Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057234

ARTICLE

AI-Driven Prioritization and Filtering of Windows Artifacts for Enhanced Digital Forensics

Juhwan Kim, Baehoon Son, Jihyeon Yu and Joobeom Yun*

Department of Computer and Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul, 05006, Republic of Korea

*Corresponding Author: Joobeom Yun. Email: jbyun@sejong.ac.kr

Received: 12 August 2024 Accepted: 25 October 2024 Published: 18 November 2024

ABSTRACT

Digital forensics aims to uncover evidence of cybercrimes within compromised systems. These cybercrimes are often perpetrated through the deployment of malware, which inevitably leaves discernible traces within the compromised systems. Forensic analysts are tasked with extracting and subsequently analyzing data, termed as artifacts, from these systems to gather evidence. Therefore, forensic analysts must sift through extensive datasets to isolate pertinent evidence. However, manually identifying suspicious traces among numerous artifacts is time-consuming and labor-intensive. Previous studies addressed such inefficiencies by integrating artificial intelligence (AI) technologies into digital forensics. Despite the efforts in previous studies, artifacts were analyzed without considering the nature of the data within them and failed to prove their efficiency through specific evaluations. In this study, we propose a system to prioritize suspicious artifacts from compromised systems infected with malware to facilitate efficient digital forensics. Our system introduces a double-checking method that recognizes the nature of data within target artifacts and employs algorithms ideal for anomaly detection. The key ideas of this method are: (1) prioritize suspicious artifacts and filter remaining artifacts using autoencoder and (2) further prioritize suspicious artifacts and filter remaining artifacts using logarithmic entropy. Our evaluation demonstrates that our system can identify malicious artifacts with high accuracy and that its double-checking method is more efficient than alternative approaches. Our system can significantly reduce the time required for forensic analysis and serve as a reference for future studies.

KEYWORDS

Digital forensics; autoencoder; logarithmic entropy; prioritization; anomaly detection; windows artifacts; artificial intelligence

1 Introduction

According to SonicWall's cyber threat report [1], there were 6.06 billion malware attacks in 2023. Similarly, Elastic's global threat report indicates that Windows accounted for 94% of all behavior alerts from operating system-based computing systems that year [2]. With a significant desktop operating system market share of 73% in 2023 [3], Windows is a primary target for malware attacks, making the study of Windows malware—malicious software—crucial. However, the sheer volume of malware infections makes manual investigation of a Windows system by security analysts impractical. Analyzing a system infected with malware is time-consuming due to the variety and large volume of system artifacts involved. Consequently, there is a pressing need for automated digital forensics techniques to analyze computer incidents and malware infections efficiently.

Digital forensics [4] is investigating electronic devices, such as computers and smartphones, to collect, preserve, and analyze digital data for solving crimes or other legal matters. This technique can also be used to analyze and track malicious code activities, such as how the code intruded, what directories and files were accessed, whether the backdoor was hidden, and what information was leaked. This information is usually stored in digital forensic artifacts on the system, which refer to any tangible item produced or used during the operation of a computer system [5,6]. System artifacts encompass a wide range of elements, including technical documents, code files, databases, configuration files, and system logs. Notably, given the Windows operating system's susceptibility to malware, prioritizing digital forensics for Windows is crucial. Windows system artifacts encompass system files, registry entries, event logs, prefetch files, user profiles, and application data files. These artifacts comprise valuable information related to system configurations, user activities, and system events, which can be instrumental in forensic investigations. Artifacts can be categorized into two types: *numerical data* (such as time information for processes and memory usage details) and *text data* (such as process paths and digital signatures).

Among the plethora of digital forensic artifacts, the manual identification of suspicious traces is a process, which is both time-consuming and labor-intensive. A method to surmount this inefficiency is the application of artificial intelligence (AI), which expedites the detection of malicious artifacts, thereby reducing analysis time and effort. DS4N6 [7] proposed an automated artifact analysis by applying autoencoder [8–12], deep-learning models ideal for anomaly detection, to digital forensics. Unfortunately, traditional autoencoders struggle to learn representations for text data. It has been proven that the autoencoder, which constantly reconstructs each dimension of the input vector on the same basis, is unsuitable for extremely high-dimensional and sparse text data [13]. In contrast, Cinque et al. [14], researchers in text data, utilized logarithmic entropy to filter interesting events and prioritize them from vast amounts of application logs. Entropy requires the construction of a knowledge base from normal data to define the occurrence probabilities of potential events. However, numerical data such as time information or memory usage history varies across individual systems, making knowledge base construction difficult.

We propose a system for prioritizing suspicious artifacts from compromised systems infected with malware to facilitate efficient digital forensics. Our system introduces a double-checking method that recognizes the nature of data within target artifacts and employs algorithms ideal for anomaly detection. Recognizing the challenge of accurately detecting anomalous artifacts induced by malware [15], we focus on prioritizing and recommending the analysis of suspicious artifacts. Our system follows these steps. Initially, it collects target artifacts from various sources via an artifact collection tool. Subsequently, it identifies data within the target artifacts based on user-defined features. Next, it scales numerical data and tokenizes textual data among the identified data. The steps up to this point represent the preprocessing of the target artifact. Thereafter, our system calculates the loss values for the target artifacts using an autoencoder trained on benign artifacts. Then, it retains only those artifacts that exceed the optimal loss value calculated through evaluation, filtering the remaining artifacts for subsequent prioritization. Next, our system calculates the entropy values for the target artifacts filtered from the initial prioritization using logarithmic entropy, which leverages a knowledge base containing information on benign artifacts. It then retains only those artifacts that exceed the optimal entropy value calculated through evaluation, filtering the remaining artifacts.

Finally, our system merges all the prioritization results for the target artifacts. To comprehensively evaluate incidents due to malware, we constructed a training dataset consisting of benign artifacts extracted from 2528 clean systems. Furthermore, we constructed a test dataset comprising benign and malicious artifacts extracted from 6271 compromised systems infected with various types of malware. Our evaluation demonstrates that our system can identify malicious artifacts with high accuracy and that its double-checking method is more efficient than alternative approaches.

The main contributions of our work are as follows:

1) We propose a system that combines AI and digital forensics to automatically prioritize suspicious artifacts, aiming to reduce manual effort and analysis time in digital forensics.
2) Our system introduces a double-checking method that utilizes ideal algorithms for anomaly detection, namely autoencoder and logarithmic entropy. Through experiments, we demonstrate that our system is more efficient than other approaches.
3) We constructed a training dataset comprising benign artifacts extracted from 2528 clean systems and a test dataset comprising benign and malicious artifacts extracted from 6271 compromised systems infected with various types of malware. We performed a comprehensive evaluation of intrusion incidents using all of these datasets.
4) Our evaluation demonstrates that our system can accurately identify malicious artifacts. Our system achieved an area under the curve (AUC) score of 0.96 in the experiments on malicious artifact identification using the receiver operating characteristic (ROC) curve metric. This represents about a 22% improvement over the current state-of-the-art work.

The remainder of this paper is organized as follows: Section 2 reviews existing research and compares it with our system. Section 3 introduces the proposed method and the system that adopts it. Section 4 presents the evaluation questions our system aims to address and evaluates our system in answering these questions. Section 5 discusses the limitations of our system. Finally, Section 6 summarizes the entire content of this study.

2 Related Work

Previous studies proposed systems and approaches for extracting security knowledge (e.g., IDS alerts, system and application logs, and Windows artifacts) from various data sources within critical computer systems. These studies employ filtering to identify interesting events from the security knowledge, employ prioritization to assign importance to each identified event, and selectively integrate AI to enhance efficiency in these processes. In this section, we categorize and describe diverse studies that extract security knowledge based on the form of the target or objective.

2.1 Studies on Security Events and Alerts

2.1.1 Filtering-Only Knowledge Extraction

Julisch et al. [16] proposed a data mining approach to address the challenge of efficiently handling a large number of alarms generated by intrusion detection systems (IDS). It utilizes historical alarm data to extract actionable knowledge that can aid in handling future alarms more effectively. Valeur et al. [17] proposed a comprehensive approach to intrusion detection alert correlation. The approach includes a normalization component for standardizing alert formats, a fusion component for combining alerts from different systems, and a verification component for determining the success of attacks. Adaptive learner for alert classification (ALAC) [18] is a system that reduces false positives in intrusion detection by classifying alerts using the RIPPER algorithm, a fast and effective rule learner. ALAC enables real-time alert classification and continuously updates and refines the alert classification model by integrating feedback from intrusion detection analysts. Bakar et al. [19] proposed an intrusion alert quality framework (IAQF) to address the issue of poor data quality in logs and alerts generated by security monitoring sensors such as IDS and intrusion protection systems (IPS). The framework calculates a data quality score for each parameter and extends the alert information with additional data attributes. Spathoulas et al. [20] proposed a postprocessing filter to reduce false positives in IDS. It improves the accuracy of an IDS by utilizing the statistical properties of the input alert set to differentiate between actual attacks and false positives. Cotroneo et al. [21] proposed an automated framework composed of filters and decision trees to process a large volume of security alerts and identify their underlying causes. The framework employs logarithmic entropy as term weights for alert filtering and conceptual clustering for underlying cause classification. Vaarandi et al. [22] proposed an unsupervised framework for detecting anomalous messages from syslog log files. This framework aims to address the shortcomings of existing methods that rely on labeled training datasets and manual rules generated by human experts. The framework automatically discovers event patterns from log files, leveraging data mining techniques, enabling real-time detection of anomalous messages.

