



Contents lists available at ScienceDirect

## Forensic Science International: Digital Investigation

journal homepage: [www.elsevier.com/locate/fsidi](http://www.elsevier.com/locate/fsidi)

DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA

## volGPT: Evaluation on triaging ransomware process in memory forensics with Large Language Model

Dong Bin Oh<sup>a</sup>, Donghyun Kim<sup>b</sup>, Donghyun Kim<sup>c</sup>, Huy Kang Kim<sup>a,\*</sup><sup>a</sup> School of Cybersecurity, Korea University, Anam-ro 145, Seoul, 02841, Republic of Korea<sup>b</sup> Department of Cyber Security, Ajou University, World Cup-ro 206, Suwon, 16499, Republic of Korea<sup>c</sup> Department of Computer Science, Korea Military Academy, Hwarang-ro 574, Seoul, 01805, Republic of Korea

## ARTICLE INFO

## Keywords:

digital forensics  
 Memory forensics  
 Ransomware  
 Volatility  
 ChatGPT  
 Large language model (LLM)

## ABSTRACT

In the face of the harm that ransomware can inflict upon users' computers, the imperative to efficiently and accurately triage its processes within memory forensics becomes increasingly crucial. However, ransomware perpetrators employ sophisticated techniques, such as process masquerading, to evade detection and analysis. In response to these challenges, we propose a novel ransomware triage method leveraging a Large Language Model (LLM) in conjunction with the Volatility framework, the de-facto standard in memory forensics. We conducted experiments on memory dumps infected by five different ransomware families, utilizing LLM-based approaches. Through extensive experiments, our method named volGPT demonstrated high accuracy in identifying ransomware-related processes within memory dumps. Additionally, our approach exhibited greater efficiency and provided more comprehensive explanations during ransomware triage than other state-of-the-art methods.

## 1. Introduction

In the field of digital forensics, memory forensics is a technique that enables the extraction of volatile information in memory which is not acquirable through disk analysis. By analyzing memory dumps, it becomes possible to gain insights into the behavior of malware and identify traces of malware that may unnoticed through conventional disk based digital forensics. To conduct these tasks, the Volatility framework has been developed as a widely adopted and utilized memory forensics tool. Moreover, ongoing research in memory forensics focuses on various approaches, including the analyzing of memory artifacts associated with fileless malware, anti-forensic measures, APT malware, and automation, contributing to the continuous advancement of techniques in response to evolving cybersecurity challenges (Case and Richard, 2017).

## 1.1. Motivation

Through memory forensics, we can extract process names, process relationships, and network details from kernel structures. Based on extracted information, it is necessary to examine and interpret the relationships and actual behaviors of processes. Through this procedure,

we can extract suspicious memory forensic artifact which related to malware. Triage is the task that distinguishes the target for analysis from this vast amount of information. From the perspective of initial response in memory forensics, triage requires significant resources, especially in terms of domain knowledge and time.

We propose leveraging Large Language Models (LLM) as one of the solutions to address the challenges of triage. Large language models, which are trained on extensive text corpus, enables to perform various Natural Language Processing (NLP) tasks. In 2020, BERT, one of the large language model started emerging (Brown et al., 2020), and by 2023, developers introduced interactive conversational models named ChatGPT, which the general public could access. Furthermore, leveraging LLMs, optimization tailored to various fields is carried out, enabling the generation of expert-level responses, for instance, BloombergGPT (Wu et al., 2023) and MathGPT (Scarlatos and Lan, 2023).

By harnessing the capabilities of LLMs, we aim to enhance the efficiency and accuracy of triage in the context of memory forensics. Recently, there are many researches has been underway to explore the applicability of services based on large language models in the field of digital forensics. Some studies examine the potential of applying ChatGPT, a representative LLM-based conversational AI service, to the field of digital forensics like (Scanlon et al., 2023). This research

\* Corresponding author.

E-mail address: [cenda@korea.ac.kr](mailto:cenda@korea.ac.kr) (H.K. Kim).<https://doi.org/10.1016/j.fsidi.2024.301756>

Available online 5 July 2024

2666-2817/© 2024 The Author(s). Published by Elsevier Ltd on behalf of DFRWS. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

demonstrates that ChatGPT can play a supportive role for investigators in interpreting digital forensics data.

In this study, we focus on the triage of ransomware memory forensics. Numerous types of malware can be investigated using memory forensics. But ransomware has recently emerged as a particularly serious issue. Especially with certain ransomware, malware analysis may only be possible through the utilization of memory forensics, as they employ techniques like process masquerading, process hiding, and process injection. Ransomware poses a threat not only to victims but also to analysts, necessitating effective and efficient analysis.

## 1.2. Contribution

The contributions of this paper are outlined as below:

- Proposing volGPT, the first prompt-based large language model for a memory forensics, providing analysts with triage automation and explanations for triage reasons.
- volGPT demonstrates the capability to detect processes suspected of malicious behavior with an accuracy of around 94 %, relying on the relationships among processes and their names.
- We found the efficiency for analysts by utilizing LLM-based volGPT, which reduces the number of suspicious processes within memory dumps by around 10 %.

The structure of the paper is as follows: In Section 2, we describe memory forensic method for Windows systems using the Volatility framework. Section 3 explains the design and procedure of volGPT, utilizing GPT in a prompt-based approach. In Section 4, we evaluate the performance of volGPT and examine the reasoning behind volGPT's responses. Section 5 discusses the limitations of our study and areas for improvement based on the experiment results. In Section 6, we review related state-of-the-arts in the context of memory forensics automation and LLM-based decision-making systems. Finally, Section 7 concludes the paper and discusses about future works.

## 2. Windows malware triage with volatility

Triage in memory forensics serves the crucial purpose of rapidly assessing and prioritizing extensive memory artifacts before malware analysis. Triage aims to swiftly identify significant information including malicious code, processes, or other threats, allowing investigators to efficiently allocate resources. By automating and streamlining the initial examination of memory dump, triage facilitates the quick identification of key insights, enabling investigators to focus on pertinent evidence and optimize their efforts within the constraints of time and resources. Volatility is an open-source framework that can be used as de-facto standard in memory forensics. It is implemented in Python under the GNU General Public License and supports memory dumps from 32 to 64-bit Windows operating systems and service packs, also Linux and Mac OS X. With Volatility, we can perform investigation on process lists, ports, network connections, registry, DLLs, crash dumps, and cached sectors within memory dump (Ligh et al., 2014a,b).

