crypto-ransomware. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)* (pp. 472-478). IEEE. https://doi.org/10.1109/UEMCON.2017.8249052

[12] Jung, S., & Won, Y. (2018). Ransomware detection method based on context-aware entropy analysis. *Soft Computing-A Fusion of Foundations, Methodologies & Applications*, *22*(20). https://doi.org/10.1007/s00500-018-3257-z

[13] Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., & Kirda, E. (2016). {UNVEIL}: A {Large-Scale}, automated approach to detecting ransomware. In *25th USENIX security symposium (USENIX Security 16)* (pp. 757-772).

[14] Kok, S., Abdullah, A., Jhanjhi, N., & Supramaniam, M. (2019). Ransomware, threat and detection techniques: A review. *International Journal of Computer Science and Network Security*, *19*(2), 136-146.

[15] Lee, K., Lee, S. Y., & Yim, K. (2019). Machine learning based file entropy analysis for ransomware detection in backup systems. *IEEE access*, *7*, 110205-110215. https://doi.org/10.1109/ACCESS.2019.2931136

[16] Maniath, S., Ashok, A., Poornachandran, P., Sujadevi, V. G., AU, P. S., & Jan, S. (2017, October). Deep learning LSTM based ransomware detection. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)* (pp. 442-446). IEEE. https://doi.org/10.1109/RDCAPE.2017.8358312

[17] Moore, C. (2016, August). Detecting ransomware with honeypot techniques. In 2016 Cybersecurity and Cyberforensics Conference (CCC) (pp. 77-81). IEEE. https://doi.org/10.1109/CCC.2016.14

[18] Nyonyoh, R. (2025). Ransomware and the Vulnerability of Critical Infrastructure: A National Security and Economic Analysis. *Asian Journal of Economics, Business and Accounting*, *25*(4), 412-422. https://doi.org/10.9734/ajeba/2025/v25i41759

[19] Pathak, P. B., & Nanded, Y. M. (2016). A dangerous trend of cybercrime: ransomware growing challenge. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *5*(2), 371-373.

[20] Poudyal, S., Dasgupta, D., Akhtar, Z., & Gupta, K. (2019, October). A multi-level ransomware detection framework using natural language processing and machine learning. In *14th International Conference on Malicious and Unwanted Software" MALCON* (No. October 2015).

[21] Poudyal, S., Subedi, K. P., & Dasgupta, D. (2018, November). A framework for analyzing ransomware using machine learning. In *2018 IEEE symposium series on computational intelligence (SSCI)* (pp. 1692-1699). IEEE. https://doi.org/10.1109/SSCI.2018.8628743

[22] Scaife, N., Carter, H., Traynor, P., & Butler, K. R. (2016, June). Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th international conference on distributed computing systems (ICDCS)* (pp. 303-312). IEEE. https://doi.org/10.1109/ICDCS.2016.46

[23] Sharmeen, S., Ahmed, Y. A., Huda, S., Koçer, B. Ş., & Hassan, M. M. (2020). Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches. *IEEE Access*, *8*, 24522-24534. https://doi.org/10.1109/ACCESS.2020.2970466

[24] Sophos (2024). The State of Ransomware 2024, *Sophos News.*
[25] Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018, August). Detecting ransomware using support vector machines. In *Workshop Proceedings of the 47th International Conference on Parallel Processing* (pp. 1-6). https://doi.org/10.1145/3229710.3229726
[26] Tamminen, J. (2025). *Cybercrime business evolved: A concept analysis of Ransomware-as-a-Service* (Doctoral dissertation, University of Jyväskylä).

Authors Biography

Seoyeon Kim received the B.S. degree in the Department of Information Security from Soonchunhyang University, Korea, in 2023, and the M.S. degree in the Department of Mobility Convergence Security from the same university in 2025. She is currently a junior researcher in automotive security at Kia Motors, where their work focuses on automotive cybersecurity. Their research interests include malware analysis, network security, and vehicle/automotive security.

TaeGuen Kim received his B.S. degree in Electronics and Computer Engineering in 2011 and his M.S. degree in Computer and Software Engineering in 2013, both from Hanyang University, South Korea. He earned his Ph.D. degree in Computer and Software Engineering from the same university in 2018. He subsequently worked as a Senior Research Engineer at Hyundai Motor Company (HMC), followed by an Assistant Professor position at Soonchunhyang University. Since September 2024, he has been serving as a faculty member at Korea University (Sejong Campus). His research interests include malware analysis, AI-driven security solutions, and automotive cybersecurity.

Jae Yeol Jeong received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in information security from Korea University, South Korea, in 2010, 2013, and 2022, respectively. From March 2022 to February 2024, he was a Research Professor with the BK21 Four R and E Center for Cybersecurity, Korea University. He is currently a professor in the Department of Information Security at Soonchunhyang University. His research interests include personal data security, biometric authentication security, drone security, and key exchange protocols, and he is actively conducting research in these areas.