2.1.2 Filtering and Prioritization Knowledge Extraction

Porras et al. [23] proposed a mission-impact-based approach to information security (INFOSEC) alarm correlation, which aims to help security analysts cope with the high volume of alerts generated by INFOSEC devices. The approach uses topology analysis to identify the critical assets and attack paths in a network, alert prioritization to rank alerts based on their potential impact on mission objectives, and common attribute-based alert aggregation to group alerts that are related to the same attack scenario. Noel et al. [24] proposed an approach for optimal IDS sensor placement and alert prioritization. This approach utilizes attack graphs to predict all potentially vulnerable paths through the network and strategically deploys IDS sensors to cover these paths with a minimum number of sensors required. Zomlot et al. [25] proposed an extended Dempster-Shafer model for prioritizing intrusion analysis and addressing fundamental issues in applying Dempster-Shafer theory [26] to intrusion analysis. By integrating their methodology with an IDS alert correlation framework, they computed a numerical confidence score for each hypothesis and ranked the outcomes accordingly. Chakir et al. [27] proposed a real-time risk assessment framework for IDS to address the challenge of managing a high volume of alerts and false positives. This framework encompasses the consolidation of alerts into meta-alerts, categorizing them, and computation of risk levels to prioritize alerts according to their assessed risk. MADE [28] is a system that identifies and prioritizes malicious activities within enterprise networks. MADE trains a machine learning model from a comprehensive set of features related to enterprise malware communication using a random forest algorithm [29], and it computes the probability that unknown domains extracted from web proxy logs are malicious.

Comparison with our study. Previous studies play a crucial role in identifying interesting events from IDS alerts and system logs generated by specific security products. However, our focus lies in assigning analysis priorities to Windows artifacts extensively collected from systems infected with malware for digital forensic investigations. Thus, these studies fall outside our research scope. Moreover, some of these studies do not employ AI, require manual effort, do not prioritize the identified events, and necessitate detailed information about the environment such as IDS sensor placement. In contrast, our study automates digital forensics using an autoencoder, an ideal deep-learning neural network for anomaly detection. It filters and prioritizes target artifacts without relying on the physical environment.

2.2 State-of-the-Art Studies

XTEC [30] acquires and analyzes artifacts from Windows event logs, prefetch, and application compatibility caches (shimcache) within the Windows system to detect malicious behaviors associated with advanced persistent threats (APTs). XTEC estimates the file execution history from these artifacts, divides the timeline into intervals at each instance of file execution, and identifies sequences of events (i.e., Windows commands) within these intervals. These sequences of events are fed into a random forest model trained on normal artifacts, and anomaly scores are calculated. DS4N6 project [7] is a study that performs anomaly detection on scheduled task artifacts using a deep-learning neural network called an autoencoder, which consists of an encoder network that reduces the dimensionality of the input data and a decoder network that reconstructs the encoded input data. This is based on the observation that the loss value increases relatively when anomalous data is given as input data, after training the autoencoder model with a real-world scheduled task artifact set consisting of scheduled task event logs and file system metadata of the scheduled task XML files. Additionally, the DS4N6 project developed a long short-term memory (LSTM) autoencoder model that integrates the LSTM architecture with the autoencoder framework to enhance the detection of malicious scheduled tasks. Cinque et al. [14] introduced a method for contextual filtering and prioritization of log data aimed at improving security situational awareness. This method utilizes a logarithmic entropy-based system to prioritize events in various air traffic control (ATC) application text logs, emphasizing their informational value to highlight significant incidents. They demonstrated that this approach could effectively filter out uninteresting events while preserving those critical for security analysts.

Comparison with our study. We have surveyed recent studies that prioritize Windows artifacts collected from systems infected with malware or employ algorithms ideal for anomaly detection. The algorithms used, along with their respective advantages and disadvantages, are summarized in Table 1.

Table 1: Comparison of state-of-the-art studies effective in anomaly detection

Refs.	Used algorithms	Advantages	Disadvantages
DS4N6 project [7]	Autoencoder & LSTM-autoencoder	Prioritization of scheduled task artifacts	Difficulty learning representations for text data
Cinque et al. [14]	Logarithmic entropy	Prioritization of ATC logs	Difficulty calculating entropy for numerical data
XTEC [30]	Random forest	APT attack detection	Limited AI features and unproven efficiency
ChatGPT4DF [31]	Generative pre-trained transformers	Assisting in the detection of deviations from typical behaviors	Phenomenon of hallucination
Real-Time [32]	YOLOv8	Real-time detection	Available in limited areas
Our system	Autoencoder & Logarithmic entropy	Double checking-based prioritization of artifacts	Performance overhead

Artifacts include numerical and text data that can aid forensic investigations. However, previous studies have not considered the nature of the data within the artifacts. DS4N6 [7] struggles to learn representations for text data. It has been proven that the autoencoder, which constantly reconstructs each dimension of the input vector on the same basis, is unsuitable for extremely high-dimensional and sparse text data [13]. Cinque et al. [14] struggled to calculate the entropy for numerical data. Entropy requires constructing a knowledge base from normal data to define the occurrence probabilities of potential events. However, numerical data such as time information or memory usage history varies across individual systems, making knowledge base construction difficult. In contrast, our system leverages the advantages of autoencoder and logarithmic entropy, which are well-established algorithms for anomaly detection. While this approach introduces some performance overhead due to the double-checking of artifacts, it is particularly valuable in digital forensics, where post-analysis is crucial. In this context, rapid real-time results are not a critical requirement, allowing for a more precise and accurate prioritization of suspicious artifacts without relying on the data they contain.

For efficient digital forensics, it is essential to demonstrate the ability to handle artifacts from systems infected with malware. However, the use of solely a single malware sample by DS4N6's [7] for evaluation limits it to exploratory research. Furthermore, the lack of detailed evaluation disclosure by XTEC [30] due to confidentiality issues results in insufficient experimental validation. Moreover, they do not acquire and analyze artifacts extensively. In contrast, we constructed a training dataset comprising benign artifacts extracted from 2528 clean systems, and a test dataset comprising benign and malicious artifacts extracted from 6271 compromised systems infected with various types of malware. Furthermore, we demonstrate that our system is more efficient than alternative approaches through a series of experiments.

2.3 Recent Digital Forensic Development

Digital forensics has evolved significantly in recent years, driven by the increasing complexity of digital devices and the growing prevalence of cybercrime. We discuss recent advancements in digital forensic technology, focusing on developments in cloud forensics, AI, and real-time forensics. Although our system does not currently include these cutting-edge technologies, we could consider applying them in future work.

- **Transformer models:** By training on large datasets, transformer models can learn what normal behavior looks like and identify deviations from this norm [31]. This can be useful in detecting unusual activities that may indicate a security breach or other malicious actions.
- **Real-time forensics:** Integrating AI and machine learning models, such as YOLOv8, has significantly enhanced real-time forensic capabilities. These models can quickly analyze large volumes of data, identify patterns, and detect anomalies in real time, aiding in the rapid identification and classification of suspicious activities [32].
- **Cloud forensics:** Researchers are proposing comprehensive frameworks and taxonomies to categorize and address the various aspects of cloud forensics. These frameworks aim to provide holistic solutions to overcome the challenges faced during forensic investigations in cloud environments [33].

Transformer models in digital forensics show promise in generating scripts for digital forensic tasks, providing explanations, and assisting in detecting deviations from typical behaviors. However, the phenomenon of 'hallucination' can produce incorrect information. Real-time forensic models can quickly analyze large volumes of data, identify patterns, and detect anomalies in real time, aiding in rapidly identifying and classifying suspicious activities. However, real-time forensic models are not diverse and only available in limited environments.