### 2.1. Process analysis

To understand the structure of Windows processes in memory dumps, Volatility uses its own plugins to identify and extract information. The *pslist* plugin can be used to identify active processes and list them as table form. It shows information about the process name, process ID (PID), parent process ID (PPID), base address, and when the process started and exited. Although it cannot detect unlinked or hidden processes (Ligh et al., 2014a,b), the *pscan* can. This plugin involves exploring unique pool tags associated with pool objects. By searching for these pool tags, it is possible to uncover hidden, terminated, or returned EPROCESS objects, allowing for the retrieval and examination of process

information (Schuster, 2006). Also, the *psxview* plugin is useful in identifying hidden processes, rootkits, and hooks. It can be used to analyze Windows processes in memory dumps by comparing with other plugin results (Ligh et al., 2014a,b).

### 2.2. VAD analysis

Virtual Address Descriptor (VAD) is a tree structure that contains all the information related to memory region reserved for a specific process by the memory manager. For each chunk of continuous virtual memory address allocations, the memory manager creates a corresponding VAD node that contains information about this continuous allocation and other subnodes (Yosifovich et al., 2017). The *vadinfo* plugin is used to display information about a process's virtual address descriptor nodes. It shows the address of the MMVAD (Memory Mapped VAD) structure, the start and end addresses of the VAD node, the tag, flags, and protection of the VAD node (Dolan-Gavitt, 2007). This plugin is useful for analyzing Windows processes as it can provide detailed information about mapped files or loaded DLL, specific permission about VAD area. Also The *malfind* leverages VAD information to find anomalies like injected code or hidden processes by examining memory regions that should not typically contain executable code, based on the VAD protection flags and other characteristics.

### 2.3. Windows system artifact analysis

The *hivescan* plugin allows analysis of the registry mapped in memory. The registry stores content in a key-value format, containing information about system user settings, system configuration (Saidi et al., 2013). Additionally, the *evtlog* plugin facilitates the analysis of the Windows event log in memory. Investigating event logs allows the detection of various events occurring within the Windows system, providing insights into the propagation and persistence behaviors of malwares (Kang et al., 2018).

The *cmdline* plugin extracts process parameters for executable files executed through the *cmd.exe* present in memory. This plugin enables the identification not only of the name of the malicious code executed by the attacker but also the specific behaviors undertaken by the malware based on the conditions set for its execution. The *notepad* plugin focuses on the extraction of information about text content written or viewed by users through *notepad.exe*, a default application in Windows. The notepad plugin can reveal user-related information (e.g. IDs and passwords). Additionally, it can provide insights into malicious behaviors such as cryptocurrency wallet addresses for ransomware payments and C&C server domains utilized by the malware. Even though system artifacts on the disk may become inaccessible due to deletion or tampering, analyzing these artifacts through a memory dump enables the investigation of various behaviors that occurred on the Windows system.

## 3. Methodology and experiments

### 3.1. Methodology overview

Fig. 1 presents an overview of the proposed volGPT. Investigator can use volGPT as a form of the Volatility plugin for suspicious process triaging. Subsequently, following the procedures implemented within the volGPT plugin, process information extracted from *pslist*, *vadinfo*, and *malfind* plugins is sent to the LLM in the form of decision rule-based prompt, incorporating domain knowledge related to triage in memory forensics. The LLM interprets the prompt and provides the user with information on suspected processes in a unified format, explaining what the suspected process is and the reasons for triaging it.

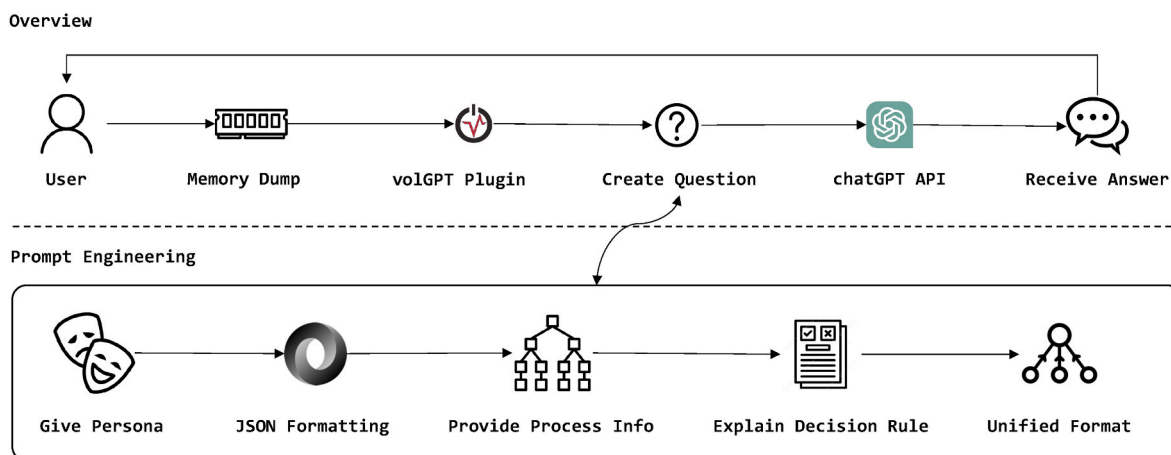


Fig. 1. Overview of volGPT.

### 3.2. Experimental environment

For our experiments, we utilized Python 3.9, Volatility 2.5.2, and the OpenAI API for *gpt-3.5-turbo*. In Volatility 2.5.2, we employed the *pslist*, *vadinfo* and *malfind* plugins. The *gpt-3.5-turbo* model is known to have been trained on natural language corpus up until September 2021. To ensure that *gpt-3.5-turbo* generates consistent triage explanations and results, we set the ‘temperature’ hyper-parameter to 0, while leaving other parameters at their default values. Temperature is a parameter that controls the degree of diversity, allowing us to reduce the randomness in the output of the GPT. Despite the specified configuration, the stochastic nature of GPT model still have randomness. To address this issue, we execute volGPT three times under the same conditions and used the result with the highest frequency for evaluation.