3 Methodology

The goal of this study is to design a system that prioritizes suspicious artifacts from compromised systems infected with malware, thereby facilitating efficient digital forensics. To achieve this goal, our system utilizes a double-checking method that incorporates autoencoder and logarithmic entropy algorithms, which are well-suited for anomaly detection. This method accurately identifies the nature of the data containing the target artifacts. Our system preprocesses these artifacts, prioritizing and filtering them to enhance the efficiency of the investigative processes of forensic analysts.

Components. Our system comprises several components. We summarize these components below and provide detailed explanations in the following subsections:

1) **Preprocessing of artifacts.** This component identifies data (numerical and text) contained within the target artifacts based on user-defined features. Subsequently, it scales the numerical data and tokenizes the text data for subsequent steps.
2) **Autoencoder-based prioritization.** This component calculates the loss value for target artifacts using an autoencoder trained on benign artifacts. Subsequently, it retains only those artifacts that exceed the optimal loss value determined through evaluation, while filtering the remainder.
3) **Entropy-based prioritization.** This component calculates the entropy value for target artifacts by leveraging a knowledge base containing information about benign artifacts. Subsequently, it retains only those artifacts that exceed the optimal entropy value determined through evaluation, while filtering the remainder.
4) **Merging prioritization results.** This component finally merges the prioritization results of all target artifacts.

3.1 Workflow of Our System

The workflow and overview of our system are illustrated in Fig. 1. Initially, our system utilizes a splitter to separate the data within target artifacts into two categories: numerical data and text data. The numerical data represent metrics such as process execution and termination times, memory usage, and the number of read/write operations, while the text data include the process name, path, and digital signatures. Subsequently, our system scales the numerical data using a scaler and tokenizes the text data using a tokenizer. Next, our system computes the loss values of the target artifacts using an autoencoder trained on benign artifacts. It prioritizes them in descending order of loss values using an initial prioritizer. Next, our system retains only those artifacts that exceed the optimal loss value determined through evaluation, filtering the rest for subsequent prioritization. Thereafter, our system calculates the entropy values of the filtered target artifacts using an entropy calculator that leverages a knowledge base containing information on benign artifacts. It prioritizes them in descending order of entropy values using a subsequent prioritizer. Following this, our system retains only those artifacts that exceed the optimal entropy value determined through evaluation, filtering all other artifacts. Finally, our system merges the prioritization results using a merger, resulting in the final prioritized list of artifacts.

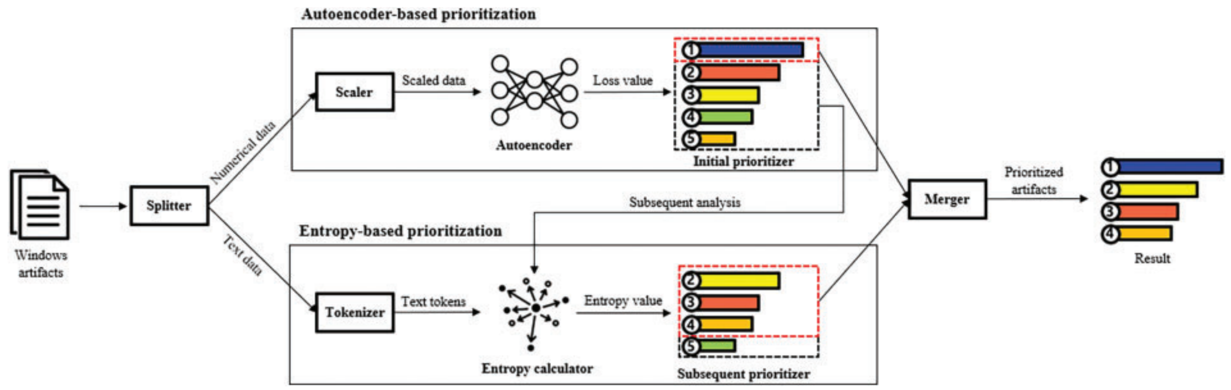

Figure 1: Workflow of our system with double-checking artifacts

3.2 Preprocessing of Artifacts

We collected artifacts from a total of eight diverse sources, namely: master file table (MFT), event logs, jumplists, prefetch files, web browser history, Windows management instrumentation (WMI), registry, and digital signatures. As delineated in Table 2, we analyze the artifacts collected from these sources and establish common features. These are all features that we can acquire from the artifacts and are useful for AI learning. We use some performance metrics as relative values such as kernel mode duration. Our system leverages these user-defined features to identify data embedded within the target artifacts. It then separates the identified data into numerical and textual data. Subsequently, our system preprocesses this numerical and text data to optimize the prioritization process for artifacts.

Table 2: Features extracted from artifacts

Property	Description
Caption	Short description of the object
DigitalSignature	Authenticity and integrity of an executable file
CSCreationClassName	Creation class name of the scoping computer system
CSName	Name of the scoping computer system
ExecutablePath	Path to the executable file of the process
KernelModeDuration	Duration time in kernel mode, in milliseconds
PageFaults	Number of page faults that a process generates
PageFileUsage	Amount of page file space that a process is using currently
ReadOperationCount	Number of read operations performed
ShimCache	Component of the application compatibility database
ThreadCount	Number of active threads in a process
UserModeDuration	Duration time in user mode, in 100 ns units
WorkingSetSize	Amount of memory in bytes that a process requires to execute efficiently
WriteOperationCount	Number of write operations performed

Our system employs an autoencoder to analyze numerical data and a logarithmic entropy to analyze text data. However, data that is not preprocessed can diminish the efficiency of artifact analysis. The range of numerical data varies in terms of minimum and maximum values and distribution patterns, which may interfere with the autoencoder model's training and loss value computation. Therefore, our system performs normalization [34–36] to transform the range of numerical data to be between 0 and 1. Consequently, normalization reduces the scale of numerical data, enabling the autoencoder model to discern the precise significance of the numerical data.

Text data, such as process paths, can comprise sentences formed by combinations of multiple words, which may interfere with the logarithmic entropy's construction of the knowledge base and computation of entropy values. Our system processes text data by tokenizing it into the smallest indivisible linguistic elements, referred to as tokens. Fig. 2 demonstrates the removal of symbols that serve merely as delimiters, particularly from text data representing process paths, followed by tokenizing strings related to these paths. This tokenization shifts the granularity of the text data from sentence level to word level, enhancing the efficiency of measurements through logarithmic entropy.

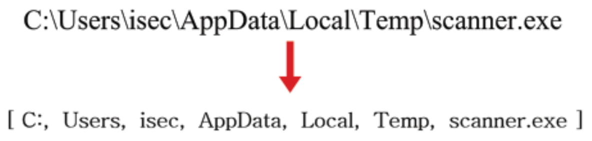
C:\Users\isec\AppData\Local\Temp\scanner.exe
[C:, Users, isec, AppData, Local, Temp, scanner.exe]

Figure 2: Tokenization for text data

3.3 Autoencoder-Based Prioritization

The autoencoder, a deep-learning neural network, comprises input, hidden, and output layers. Although the number of neurons in the input and output layers is identical, the hidden layer has fewer neurons. This implies that the original input data is compressed (encoded) to extract only the essential information and then reconstructed (decoded) from the limited information [37–40].

Inevitably, there is a discrepancy between the original input data and data reconstructed by the autoencoder, referred to as the loss value. To minimize the loss value, the autoencoder learns a compressed representation of the input data that includes only the most significant variables. Consequently, an autoencoder trained on benign artifacts struggles to reconstruct the compressed information of malicious artifacts, leading to loss of values for malicious artifacts that reach anomalous levels. This indicates that artifacts with loss values exceeding a user-defined threshold can be classified as suspicious artifacts.

We utilize the cross-entropy error to define the loss function of the autoencoder. The formula for the cross-entropy error assesses and calculates the difference between the input data and reconstructed data as follows:

$$Loss = -\sum_i p_i \log(q_i) \tag{1}$$

where i represents the number of dimensions in the data, p_i represents the distribution of the true labels, and q_i represents the distribution of the predicted values by the autoencoder.

Algorithm 1 demonstrates the entire process of autoencoder-based prioritization in our system. Initially, our system employs an autoencoder trained on benign artifacts to compress a target artifact and then reconstruct it from the compressed information (line 3). Subsequently, the system calculates the loss value between the original and reconstructed version of the target artifact (line 4). Next, it appends the pair of the target artifact and its corresponding loss value to a list (line 5). This procedure is individually applied to all target artifacts. Thereafter, the system prioritizes the list in descending order of loss values (line 7). We evaluated the optimal loss value for identifying malicious artifacts (See Section 4.3). Consequently, our system retains only those target artifacts with a loss value exceeding t (line 11), while filtering the rest for subsequent prioritization (line 13).

Algorithm 1: Autoencoder-based prioritization

```
Input: Artifact Set X
Output: Prioritized Set Y, Filtered Set Z
 1: $LossValueSet \leftarrow EmptyList$
 2: for $x \in X$ do
 3:     $y \leftarrow Autoencoder(x)$
 4:     $loss \leftarrow GetLossValue(x, y)$
 5:     $LossValueSet.append((x, loss))$
 6: end for
 7: $LossValueSet$ $in$ $descending$ $of$ $loss$
 8: $t \leftarrow$ 0.16 # SetThreshold
 9: for $(x, loss) \in LossValueSet$ do
10:     if $loss > t$ then
11:         $Y.append(x)$
12:     else
13:         $Z.append(x)$
14:     end if
15: end for
```

3.4 Entropy-Based Prioritization

Entropy is commonly employed as a measure of information uncertainty, and it has been frequently utilized in security analysis and attack detection studies [14,41–43] through the efficient processing of text data.

Inevitably, malicious artifacts exhibit sparsity when compared to benign artifacts. For instance, malicious artifacts may include suspicious process names and execution paths. Consequently, the probability of malicious artifacts originating from clean systems is low. Hence, their entropy values can be anomalous. This implies that artifacts with entropy values exceeding user-defined thresholds can be classified as suspicious.

The definition of entropy is the product of the probability of an event's occurrence and the amount of information. The amount of information is inversely proportional to the probability value, and the formula satisfying this amount of information is as follows:

$$I(x) = \log_b\left(\frac{1}{P(x)}\right) = -\log_b P(x) \tag{2}$$

where $P(x)$ represents the probability of event x occurring, and the base b of the logarithm is typically 2. As the probability of event x occurring increases, the value of $1/P(x)$ decreases, thus reducing the amount of information. Conversely, as the probability decreases, the amount of information increases.

The entropy of an event X is the sum of the products of each event x's occurrence probability and the amount of information, and its formula is as follows:

$$E(X) = \sum_{i=1}^{n} P(x_i)\left(-\log_b(P(x_i))\right) \tag{3}$$

We substitute the event's occurrence probability with the probability of the appearance of text data contained in the target artifact from the knowledge base. This knowledge base includes tokenized text data from benign artifacts. Therefore, we replace the original formula's $P(x_i)$ with the appearance probability of token j, denoted as p_j, and the formula is as follows:

$$E(T) = -\sum_{j=1}^{n} p_j \log_2 p_j \tag{4}$$

where the range of j is $1 \leq j \leq n$, and n represents the number of tokens contained within the target artifact. Specifically, p_j is the count of token j stored in the knowledge base divided by the total count of tokens stored in the knowledge base. Specifically, 1 is added to prevent errors in the logarithmic calculation.

Algorithm 2 demonstrates the entire process of entropy-based prioritization in our system. Initially, our system computes the entropy values for the target artifacts by utilizing a knowledge base that includes information on benign artifacts (Line 3). Subsequently, it appends the pair of the target artifact and its corresponding entropy value to a list (Line 4). This procedure is individually applied to all target artifacts. Thereafter, the system prioritizes the list in descending order of entropy values (Line 6). We evaluated the optimal entropy value for identifying malicious artifacts (See Section 4.3). Consequently, our system retains only those target artifacts with an entropy value exceeding t (Line 10), while filtering the rest (Line 12).

Algorithm 2: Entropy-based prioritization

```
Input: Artifact Set X, Knowledge Base K
Output: Prioritized Set Y, Filtered Set Z
 1: $EntropyValueSet \leftarrow EmptyList$
 2: for $x \in X$ do
 3:     $entropy \leftarrow GetEntropyValue(x, K)$
 4:     $EntropyValueSet.append((x, entropy))$
 5: end for
 6: $EntropyValueSet$ in descending of loss
 7: $t \leftarrow$ 0.50 # SetThreshold
 8: for $(x, entropy) \in EntropyValueSet$ do
 9: if $entropy > t$ then
10:     $Y.append(x)$
11: else
12:     $Z.append(x)$
13: end if
14: end for
```

3.5 Merging Prioritization Results

Our system aggregates the prioritization results for the target artifacts (See Sections 3.3 and 3.4). Through evaluation, we demonstrated that autoencoder-based prioritization is slightly more efficient in identifying malicious artifacts than entropy-based prioritization (See Section 4.2). Consequently, we produce the final prioritization results by merging the results of autoencoder-based prioritization followed by entropy-based prioritization.

The artifacts encompass two types of data, and our system utilized two prioritization algorithms to recognize the nature of these artifacts. This implies that an artifact deemed lower priority by the autoencoder-based prioritization can be assigned a higher priority by the entropy-based prioritization. By adopting this approach, our system can increase the true positive rate (TPR) for identifying malicious artifacts while reducing the false positive rate (FPR).

4 Evaluation

4.1 Overview

Evaluation questions. We evaluate our system by considering the following evaluation questions: (1) whether it can distinguish between malicious and benign artifacts from compromised systems; (2) what is the optimal threshold for it to identify malicious artifacts; (3) whether it outperforms alternative models in anomaly detection; and (4) whether it outperforms state-of-the-art work in terms of effectiveness.

To address these questions, we conducted a series of experiments. Initially, we evaluated whether the algorithms adopted by our system (autoencoder and logarithmic entropy) could distinguish between benign and malicious artifacts (See Section 4.2). Subsequently, we evaluated the performance of our system in identifying malicious artifacts based on various thresholds and empirically determined the optimal threshold (See Section 4.3). Furthermore, we constructed self-comparative systems by replacing our system's algorithms with other anomaly detection models and compared their performance against our system (See Section 4.4). Finally, we compared our system with state-of-the-art work on anomaly detection for artifacts (see Section 4.5).

Experimental environment. All of our experiments were conducted on widely used Windows-based systems. Despite being equipped with protective features against external threats, such as firewalls and antivirus software, these systems have become a primary target for attackers as their proliferation has led to an annual increase in security breaches. Consequently, we adopted a Windows-based system as our experimental environment to address and study these security breaches.

Experimental datasets. The dataset for our experiment is bifurcated into two main categories: a malware dataset and an artifact dataset. The malware samples are available in PE and non-PE formats. However, our study focuses exclusively on the PE format malware samples. Initially, we collected approximately 80,000 malware samples from sources including VirusTotal [44], VirusShare [45], and the Korea Internet & Security Agency (KISA) [46]. Nevertheless, certain samples were found to be non-operational due to either attempts to access defunct C2 servers or were neutralized by Windows security patches. Consequently, we utilized the ANYRUN sandbox service [47] to verify the operational status of all malware, confirming that 6271 samples were functioning correctly. This malware dataset was employed to construct compromised systems.

For the artifact dataset, we extracted benign artifacts from 2528 clean systems in Sejong University, which were systems used in labs, offices, VMs, and research labs. We verified that they were clean systems using the results of the antivirus program. The operating systems are Windows 10, 11, and Windows Server 2019. The artifacts extracted from clean systems were utilized as a training dataset for our system's core algorithms. In contrast, benign and malicious artifacts from 6271 compromised systems infected with various types of malware were utilized as a test dataset for multiple experiments conducted on our system. We infected malware samples on reverted clean VMs to generate the compromised systems, so the compromised system was infected by one malware sample. In summary, we used various clean systems for training and 6271 compromised VMs for testing.

Fig. 3 illustrates the process of extracting artifacts from compromised systems and identifying data from the extracted artifacts. Initially, we utilized a virtual machine to establish a Windows-based system. Next, we copy and execute malware on the virtual machine. We then wait until the virtual machine is sufficiently infected by the malware to become a compromised system. Next, we execute our proprietary artifact extraction tool to acquire artifacts from various sources. Next, we identify the data contained within the extracted artifacts based on the features established in Table 2 and parse the identified data into JSON format. As exemplified in Table 3, these JSON data outputs include features and values for each artifact. The values in Table 3, comprising numerical or text data, will be preprocessed to optimize the process of artifact prioritization (See Section 3.2). Finally, the JSON data outputs are stored in our database.