### 3.3. Prompt engineering

Prompt engineering is a fine-tuning method for LLM models, instructing the model with appropriate commands to generate the desired results. While LLM models strive to produce optimal results based on the input prompt, achieving higher-quality and more relevant outcomes as per the user’s desires requires a more extensive dataset and detailed instructions within the prompt. In essence, prompt engineering involves selecting the most suitable format, syntax, words, and symbols to guide the LLM model to interact meaningfully with the user. We have experimented with various prompt engineering techniques to obtain the desired results from the GPT model.

#### 3.3.1. Persona pattern

By employing the persona pattern in a prompt, the output of LLM takes on a specific role’s perspective. This results in the LLM interpreting the user’s language and producing results that are closer to the intended meaning, thereby increasing the probability of generating outcomes that are suitable for the user (White et al., 2023). The prompt engineering used in this study is tailored to the domain of digital forensics. As such, a persona was assigned to represent a cyber investigator, focusing on the task of process triaging.

#### 3.3.2. JSON-based data structure

Currently, GPT-3.5 model supports real-time interactions where it takes text as input and generates text as output. However, artifacts extracted from memory dumps often involves relationships with two or more dimensions and encompasses various data structures. When dealing with hierarchical parent-child relationships, multiple columns, or extensive binaries, simply passing this input as single-line text poses significant challenges for GPT in accurately extracting and interpreting

the information. To address this issue, the input is structured in JSON format, which provides a syntax that computers can easily interpret, accommodating diverse data structures through key-value pairs. In cases where extensive binary representation is necessary, as seen in the results of the *malfind* plugin, GPT-3.5 model has a limit of 4097 tokens per session. Therefore, for exceeding this token limit, we truncated and used it as an input for GPT model.

#### 3.3.3. Setup for JSON interpretation

Even with the input is presented in an JSON format, additional explanations are crucial to ensure that GPT accurately understands the user’s intentions. For instance, if there is examples similar to Fig. 2, a comprehensive explanation of the entire dataset and individual explanations for each key should be provided. Explaining that the “pid” value represents the process ID, “ppid” is the parent process ID, and “name” indicates the process name, helps GPT model interpret and remember the given JSON accurately. We used a Python script to automatically parse these data.

#### 3.3.4. Provide decision rule

After making an input of the JSON mentioned above, we can provide explanations for the rules we intend to triage (see Table 1). Utilizing the same key names in JSON as the column names used in the preceding triage explanation, we propose a decision ruleset that leverages fundamental domain knowledge employed in conducting triage in memory forensics.

#### 3.3.5. Using the template for standardized output

LLM can produce different results for the same question depending on the given context and parameters. To evaluate volGPT, prompt engineering must be performed to ensure consistent answers in the same format. We utilized a template to output results in a consistent format, focusing on identifying suspicious malicious processes in memory

```
[
  {'pid': 4, 'ppid': 0, 'pname': 'System'},
  {'pid': 92, 'ppid': 4, 'pname': 'Registry'},
  {'pid': 304, 'ppid': 4, 'pname': 'smss.exe'},
  {'pid': 500, 'ppid': 404, 'pname': 'wininit.exe'},
  {'pid': 508, 'ppid': 488, 'pname': 'csrss.exe'},
  {'pid': 588, 'ppid': 488, 'pname': 'winlogon.exe'},
  {'pid': 644, 'ppid': 500, 'pname': 'services.exe'},
  {'pid': 660, 'ppid': 500, 'pname': 'lsass.exe'},
  ...
]
```

Fig. 2. JSON based data interpretation.

**Table 1**  
Process triage judgement ruleset.

Plugin	Data	Rule
Process List (pslist)	PID, PPID, Process Name	When the name of the process has a unique name, unlike the system process. It's similar process name contains the description of malware evasion techniques or actual malwares. It's not svchost.exe or explorer.exe, which is used by Windows operating systems, but rather if the process name is malicious.
VAD Area Information (vadinfo)	PID, PPID, VAD Permission, VAD Area	VAD Page permission is "PAGE_EXECUTE_READWRITE"
Malware Find (malfind)	PID, PPID	"MZ", "shellcode" string certainly exists in the vad area

dumps. The template includes PID, process name, and the reason for the judgment in triaging (White et al., 2023).

### 3.3.6. Generated prompt

Therefore, the final configured prompt is as presented in Fig. 3. By sending this prompt through GPT's API request, we can receive responses and obtain results for Triage.

## 3.4. Dataset

The experiment dataset was the Virtual Machine (VM) Memory Dumps for Ransomware Forensics (Arfeen et al., 2020). This dataset consists of 100 memory dumps each for WannaCry, TeslaCrypt, Vipasana, Cerber, and HiddenTear ransomware. The generation process involved running four legitimate processes on Windows 8 to create a baseline. Although we used Windows 8, there is no significant difference in terms of process relationship compared to Windows 10 or 11. Ransomware was then executed, and memory dumps were created every 5 min by creating virtual machine snapshots. Once the specified number of memory dumps had been created, the memory state was restored using the snapshot, and the additional dataset was generated by infecting the virtual machine with new ransomware.