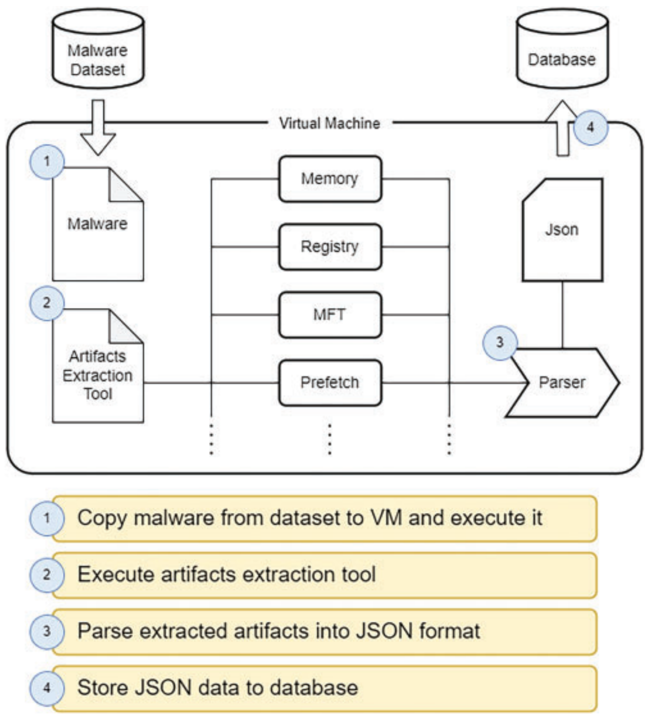
Malware Dataset
Database
Virtual Machine
1
Malware
4
Memory
Registry
Json
2
MFT
3
Artifacts Extraction Tool
Prefetch
Parser
1 Copy malware from dataset to VM and execute it
2 Execute artifacts extraction tool
3 Parse extracted artifacts into JSON format
4 Store JSON data to database

Figure 3: The process of extracting artifacts from compromised systems

Table 3: Example of identified features and values from the artifact

Features	Values
Caption	"executable.exe"
CreationDate	20220414121210 2.185014
CreationClassName	"Win32_Process"
ExecutablePath	"C:\Users\username\AppData\Local\Temp\executable.exe"
PageFileUsage	91824

4.2 Anomaly Detection in Artifacts

Experimental setup. This experiment aims to validate the hypothesis adopted by our system, which posits that artifacts extracted from compromised systems exhibit higher anomaly values when compared to those extracted from a clean system. Consequently, our system prioritizes the artifacts extracted from the compromised and clean systems and then compares the loss and entropy values among the top 25 artifacts with the highest priority. Our system employs an autoencoder to calculate the loss values and logarithmic entropy to calculate the entropy values.

Experimental results. Figs. 4 and 5 present graphs comparing the anomaly values between artifacts extracted from compromised systems and a clean system. The graphs illustrate that the red lines, representing the compromised systems, are positioned relatively higher than the blue lines of the clean system. This represents the efficacy of our system's adopted autoencoder and logarithmic entropy in anomaly detection for suspicious artifacts. As shown in Fig. 4, the loss values between the top prioritized artifacts from compromised and clean systems differ by up to 52%, with an average difference of 40%. Similarly, as depicted in Fig. 5, the entropy values differ by up to 25%, with an average difference of 22%. An interesting finding from these results is that the autoencoder exhibits greater efficiency in anomaly detection than logarithmic entropy. Consequently, our system employs the autoencoder for initial prioritization, followed by logarithmic entropy for subsequent prioritization. The reason for using these two methods is that they work complementarily, so anomalies that one method cannot detect can be detected by the other method.

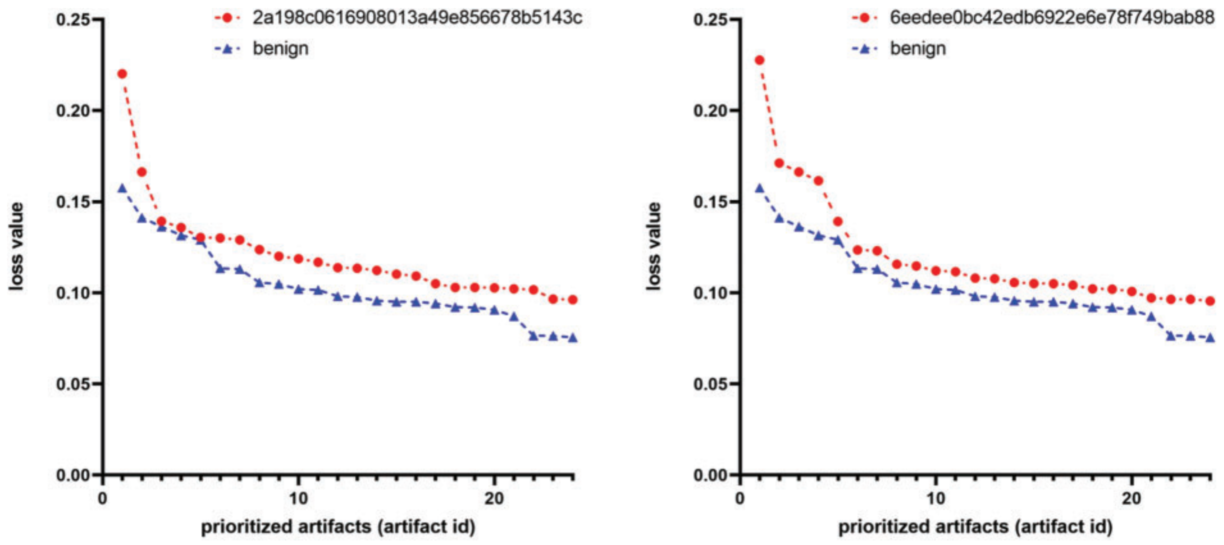

Figure 4: (Continued)

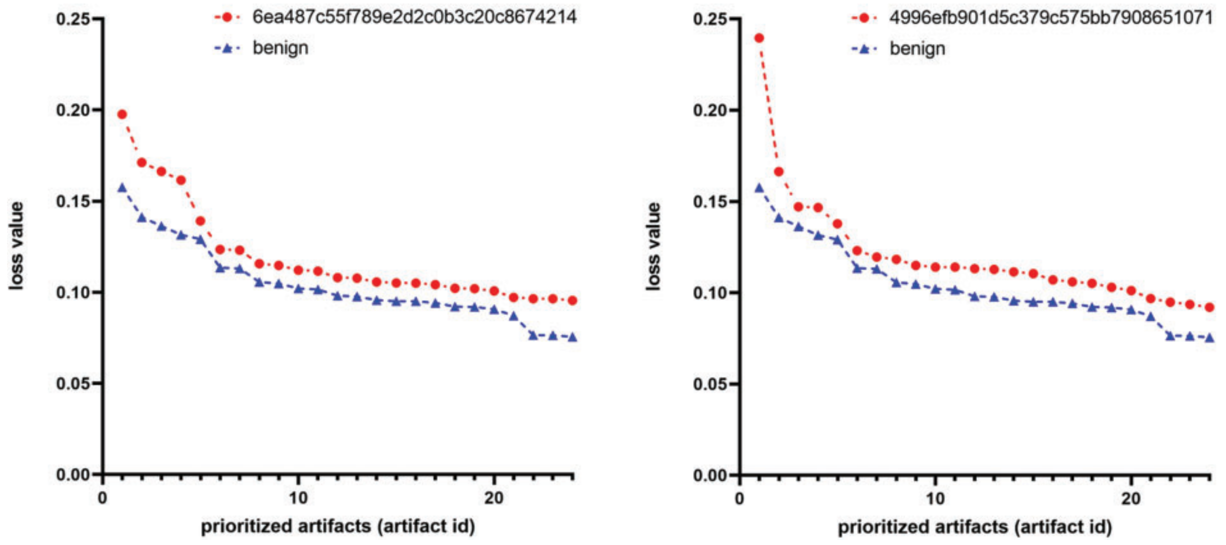

Figure 4: Comparison of loss values in artifacts from compromised *vs*. clean systems

4.3 Optimal Threshold Selection

Experimental setup. This experiment aims to calculate the optimal threshold for identifying malicious artifacts in our system. These thresholds serve to prioritize suspicious artifacts while filtering the rest. This implies that incorrect thresholds can inadvertently filter out malicious artifacts. Therefore, we must compute the optimal threshold for anomaly detection algorithms via empirical evaluation to guide the best prioritization outcomes. We consider identification successful if artifacts exceeding the threshold are malicious. For instance, if the threshold for the loss value is set at 0.1, and an artifact exhibits a loss value exceeding this threshold, it is deemed true if the artifact is malicious and false otherwise. Consequently, our system utilizes a test dataset comprising both benign and malicious artifacts to prioritize them. Hence, the optimal thresholds for loss and entropy values, which enable the identification of malicious artifacts, are selected.

Experimental results. Table 4 presents the ratio of TPR and FPR for artifacts exceeding various thresholds. We provide Table 4 to select the optimal threshold value for each of the two combined methods (i.e., autoencoder-based and entropy-based prioritization). TPR represents the proportion of malicious artifacts, while FPR represents the proportion of benign artifacts. For instance, when the threshold of loss value is set at 0.18, the TPR is 66.5%, and the FPR is 68.3%. Ideally, effective thresholds for identifying malicious artifacts prioritize high TPR and low FPR. As thresholds decrease (allowing more artifacts), TPR increases, but relatively, FPR also rises. Therefore, we aim to achieve a TPR of at least 90% while minimizing FPR. Consequently, our system uses a threshold of 0.16 for the loss value of autoencoder-based prioritization and a threshold of 0.50 for the entropy value of entropy-based prioritization.

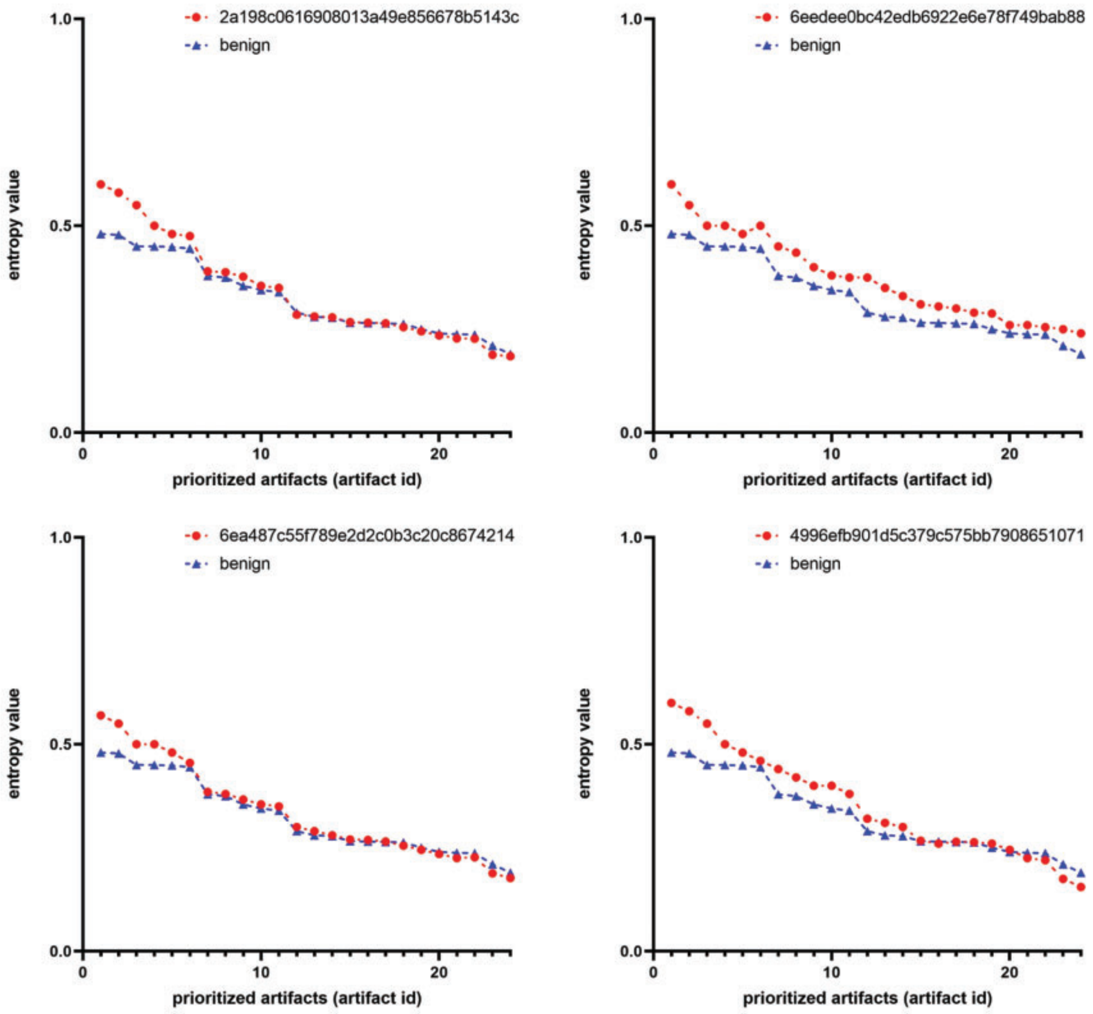

Figure 5: Comparison of entropy values in artifacts from compromised *vs.* clean systems

Table 4: Optimal threshold selection for malicious artifact identification

Threshold		TPR (%)	FPR (%)
Loss value	0.18	66.5	68.3
	0.16	**90.4**	**65.6**
	0.14	90.8	71.6
	0.12	92.0	77.5
Entropy value	0.52	73.3	75.8
	0.50	**90.1**	**68.2**
	0.48	91.4	71.3
	0.46	93.5	78.9

4.4 Comparison with Anomaly Detection Systems

Experimental setup. This experiment aims to compare the performance of our system's adopted algorithms with other anomaly detection algorithms. To achieve this, we constructed four self-comparison systems by replacing only the anomaly detection algorithms while keeping our system as the base (i.e., four systems used the same features from Table 2). These systems utilize the following anomaly detection algorithms: autoencoder (AE), principal component analysis (PCA), isolation forest (IF), and one-class SVM (OCSVM). For a comparative evaluation between our system and self-comparison systems, we employ ROC curves. The ROC curve represents the ratio of FPR to TPR for all possible thresholds, providing a graphical representation of the classification model's performance. The AUC is used to evaluate the performance of the classification model, where a higher AUC score indicates better classification accuracy. We set the threshold range for the ROC curve from the top 1% to the top 30% of prioritized artifacts. When artifacts fall within this threshold range and are malicious, the TPR increases, while for benign artifacts, the FPR increases.

Experimental results. Fig. 6 depicts the ROC curves for all systems utilized in the experiment. The ROC curve graph indicates that higher TPR and lower FPR are maintained when the curve is closer to the upper left corner. Fig. 6 demonstrates that our system's graph closely aligns with the upper left corner, and the AUC score for our system, representing the lower region of the graph, is 0.96, which is higher than other systems. For instance, our system achieves approximately 210% higher AUC score than the system using OCSVM. An interesting point from Fig. 6 is that a system using only AE achieves an AUC score of 0.95, suggesting that AE is an ideal algorithm for anomaly detection. However, it does not handle text data contained in artifacts, so targeting broader datasets may result in lower AUC scores. In contrast, our system recognizes numerical and text data contained within the artifacts, handling numerical data with AE and text data with logarithmic entropy.

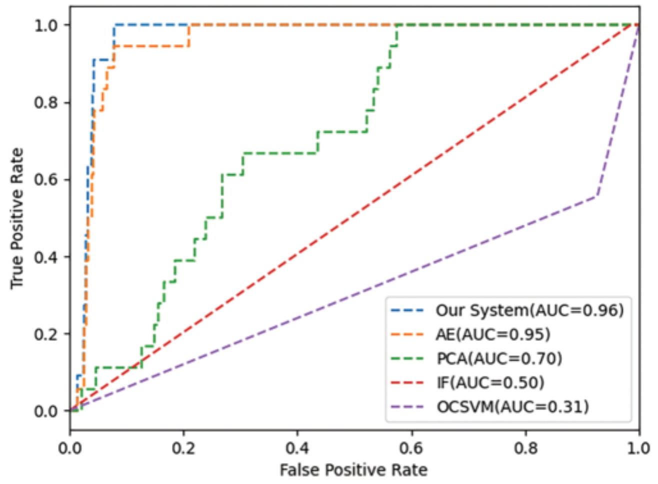

Figure 6: Comparison with anomaly detection systems using ROC curve

4.5 Comparison with State-of-the-Art Work

Experimental setup. This experiment aims to compare the performance of our system with that of state-of-the-art work. As previously mentioned, DS4N6 [7] is a study that employs an autoencoder for anomaly detection in Windows artifacts (See Section 2.2). This makes it a suitable benchmark for comparison with our system due to the similarity in objectives and methods of our research.

Consequently, we selected the DS4N6 as the benchmark for state-of-the-art work. We utilize the ROC curve to measure their AUC scores. The methodology of this experiment is conducted in the same manner as described in Section 4.4. Additionally, we further measure the distribution of malicious artifacts prioritized by each system.

Experimental results. Fig. 7 depicts the ROC curves for all systems utilized in the experiment. The AUC score for DS4N6 is 0.79, which is higher than the system using PCA shown in Fig. 6, but lower than our system. Our system achieves an AUC score of 0.96, approximately 22% higher than DS4N6. These results indicate that DS4N6 is limited to handling only scheduled task artifacts and cannot manage artifacts collected from various sources.