## 3.5. Data preprocessing

The dataset used in the experiment created using snapshots of the virtual machine from the time the ransomware was executed every 5 min. Some ransomware terminates and self-deletes after encrypting files on the victim's computer. In such cases, the process may not be present after the process terminates through memory forensics. Therefore, we excluded these memory dumps from the experiment by the following data preprocessing procedure:

- Step 1: Execute ransomware within a VM and monitor the processes generated by the ransomware.

```

● :
You are a cyber investigator and need to analyze the data given.

When we look for malicious processes, we can judge through various methods.
The first is when the name of the process has a unique name, unlike the system process.
It's like process name contains the description of malware evasion techniques or malicious name.
It's not like svchost.exe or explorer.exe, which is used by Windows operating systems, but rather
if the process name is malicious.

Now This is a JSON list of the PIDs, PPIDs, and process names of the processes.
In the JSON, the value "pid" is the ID of the process,
"ppid" is the ID of the parent process of the process, and "pname" is the name of the process.
... (JSON Data Section)

Based on the criteria you told us, please determine which processes are malicious.
Just print out the exact ones that meet the conditions
Please let me know the malicious process name, PID, and the reason for the judgment.
    
```

● Persona ● Explain Decision Rule ● Provide Process Info ● Unified Format

**Fig. 3.** Prompt example of volGPT.

- Step 2: Identify processes in the memory dump using the pslist plugin of Volatility 3.
- Step 3: Generate the label for the suspicious process list by taking the intersection of the process lists extracted in Steps 1 and 2.
- Step 4: The intersection serves as the label for the list of suspicious processes to be identified in the dataset.

The information about the dataset and label used to evaluate volGPT is explained in Table 2.

## 3.6. Evaluation metric

For evaluation, we considered accuracy and triage efficiency as performance metrics. Triage efficiency is an indicator that can be used to judge how much work efficiency it gives to the investigator. This metric represents the number of identified suspicious processes relative to the overall processes, calculating the number of how many suspicious processes can be identified in the total processes outputted as results by Volatility 3 pslist plugin.

- Accuracy =  $(TP+TN)/(TP+TN+FP+FN)$
- Triage efficiency =  $(TP+FP)/(TP+TN+FP+FN)$

## 4. Triage on ransomware infected memory dump

### 4.1. Experimental performance

We evaluated the triage performance of volGPT framework on the dataset shown in Table 2. Based on the suspicious process labels, the proposed framework achieved a detection accuracy of up to 99 % and a minimum of 87 % in five ransomware families. In experiments involving all ransomware types, there seems to be a tendency for volGPT to avoid misidentifying benign processes as suspicious, resulting in higher accuracy. The result from volGPT is meaningful because it is important to swiftly identify legitimate processes to obtain a list of suspicious processes. Furthermore, in terms of triage efficiency, we confirmed the ability to reduce the number of suspicious processes to be investigated per memory dump from a maximum of 21 times to a minimum of 6 times as Table 3. By examining the responses of volGPT to the five types of ransomware, we analyzed the criteria of triage, the reason for false positives and false negatives, and why the suspected process was identified in the VAD area.

### 4.2. Detailed analysis on WannaCry ransomware

#### 4.2.1. Reason of judgement: true positive

4.2.1.1. *ed01ebfbc9eb5b*. According to our analysis, the process name of WannaCry ransomware is composed of random alphanumeric characters, which is different from the naming convention of benign processes. Therefore, volGPT identified those processes as suspicious.

4.2.1.2. *Wanadecryptor*. The process named *wanadecryptor* was identified as a suspicious target for triage because it is associated with the *wanadecryptor* malware, which has been used in WannaCry ransomware attacks.

4.2.1.3. *Taskhsvc*. Our framework identified the *taskhsvc* process as a malicious masquerading of the legitimate *taskhost.exe*. Also, this was spawned from the *wanadecryptor*, which can be determined as suspicious activity.

4.2.1.4. *Conhost*. The *conhost.exe* manages and displays the Windows console for command-line applications. is expected to be spawned from a Windows shell process. However, our framework identified the *conhost*

**Table 2**

Dataset description.

Ransomware	Number of Preprocessed dump	Number of Total Process	MD5 Hash	Label
WannaCry	100	5826	84c82835a5d21bbcf75a61706d8ab549	ed01ebfbc9eb5b (filename of ransomware), wanadecryptor, taskhsvc, conhost, vssvc
TeslaCrypt	13	767	209a288c68207d57e0ce6e60ebf607296e080aa085293bb9fbdcc9015337d3096d3d62a4cff19b4f2cc7ce9027c33be8	fjsauxu, prvudcr (filename of ransomware), vssvc, vssadmin
Cerber	100	4685	8b6bc16fd137c09a08b02bbe1bb7d670	cerber (filename of ransomware), mshta, notepad, cmd
Vipasana	2	97	2aea3b217e6a3d08ef684594192cafc8a890e2f924dea3cb3e46a95431ffae39adb5c262ca4f95fee36ae4b9b5d41d4565a91bdb3b625460859180e9492fe40e	0442cfabb32126, c0cf40b8830d66,e49778d20a2f9b (filename of ransomware), cmd
HiddenTear	203	5061	65a91bdb3b625460859180e9492fe40e	hidden-tear (filename of ransomware)

**Table 3**

Overall Performance of volGPT.

Ransomware	All	TP	FN	FP	TN	Accuracy	Triage efficiency
WannaCry	5826	282	3	245	5296	95.74322	11.0550285
TeslaCrypt	767	18	14	77	658	88.1355932	8.07368421
Cerber	4685	129	0	594	3962	87.321238	6.47994467
Vipasana	97	4	2	2	89	95.8762887	16.1666667
HiddenTear	5061	203	0	28	4830	99.4467497	21.9090909
Total	16436	636	19	946	14835	94.1287418	10.3893805

process as suspicious because it was spawned from the suspicious *taskhsvc* process, which is an unusual parent-child process relationship.

#### 4.2.2. Reason of judgement: false negative

4.2.2.1. *vssvc*. WannaCry can delete Volume Shadow Copy using *vssvc.exe* to prevent the recovery of encrypted files. While volGPT considered *vssvc* as a legitimate process, the fact that it is spawned from processes associated with ransomware should designate it as a suspicious process. However, despite this, due to a lack of clear prompt instructions or sufficient background knowledge provided in the prompt, volGPT may not have identified this as a suspicious process.

#### 4.2.3. Reason of judgement: VAD area

WannaCry does not perform process injection or DLL injection. For this reason, our attempt to triage volGPT using the VAD area did not reveal any distinctive findings.

### 4.3. Detailed analysis on cerber ransomware

#### 4.3.1. Reason of judgement: true positive

4.3.1.1. *Cerber*. Our framework explained the process name *cerber* does not correspond to a known system process or benign process, and this name may be associated with the Cerber ransomware.

4.3.1.2. *Mshta*. The *mshta.exe* can interpret hta files in Windows and execute scripts related to HTML applications. however it can be abused to execute malicious code contained in these scripts. Considering this, our framework identified the *mshta* process as suspicious by confirming that the parent process of this process is *cerber*, which is an abnormal relationship, as shown in the example in Appendix A.

4.3.1.3. *Notepad and cmd*. In the *cmd.exe*, it is possible that malicious code exists in the executed commands, but the *notepad.exe* is a text editor that is little relevant to malware. Our framework identifies the both *notepad* and *cmd* processes as suspicious because their parent process has already been flagged as suspicious.

#### 4.3.2. Reason of judgement: false negative

Using our proposed framework, we found that there were none of the false negatives during the experiment of the Cerber ransomware.

#### 4.3.3. Reason of judgement: VAD area

As a result of the triage, the *cerber* process was detected. Our framework determined that the process was suspicious because it had a PAGE\_EXECUTE\_READWRITE region in some of its VAD and the difference between the start of VPN and the end of VPN values was small, therefore, the specific memory range was given both write and execute permissions for miscellaneous purpose. In addition, the process contained the "MZ" signature, which represents the structure of a PE file, and strings for APIs that are often used by malware, for example, WriteFile, GetModuleHandleA, and CloseHandle, in the VAD region. Based on these findings, our framework concluded that the process may be malicious and suggested further analysis by the investigator, as illustrated in Fig. 4.

### 4.4. Detailed analysis on TeslaCrypt ransomware

#### 4.4.1. Reason of judgement: true positive

4.4.1.1. *Fjsauxu, prvudcr*. According to our analysis, the process name of TeslaCrypt ransomware is composed of random alphanumeric characters, which is different from the naming convention of benign processes. Therefore, volGPT identified the process as suspicious.

#### 4.4.2. Reason of judgement: false negative

4.4.2.1. *vssvc, vssadmin*. TeslaCrypt is similar to WannaCry in that it can delete Volume Shadow Copies using the *vssvc.exe* and *vssadmin.exe*, as depicted in Fig. 5. Both processes are legitimate Windows processes, but despite being identified as suspicious processes by volGPT due to being spawned from ransomware processes, the model may not have recognized them as such due to a lack of specific prompt instructions or background knowledge about ransomware provided in the prompt.

#### 4.4.3. Reason of judgement: VAD area

Based on the results of Volatility's malfind plugin, our framework determined that the *prvudcr* process contains a PE file that starts with the

Volatility3 windows.malfind Result

```

3724 cerber.exe 0x2d0000 0x2fffff VadS
4d 5a 90 00 03 00 00 00 MZ.....
04 00 00 00 ff ff 00 00 .....
b8 00 00 00 00 00 00 00 .....
40 00 00 00 00 00 00 00 @.....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 d8 00 00 00 .....
0x2d0000: dec ebp
0x2d0001: pop edx
0x2d0002: nop
0x2d0003: add byte ptr [ebx], al
0x2d0005: add byte ptr [eax], al
0x2d0007: add byte ptr [eax + eax], al
0x2d000a: add byte ptr [eax], al

3724 cerber.exe 0x310000 0x340fff VadS
33 33 33 00 00 00 00 00 333.....
00 00 00 00 00 00 00 00 .....
00 47 65 74 4d 6f 64 75 .....
6c 65 48 61 6e 64 6c 65 .....
41 00 57 72 69 74 65 46 .....
69 6c 65 00 00 00 00 00 .....
00 00 00 43 6c 6f 73 65 .....
48 61 6e 64 6c 65 00 00 .....
0x310000: xor esi, dword ptr [ebx]
0x310002: xor eax, dword ptr [eax]
    
```

volGPT Result

```

Based on the given criteria, the following processes can be determined

1. Process Name: cerber.exe
   PID: 3724
   Reason for Judgement: The vad_data of this process contains the "MZ" string,
   which indicates the presence of a malicious executable file.

2. Process Name: cerber.exe
   PID: 3724
   Reason for Judgement: The vad_data of this process contains the string "GetModuleHandleA"
   and "WriteFile", which are Windows API functions commonly used by malware.

3. Process Name: cerber.exe
   PID: 3724
   Reason for Judgement: The vad_data of this process contains the string "CloseHandle",
   which is Windows API function commonly used by malware.
    
```

Fig. 4. VAD area explanation of cerber ransomware by volGPT.

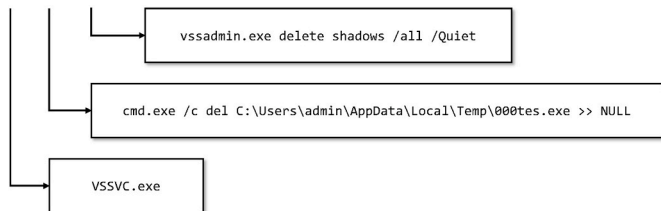


Fig. 5. Volume shadow copy related process.

“MZ” signature and the process may be involved in malicious activity.

4.5. Detailed analysis on vipasana ransomware

4.5.1. Reason of judgement: true positive

4.5.1.1. 0442cfabb32126, c0cf40b8830d66, e49778d20a2f9b. According to our analysis, the process name of Vipasana ransomware is composed of random alphanumeric characters, which is different from the naming convention of benign processes. Therefore, volGPT identified those processes as suspicious.

4.5.2. Reason of judgement: false negative

4.5.2.1. cmd. Vipasana ransomware uses batch scripts to perform file encryption and other malicious activities. cmd.exe is originally a legitimate Windows process, but it can also be used to execute malicious scripts, as shown in Fig. 6. However, it seems that volGPT was unable to detect these activities due to a lack of specific decision rules.

4.5.3. Reason of judgement: VAD area

Similar to the case of Cerber Ransomware analysis, our framework



Fig. 6. cmd.exe spawn case.

determined that the process was suspicious because it had a PAGE\_EXECUTE\_READWRITE region in some of its VAD and the difference between the start of VPN and the end of VPN values was small, therefore, the specific memory range was given both write and execute permissions for a specific purpose. Furthermore, according to our framework’s interpretation of the Volatility malfind plugin results, this suggests that further analysis is required to determine whether the process is malicious.