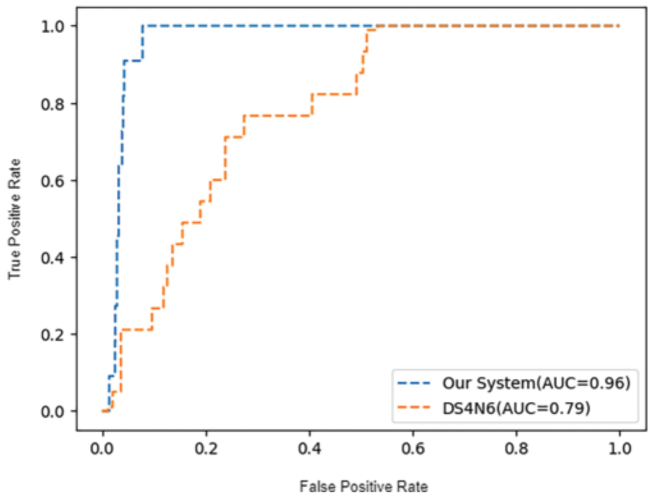

Figure 7: Comparison with DS4N6 using ROC curve

Fig. 8 depicts the distribution of malicious artifacts across all systems used. This distribution indicates the average percentile ranking of prioritized malicious artifacts within the total prioritized artifacts. For the artifacts prioritized by DS4N6, malicious artifacts are distributed in the top 12.3%. For the artifacts prioritized by our system, malicious artifacts are distributed in the top 2.6%. The distribution of these malicious artifacts in the top percentile improves the efficiency of forensic analysis. For instance, consider a forensic analyst sequentially analyzing 1000 prioritized artifacts using these systems. With DS4N6, a minimum of 123 artifacts would require manual analysis. With our system, a minimum of 26 artifacts would require manual analysis. Consequently, our system demonstrates greater efficiency in prioritizing malicious artifacts than DS4N6.

4.6 Complexity

Autoencoders have several key hyperparameters that influence their complexity. The number of hidden layers determines the network's depth and ability to capture complex patterns. The number of neurons in each layer affects the network's capacity for data representation. The size of latent space balances model complexity and performance. The activation function is crucial for the network's nonlinearity and learning ability. The objective function measures the difference between input and output data. The optimization algorithm minimizes the objective function during training. The learning rate dictates the step size during optimization. The number of epochs represents full dataset passes during training. Batch size affects gradient noise and optimization efficiency [48].

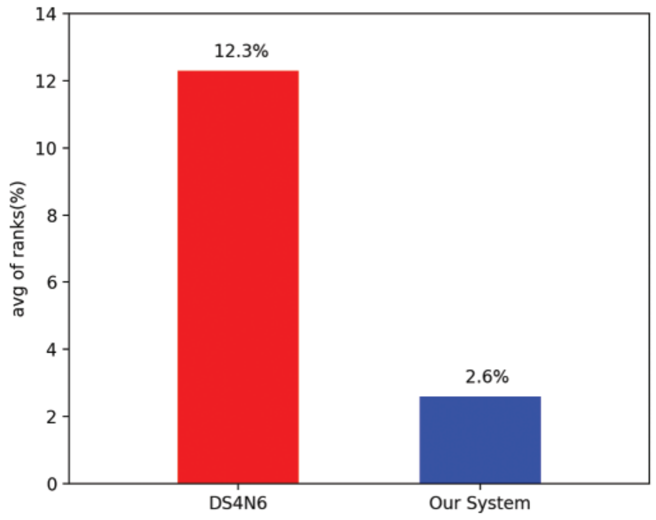

Figure 8: Comparative distribution of malicious artifacts between DS4N6 and our system

Logarithmic entropy involves calculating the entropy of a given dataset and then applying a logarithmic transformation. This calculation measures the uncertainty or randomness in the dataset [49]. The computational complexity of calculating logarithmic entropy depends on the size and distribution of the dataset. Larger datasets require more computational resources to estimate the probabilities and the entropy. However, the real challenge comes with high-dimensional data, which can significantly increase the computational complexity, making it a formidable task [50].

5 Discussion and Limitation

In this section, we discuss the limitations of our system for automatically prioritizing suspicious artifacts and directions for future research.

1) As mentioned in Section 3.2, our system collects artifacts from eight diverse sources within the Windows environment, including MFT, event logs, jumplists, prefetch files, web browser history, WMI, registry, and digital signatures. However, artifacts can also be automatically generated from various sources beyond these. For example, Device Guard, generates logs of application execution policies, and Swapfile.sys, maintains a record of recently executed files. Given that the accessibility and architecture of artifacts are subject to alteration due to updates in the operating system and shifts in security protocols, ongoing research and updated expertise are imperative for effective digital forensic analysis. In the future, our research will aim to expand the scope of artifact collection by integrating additional sources and developing algorithms capable of adapting to changes in artifact generation patterns.
2) We conducted a study focused on acquiring and analyzing artifacts from Windows systems. However, Windows is just one of several operating systems. Other operating systems, such as Linux, Mac OS, and Android, also exist, each managing systems and generating logs in unique ways. For instance, Linux primarily stores logs in text file format within the/var/log directory, while Mac OS uses a Unix-based logging system. Furthermore, Android, although based on the Linux kernel, employs a logging system specialized for mobile environments. In the future, our research will aim to delve into the intricacies of artifact analysis across different operating systems. Recognizing the distinct methods by which systems such as Linux, Mac OS,

and Android manage and generate logs, we plan to develop a comprehensive understanding encompassing these diverse environments. This endeavor will involve a comparative study of the artifact generation processes and the creation of sophisticated tools designed to cater to the unique forensic requirements of each operating system. Based on this approach, we aspire to enhance the efficacy of forensic investigations in a multi-OS landscape.

6 Conclusion

This study presents a novel system integrating artificial intelligence with digital forensics to prioritize suspicious artifacts in compromised systems. Our approach addresses the inefficiencies associated with manual investigation by leveraging a double-checking method that combines autoencoder and logarithmic entropy algorithm for anomaly detection.

The proposed system significantly improves the accuracy and efficiency of identifying malicious artifacts. We have shown that our system can effectively distinguish between benign and malicious artifacts through comprehensive evaluations using extensive datasets, achieving an AUC score of 0.96. This represents a substantial enhancement over existing methods, highlighting the potential of our approach to streamline digital forensic investigations. In this paper, we propose a methodology to assist in the forensic process on the Windows system. Still, this methodology can also be generalized and applied to macOS and UNIX/Linux systems through feature extraction, model training, and parameter tuning.

Our contributions include the development of a robust training dataset from clean systems and a diverse test dataset from compromised systems, which have been instrumental in validating the effectiveness of our system. The double-checking method reduces the time and effort required for analysis and ensures a higher degree of accuracy in detecting anomalies.

In conclusion, our system offers a promising solution for the automated analysis of digital forensic artifacts, paving the way for more efficient and accurate investigations of malware infections. We expect our system to be useful for post-mortem forensics, but its performance overhead makes it difficult to use in the live endpoint detection and response (EDR) system. Future work will focus on further refining the algorithms and expanding the system's capabilities to handle various digital forensic challenges.

Acknowledgement: We extend our heartfelt gratitude to the anonymous reviewers whose insightful feedback and constructive criticism significantly enhanced the quality of this manuscript. Their expertise and attention to detail were invaluable in refining our work.

Funding Statement: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2024-00437494) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Juhwan Kim, Baehoon Son; data collection: Juhwan Kim, Baehoon Son; analysis and interpretation of results: Juhwan Kim; draft manuscript preparation: Juhwan Kim; manuscript final layout and preparation for submission: Jihyeon Yu, Joobeom Yun. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Due to the potential for misuse, our malware dataset and artifact dataset are not publicly available.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Sonicwall, "Sonicwall cyber threat report (navigating the relentless surge in cybercrime)," 2024. Accessed: Feb. 1, 2024. [Online]. Available: https://www.sonicwall.com/medialibrary/en/white-paper/2024-cyber-threat-report.pdf

[2] Elastic security lab, "Elastic global threat report,' 2023. Accessed: Nov. 1, 2023. [Online]. Available: https://www.elastic.co/pdf/elastic-global-threat-report-october-2023.pdf

[3] Statcounter, "Desktop operating system market share world," 2024. Accessed: Feb. 1, 2024. [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide

[4] S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digit. Invest.*, vol. 7, no. 3, pp. S64–S73, 2010. doi: 10.1016/j.diin.2010.05.009.

[5] V. S. Harichandran, D. Walnycky, I. Baggili, and F. Breitinger, "CuFA: A more formal definition for digital forensic artifacts," *Digit. Invest.*, vol. 18, no. Suppl. 2, pp. S125–S137, 2016. doi: 10.1016/j.diin.2016.04.005.

[6] D. M. Purcell and S. Lang, "Forensic artifacts of microsoft windows vista system," in *Proc. IEEE ISI*, Taipei, Taiwan, Springer, 2008, pp. 304–319.