4.6. Detailed analysis on HiddenTear ransomware

4.6.1. Reason of judgement: true positive

4.6.1.1. Hidden-tear. According to our analysis, our framework identified the process as suspicious based on the process name, hidden-tear, which suggests that it is associated with the HiddenTear ransomware.

4.6.2. Reason of judgement: false negative

In the HiddenTear ransomware, no false negatives were found when it was evaluated using our proposed framework.

4.6.3. Reason of judgement: VAD area

As shown in Fig. 7, a string that could be used as a shellcode was identified. It is necessary to further analyze whether the string actually performs the role of shellcode, but since the PAGE\_EXECUTE\_READWRITE permission is set for the region, it cannot be ruled out that the region could be exploited for shellcode. As a result, it is possible to judge that the process is suspicious.

5. Discussion

5.1. Detailed analysis on false positive

5.1.1. Hallucination

Generally, hallucination refers to the generation of inaccurate or false information by generative models. In the context of volGPT, hallucination occurs when it erroneously judges a process as suspicious based on false reasoning, contrary to the provided decision rule. Despite instructions in the prompt to consider Windows system processes (e.g. svchost.exe or explorer.exe) as normal, there are instances where volGPT erroneously determines them as suspicious. Specifically, there are cases where svchost.exe is hallucinated as suspicious, leading to the generation



**Table 4**  
Comparison on other related works.

Researches	Techniques	Dataset	Decision Unit	Overall performance	Explanations
Lashkari et al. (2021)	K-NN	VolMemLyzer-V0 (1900 samples)	Memory dump	93 %	X
Al-Qudah et al. (2023)	One-Class Support Vector Machine	CIC-MalMem-2022	Memory dump	99.4 %	X
Dener et al. (2022)	Logistic Regression	CIC-MalMem-2022	Memory dump	99.9 %	X
Smith et al. (2023)	PCA + SVM	CIC-MalMem-2022	Memory dump	99 %	X
Ali-Gombe et al. (2023)	RGB image, CNN	850 Android malware, 600 benign apps	Process	95.23 %	X
Dai et al. (2018)	Grayscale, HOG, MLP	1468 samples from 8 family Windows malware	Process	94.1 %	X
Arfeen et al. (2022)	XGBoost	Virtual Machine Memory Dumps for Ransomware Forensics	Memory dump	80.5 %	O (Feature Selection)
volGPT (Ours)	Large Language Model Prompt Engineering	Virtual Machine Memory Dumps for Ransomware Forensics	Process	94.1 %	O (Natural language)

Gradient) was extracted for classification using multi-layered perceptron. A total of 1468 samples were collected from 8 different malware families, and an F1-score of 94.1 % was achieved.

In Ali-Gombe et al. (2023), 850 Android malware samples and 600 benign apps were used to collect process dumps through CuckooDroid. Subsequently, these process dumps were converted into images representing RGB channels and used as input values for a CNN model. The trained CNN model was then used for the classification of malware and benign samples, achieving an F1-score of 95.23 %.

Landman and Nissim (2021) utilized 56 malware samples and 56 benign samples operable in a Linux environment. The study proposed a malware detection method based on a CNN model, which involves extracting ELF headers from process dumps and converting them into RGB arrays through data processing. This proposed method demonstrated various performance metrics close to 99 %.

In Liu et al. (2023), researchers conducted experiment on 1050 samples from 80 malware families and 1010 benign samples, focusing on memory-resident malware. The study proposed the MRm-DLDET model, combining ResNet-18, GRU, and attention, and used RGB images of process dumps created by executing malware as inputs. The experiment was conducted as binary classification and achieved an F1-score of 0.9836.

Song et al. (2018) proposed method that represents the relationships between kernel objects in memory using a graph neural network and automatically recognizes the characteristics of kernel objects using object detection. This model showed robustness against on pool tag manipulation and Direct Kernel Object Manipulation (DKOM), known limitations of the traditional list-traversal and signature-based approaches. It also demonstrated the ability to classify seven different memory kernel objects in research tasks, achieving performance close to an F1-score of 0.99.

Existing studies have focused on identifying memory dumps as malicious and have demonstrated excellent detection performance. However, as shown in Table 4, they have limitations in that they cannot identify which processes in the memory dump are malicious, and they do not provide an explanation for the basis of deeming something as malicious. Our research differs in that it configures volGPT to determine the maliciousness based on processes rather than memory dump criteria. It provides explanations for triage along with high detection performance and can be easily applied to Volatility.

### 6.3. Digital forensics with LLM

Large language models built upon extensive text corpus have the capability to understand and process natural language, making them increasingly applied in various fields as systems that aid in decision-making. There are studies proposing the use of LLMs as decision

support systems in the field of digital forensics. Research is ongoing in digital forensics as well, applying LLMs to assist digital forensic analysts by leveraging their decision-making capabilities.

Scanlon et al. (2023) evaluated the digital forensic investigation capabilities of ChatGPT based on GPT-4 model. The study examined its performance in situations about identifying artifacts based on user actions, code generation for carving and password recovery, incident response. It was found to be somewhat useful, but it was argued that when automating analysis with ChatGPT, supervision by a knowledgeable user is necessary due to potential issues because of incorrect assumptions and inaccuracies in analysis results.

Michelet and Breiteringer (2023) investigated whether ChatGPT based on GPT-3.5 and LLaMA can be used for writing digital forensic reports. The study founded that LLMs could assist practitioners in the report-writing process, especially in drafting and generating detailed diagrams for artifacts. However, due to the hallucination issue, it needed that verification of sources in the generated reports is necessary.

Henseler and van Beek (2023) demonstrated that ChatGPT can be used for investigating digital evidence for legal professionals. They showed that user input natural language can generate Hansken Query Language (HQL) to filter desired information from vast evidence, and it's also capable of summarizing and visualizing chat messages. The study proposed a 'detective' persona, suggesting that ChatGPT can reference various evidence and infer situations. The authors argued that ChatGPT has the potential to improve the efficiency of digital forensic investigations but needs to be objectively proven useful with more difficult situations and more data, and skilled digital forensic experts are needed to correct ChatGPT's mistakes.

Nguyen et al. (2023) proposed an approach using the LLaMA 2 7B-parameter model to detect online sexual predatory and abusive expressions. Unlike previous studies that selected features and designed

```
(Prompt) >>> sp{
You are a cyber investigator and need to analyze the data given.

When we look for malicious processes, we can judge through various methods.
The first is when the name of the process has a unique name, unlike the system process.
It's like process name contains the description of malware evasion techniques or malicious name.
It's not like svchost.exe or explorer.exe, which is used by Windows operating systems, but rather if the process name
is malicious.

Now this is a JSON list of the PIDs, PPIDs, and process names of the processes.
In the JSON, the value "pid" is the ID of the process,
"ppid" is the ID of the parent process of the process, and "pname" is the name of the process.
...

Based on the criteria you told us, please determine which processes are malicious.
Just print out the exact ones that meet the conditions.
Please let me know the malicious process name, PID, and the reason for the judgment.

: Based on the given criteria, the following processes are considered malicious:
1. Process Name: 1e0f1b9b697ab49
   PID: 476
   Reason for Judgment: The process name is a random string of characters, which does not follow the naming convention of
   legitimate processes. This indicates that it may be a malicious process.
2. Process Name: cmd.exe
   PID: 1872
   Reason for Judgment: Although cmd.exe is a legitimate process, in this case, it is considered malicious because it is a
   child process of the previously identified suspicious process (1e0f1b9b697ab49). The presence of cmd.exe as a child process of
   a suspicious process suggests malicious activity.

Please note that this analysis is based on the information provided and additional investigation may be required for a
comprehensive determination.
```

**Fig. 8.** Example of tool usage.

classifiers, this approach showed strong performance through an LLM utilizing the Low Rank Adaptation (LoRA) method. Furthermore, its usefulness was demonstrated in Roman Urdu and Urdu. The results of the proposed approach suggested the potential applicability in various professional fields, including cybersecurity, and the need for experiments using data from those fields.

These studies are in the initial stages of applying LLMs to digital forensics, lacking in detailed utilization methods and performance evaluation, making it difficult to prove their practicality in digital forensic procedures. Indeed, further research is needed to evaluate whether LLMs can provide practical assistance in digital forensics. There is a necessity to address numerous use cases, assess performance, and identify areas for improvement.

### 7. Conclusions

We aimed to automate the process triage for incident response by utilizing a GPT model to extract process information from memory dumps infected with ransomware, using the Volatility framework. To achieve this goal, we developed a tool named volGPT, as illustrated in Fig. 8. This tool utilizes Volatility plugins to extract information about process lists and VAD regions, providing results and explanations for triage based on pre-defined rules and prompts.

In our experiments with five different ransomware samples, volGPT demonstrated an average detection rate of 94.12 % of accuracy. From the perspective of triage efficiency, it compressed the number of suspicious processes to be investigated into 10 % compared to analyzing the entire process list, confirming its capability to reduce the workload for analysts.

Through our research, we believe that we contribute to the automation of digital forensics and incident response using large language models. In future work, we plan to apply this methodology to other types of malware, particularly obtaining samples that involve process injection, to understand whether volGPT can identify and the reasons behind its identification. Additionally, in our study, we solely utilized the inferential capabilities of the GPT model to analyze given prompts and evaluate them. However, we believe that by employing Retrieval Augmented Generation (RAG) or fine-tuning, as seen in recent research on large language models, we can achieve higher performance and generate better explanations than our current study. Furthermore,

providing detailed explanations of ransomware features in prompts to reduce false negatives, and offering information about benign processes, could help in decreasing false positives.

With the advancement of various anti-forensic methods and the growth and diversification of analysis materials in digital forensics, there is an increase in domain-specific tasks, which in turn raises the amount of specialized knowledge investigators must acquire. However, due to constraints in time and resources, it's impractical for many investigators to hold expert knowledge across the entire spectrum of digital forensics and incident response. Following this study, we aspire for the creation of a digital forensic-specific LLM enriched with a broad spectrum of expert knowledge. It aims to act as a navigational aid for investigators, providing direction in their inquiries and fostering more efficient and practical digital investigations.

### CRedit author statement

Dong Bin Oh: Methodology, Conceptualization, Investigation, Writing - original draft, Validation. Donghyun Kim: Methodology, Software. Donghyun Kim: Writing - review & editing, Validation. Huy Kang Kim: Writing - review & editing, Resources, Supervision.

### Data availability

The data and tools that support the findings of this study are available from the corresponding author, upon reasonable request.

### Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Acknowledgement

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00624, Development of Intelligence Cyber Attack and Defense Analysis Framework for Increasing Security Level of C-ITS).

### Appendix A. Major Reason of Judgement from volGPT

Ransomware	Label	Major Reason of Judgement from volGPT
WannaCry	ed01ebfbc9eb5b	the process name consists of a random string of characters, which is a common technique used by malware to obfuscate its identity. this strongly suggests that it is a malicious process.
	wanadecryptor	the process name contains the description of a known malware called wanadecryptor, which is associated with the WannaCry ransomware attack.
	taskhsvc	the process name contains a misspelled word, which is a common technique used by malware to evade detection. additionally, the process is spawned by a suspicious parent process, which raises further suspicion
	conhost	although "conhost.exe" is a legitimate windows process, its parent process is determined to be malicious. this raises suspicion about the legitimacy of the conhost.exe process in this context.
TeslaCrypt	fjsauxu	the process name does not follow the naming convention of a system process, and it contains a random string of characters which is suspicious.
	prvudcr	the process name does not follow the naming convention of a system process, and it contains a random string of characters which is suspicious.
Cerber	cerber	the process name "cerber.exe" does not match any known system process or benign process. it is also associated with malware evasion techniques, as it is named after the cerber ransomware.
	mshta	the process name "mshta.exe" is a known system process, but in this case, it is considered malicious due to its association with the previously identified malicious process "cerber.exe". the spawning of mshta.exe from a suspicious process raises suspicion.
	notepad	the process name is a legitimate windows process, but in this case, it is spawned by a suspicious parent process which is also spawning other malicious processes.
	cmd	the process name is a legitimate windows process, but in this case, it is spawned by a suspicious parent process which is also spawning other malicious processes.