[7] J. Garcia, "Detecting the solarwinds malicious scheduled task with an autoencoder," Accessed: Mar. 16, 2022. [Online]. Available: https://www.ds4n6.io/blog.html

[8] M. Catillo, A. Pecchia, and U. Villano, "AutoLog: Anomaly detection by deep autoencoding of system logs," *Expert. Syst. Appl.*, vol. 191, no. 5, 2022, Art. no. 116263. doi: 10.1016/j.eswa.2021.116263.

[9] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A malware detection approach using autoencoder in deep learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022. doi: 10.1109/ACCESS.2022.3155695.

[10] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *Proc. Wireless Telecommun. Symp. (WTS 2018)*, IEEE, 2018, pp. 1–5.

[11] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Proc. 18th Iberoamerican Congress Pattern Recognit.*, Havana, Cuba, Springer, 2013, pp. 117–124.

[12] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Dis. Data Min. (KDD '17)*, Halifax, NS, Canada, 2017, pp. 665–674.

[13] S. Zhai and Z. Zhang, "Semisupervised autoencoder for sentiment analysis," in *Proc. AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, vol. 30, no. 1, pp. 1394–1400. doi: 10.1609/aaai.v30i1.10159.

[14] M. Cinque, R. D. Corte, and A. Pecchia, "Contextual filtering and prioritization of computer application logs for security situational awareness," *Future Gener. Comput. Syst.*, vol. 111, no. 5, pp. 668–680, 2020. doi: 10.1016/j.future.2019.09.005.

[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009. doi: 10.1145/1541880.1541882.

[16] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Dis. Data Min. (KDD'02)*, Edmonton, AB, Canada, 2002, pp. 366–375.

[17] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 3, pp. 146–169, 2004. doi: 10.1109/TDSC.2004.21.

[18] T. Pietraszek, "Using adaptive alert classification to reduce false positives in intrusion detection," in *Proc. 7th Symp. Recent Adv. Intrus. Detect. (RAID 2004)*, Sophia-Antipolis, France, Springer, 2004, pp. 102–124.

[19] N. A. Bakar, B. Belaton, and A. Samsudin, "False positives reduction via intrusion alert quality framework," in *Proc. 13th IEEE Int. Conf. Netw. Jointly Held 7th IEEE Malay. Int. Conf. Commun.*, Kuala Lumpur, Malaysia, IEEE, 2005, pp. 547–552.

[20] G. P. Spathoulas and S. K. Katsikas, "Reducing false positives in intrusion detection systems," *Comput. Secur.*, vol. 29, no. 1, pp. 35–44, 2010. doi: 10.1016/j.cose.2009.07.008.

[21] D. Cotroneo, A. Paudice, and A. Pecchia, "Automated root cause identification of security alerts: Evaluation in a saas cloud," *Future Gener. Comput. Syst.*, vol. 56, no. 4, pp. 375–387, 2016. doi: 10.1016/j.future.2015.09.009.

[22] R. Vaarandi, B. Blumbergs, and M. Kont, "An unsupervised framework for detecting anomalous messages from syslog log files," in *Proc. NOMS 2018–2018 IEEE/IFIP Netw. Operat. Manag. Symp.*, Taipei, Taiwan, IEEE, 2018, pp. 1–6.

[23] P. A. Porras, M. W. Fong, and A. Valdes, "A mission-impact-based approach to infosec alarm correlation," in *Proc. Int. Workshop Recent Adv. Intrus. Detect. (RAID 2002)*, Zurich, Switzerland, Springer, 2002, pp. 95–114.

[24] S. Noel and S. Jajodia, "Optimal ids sensor placement and alert prioritization using attack graphs," *J. Netw. Syst. Manag.*, vol. 16, no. 3, pp. 259–275, 2008. doi: 10.1007/s10922-008-9109-x.

[25] L. Zomlot, S. C. Sundaramurthy, K. Luo, X. Ou, and S. R. Rajagopalan, "Prioritizing intrusion analysis using dempster-shafer theory," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, Chicago, IL, USA, 2011, pp. 59–70.

[26] G. Shafer, *A mathematical theory of evidence*. Princeton, NJ, USA: Princeton University Press, 1976.

[27] E. M. Chakir, M. Moughit, and Y. I. Khamlichi, "Risk assessment and alert prioritization for intrusion detection systems," in *Proc. 3rd Int. Symp. Ubiquit. Network. (Unet 2017)*, Casablanca, Morocco, Springer, 2017, pp. 641–655.

[28] A. Oprea, Z. Li, R. Norris, and K. Bowers, "Made: Security analytics for enterprise threat detection," in *Proc. 34th Annu. Comput. Secur. App. Conf. (ACSAC'18)*, San Juan, PR, USA, 2018, pp. 124–136.

[29] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/A:1010933404324.

[30] S. P. Liew and S. Ikeda, "Detecting adversary using windows digital artifacts," in *Proc. 2019 IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, 2019, pp. 3210–3215.

[31] M. Scanlon, F. Breitinger, C. Hargreaves, J. Hilgert, and J. Sheppard, "ChatGPT for digital forensic investigation: The good, the bad, and the unknown," *Forens. Sci. Int.: Dig. Invest.*, vol. 46, 2023, Art. no. 301609.

[32] S. Karakuş, M. Kaya, and S. A. Tuncer, "Real-time detection and identification of suspects in forensic imagery using advanced YOLOv8 object recognition models," *Traitement du Signal*, vol. 40, no. 5, pp. 2029–2039, 2023. doi: 10.18280/ts.400521.

[33] P. Purnaye and V. Kulkarni, "Comprehensive study of cloud forensics," *Arch. Computat. Methods Eng.*, vol. 29, no. 1, pp. 33–46, 2022. doi: 10.1007/s11831-021-09575-w.

[34] B. Li, F. Wu, S. Lim, S. Belongie, and K. Q. Weinberger, "On feature normalization and data augmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR 2021)*, 2021, pp. 12383–12392.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML 2015)*, Lille, France, 2015, pp. 448–456.

[36] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. 30th Int. Conf. Neur. Inf. Process. Syst. (NIPS'16)*, Barcelona, Spain, 2016, pp. 901–909.

[37] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, no. 4, pp. 232–242, 2016. doi: 10.1016/j.neucom.2015.08.104.

[38] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.

[39] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, no. 1, pp. 19–31, 2016. doi: 10.1016/j.jnca.2015.11.016.

[40] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Dis. Data*, vol. 6, no. 1, pp. 1–39, 2012. doi: 10.1145/2133360.2133363.

[41] J. Cao, B. Yu, F. Dong, X. Zhu, and S. Xu, "Entropy-based denial-of-service attack detection in cloud data center," *Concurr. Comput.: Pract. Exp.*, vol. 27, no. 18, pp. 5623–5639, 2015. doi: 10.1002/cpe.3590.

[42] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 412–425, 2010. doi: 10.1109/TPDS.2010.97.

[43] K. Hong, C. Chen, Y. Chiu, and K. Chou, "Scalable command and control detection in log data through UF-ICF analysis," in *Proc. 2015 Int. Carnahan Conf. Secur. Technol. (ICCST)*, 2015, pp. 293–298.

[44] Virustotal, "Analyse suspicious files, domains, IPs and URLs to detect malware and other breaches, automatically share them with the security community," 2022. Accessed: Jun. 23, 2022. [Online]. Available: https://www.virustotal.com/

[45] VirusShare, "A repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code," 2022. Accessed: Jun. 23, 2022. [Online]. Available: https://virusshare.com/

[46] KISA, "Korea internet & security agency," 2021. Accessed: May 15, 2021. [Online]. Available: https://www.kisa.or.kr/

[47] Anyrun, "Interactive malware hunting service," 2023. Accessed: Mar. 15, 2023. [Online]. Available: https://any.run/

[48] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: A survey," *Artif. Intell. Rev.*, vol. 57, no. 28, pp. 1–52, 2024. doi: 10.1007/s10462-023-10662-6.

[49] M. K. Siddiqui, M. Hussain, S. Javed, S. Khalid, T. Noor and F. T. Tolasa, "On characterization of entropy measure using logarithmic regression model for Copper (II) Fluoride," *PLoS One*, vol. 19, no. 3, pp. 1–34, 2024. doi: 10.1371/journal.pone.0300757.

[50] C. Feng, M. F. Hanif, M. K. Siddiqui, M. Hussain, and N. Hussain, "On analysis of entropy measure via logarithmic regression model for 2D-honeycomb networks," *Eur. Phys. J. Plus.*, vol. 138, no. 10, 2023, Art. no. 924. doi: 10.1140/epjp/s13360-023-04547-4.