(continued on next page)

(continued)

Ransomware	Label	Major Reason of Judgement from volGPT
Vipasana	0442cfabb32126	the process name consists of random alphanumeric characters, which is not a typical naming convention for legitimate processes. this suggests that it may be a malicious process.
	c0cf40b8830d66	similar to the previous process, this process name also consists of random alphanumeric characters, indicating a potential malicious nature.
	e49778d20a2f9b	The process name follows the same pattern as the previous two processes, consisting of random alphanumeric characters. This raises suspicion and suggests it may be a malicious process.
HiddenTear	hidden-tear	the process name contains the name of a known malware, "hidden-tear".

## References

- Al-Qudah, M., Ashi, Z., Alnabhan, M., Abu Al-Haija, Q., 2023. Effective one-class classifier model for memory dump malware detection. *J. Sens. Actuator Netw.* 12, 5.
- Ali-Gombe, A., Sudhakaran, S., Vijayakanthan, R., Richard III, G.G., 2023. crgb\_mem: at the intersection of memory forensics and machine learning. *Forensic Sci. Int.: Digit. Invest.* 45, 301564.
- Arfeen, A., Khan, M.A., Zafar, O., Ahsan, U., 2020. Virtual Machine (VM) Memory Dumps for Ransomware Forensics. <https://doi.org/10.7910/DVN/YVL3CW>.
- Arfeen, A., Asim Khan, M., Zafar, O., Ahsan, U., 2022. Process based volatile memory forensics for ransomware detection. *Concurrency Comput. Pract. Ex.* 34, e6672.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* 33, 1877–1901.
- Carrier, T., Victor, P., Tekeoglu, A., Lashkari, A.H., 2022. Detecting obfuscated malware using memory feature engineering. In: *Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP, INSTICC. SciTePress*, pp. 177–188. <https://doi.org/10.5220/0010908200003120>.
- Case, A., Richard III, G.G., 2017. Memory forensics: the path forward. *Digit. Invest.* 20, 23–33.
- Dai, Y., Li, H., Qian, Y., Lu, X., 2018. A malware classification method based on memory dump grayscale image. *Digit. Invest.* 27, 30–37.
- Dener, M., Ok, G., Orman, A., 2022. Malware detection using memory analysis data in big data environment. *Appl. Sci.* 12, 8604.
- Dolan-Gavitt, B., 2007. The vad tree: a process-eye view of physical memory. *Digit. Invest.* 4, 62–64.
- Henseler, H., van Beek, H., 2023. Chatgpt as a Copilot for Investigating Digital Evidence.
- Kang, S., Kim, S., Park, M., Kim, J., 2018. Study on windows event log-based corporate security audit and malware detection. *Journal of the Korea Institute of Information Security & Cryptology* 28, 591–603.
- Landman, T., Nissim, N., 2021. Deep-hook: a trusted deep learning-based framework for unknown malware detection and classification in linux cloud environments. *Neural Network.* 144, 648–685.
- Lashkari, A.H., Li, B., Carrier, T.L., Kaur, G., 2021. Volmemlyzer: volatile memory analyzer for malware classification using feature engineering. In: *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*. IEEE, pp. 1–8.
- Ligh, M.H., Case, A., Levy, J., Walters, A., 2014a. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. John Wiley & Sons.
- Ligh, M.H., Case, A., Levy, J., Walters, A., 2014b. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. John Wiley & Sons, pp. 149–187.
- Liu, J., Feng, Y., Liu, X., Zhao, J., Liu, Q., 2023. Mrm-ddet: a memory-resident malware detection framework based on memory forensics and deep neural network. *Cybersecurity* 6, 21.
- Michelet, G., Breiting, F., 2023. Chatgpt, Llama, Can You Write My Report? an Experiment on Assisted Digital Forensics Reports Written Using (Local) Large Language Models, arXiv Preprint arXiv:2312.14607.
- Nguyen, T.T., Wilson, C., Dalins, J., 2023. Fine-Tuning Llama 2 Large Language Models for Detecting Online Sexual Predatory Chats and Abusive Texts, arXiv Preprint arXiv: 2308.14683.
- Saidi, R.M., Ahmad, S.A., Noor, N.M., Yunus, R., 2013. Windows registry analysis for forensic investigation. In: *2013 the International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAECE)*. IEEE, pp. 132–136.
- Scanlon, M., Breiting, F., Hargreaves, C., Hilgert, J.-N., Sheppard, J., 2023. ChatGPT for digital forensic investigation: the Good, the bad, and the unknown. *Forensic Sci. Int.: Digit. Invest.*, 301609 <https://doi.org/10.1016/j.fsidi.2023.301609>.
- Scarlato, A., Lan, A., 2023. Tree-based Representation and Generation of Natural and Mathematical Language arXiv preprint arXiv:2302.07974.
- Schuster, A., 2006. Searching for processes and threads in microsoft windows memory dumps. *Digit. Invest.* 3, 10–16.
- Smith, D., Khorsandroo, S., Roy, K., 2023. Supervised and unsupervised learning techniques utilizing malware datasets. In: *2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC)*. IEEE, pp. 1–7.
- Song, W., Yin, H., Liu, C., Song, D., 2018. Deepmem: learning graph neural network models for fast and robust memory forensic analysis. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 606–618.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al., 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models arXiv preprint arXiv:2307.09288.
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C., 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with Chatgpt arXiv preprint arXiv:2302.11382.
- Wu, S., Irsoy, O., Lu, S., Dabrovolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., Mann, G., 2023. Bloomberggpt: A Large Language Model for Finance arXiv preprint arXiv:2303.17564.
- Yosifovich, P., Solomon, D.A., Ionescu, A., 2017. *Windows Internals, Part 1: System Architecture, Processes, Threads, Memory Management, and More*. Microsoft Press, pp. 401–403.