Volatile Memory Forensics for Ransomware Detection and Encryption Artifact Identification

ASHLY M DEVASIA
Department of Computer Science and Engineering
Government Engineering College, Wayanad
Kerala, India
ashlymdevasia@gmail.com

SHIBILY JOSEPH
Department of Computer Science and Engineering
Government Engineering College, Wayanad
Kerala, India
shibily.j@gecwyd.ac.in

***Abstract*—Volatile memory forensics is a crucial area of digital forensics that focuses on the acquisition, preservation, and analysis of data stored in Random Access Memory (RAM). Unlike traditional disk forensics, it provides a real-time view of system activity by capturing transient artifacts such as running processes, network connections, loaded modules, encryption keys, clipboard data, and in-memory malware. With the rise of fileless malware, process injection, and memory-resident ransomware, RAM analysis has become essential for detecting threats that leave minimal or no traces on disk.**

This work investigates volatile memory forensics in ransomware analysis through experiments conducted in a controlled virtual environment. A ransomware simulator using AES encryption was developed to encrypt files and generate forensic artifacts. Memory dumps collected during execution were analyzed using the Volatility Framework along with YARA-based pattern matching, entropy analysis, and string extraction techniques. The investigation successfully identified malicious processes and recovered AES encryption keys from process memory. High-entropy regions were used to locate cryptographic material, while YARA rules helped detect ransomware-related patterns.

The results demonstrate that critical forensic evidence, including encryption keys and runtime traces, can be recovered from RAM, highlighting the importance of volatile memory analysis in ransomware investigations and incident response.

***Index Terms*—Volatile memory forensics, RAM analysis, ransomware detection, encryption key recovery, AES, entropy analysis, YARA, Volatility Framework, digital forensics.**

I. Introduction

Digital forensics is an important field in cybersecurity that focuses on the identification, acquisition, preservation, and analysis of digital evidence from computer systems and networks [1]. With the rapid increase in cybercrime and sophisticated malware attacks, forensic investigations have become essential for detecting malicious activities, tracing attackers, and supporting legal investigations. Traditional forensic techniques mainly focus on persistent storage devices such as hard disks and SSDs; however, modern cyber threats increasingly operate within volatile memory, leaving minimal traces on permanent storage [2].

Volatile memory, commonly known as Random Access Memory (RAM), stores valuable runtime information such as active processes, loaded DLLs, network connections, clipboard data, encryption keys, and injected malicious code [3]. Unlike persistent storage, RAM contains temporary information that is lost once the system is powered off, making timely memory acquisition extremely important during forensic investigations. Memory forensics provides investigators with a real-time view of system activity and helps uncover hidden malicious behavior that may not be visible through disk analysis alone [4].

Modern ransomware attacks have evolved significantly and now employ advanced evasion techniques such as process injection, reflective DLL loading, and memory-resident execution. Fileless malware executes directly in memory without creating executable files on disk, making traditional antivirus and signature-based detection methods less effective [5]. Ransomware also uses strong cryptographic algorithms such as AES to encrypt victim files, making encryption key recovery an important aspect of forensic analysis [6].

The primary problem addressed in this study is the difficulty in detecting ransomware-related activities and recovering encryption artifacts when limited evidence exists on permanent storage devices. Attackers increasingly rely on memory-based execution and encryption techniques to evade traditional forensic analysis. Therefore, volatile memory analysis becomes necessary for identifying suspicious processes, runtime artifacts, and cryptographic materials associated with ransomware attacks.

The objective of this work is to investigate the role of volatile memory forensics in ransomware detection and encryption artifact identification. This study focuses on analyzing memory dumps using the Volatility Framework together with YARA-based pattern matching, entropy analysis, and string extraction techniques to identify malicious processes and recover AES encryption keys from memory [7].

The major contribution of this paper includes the development of a controlled ransomware simulation environment, acquisition of RAM images during ransomware execution, extraction of encryption-related artifacts from memory, and successful recovery of AES encryption keys using entropy-based analysis techniques. The study also demonstrates the importance of volatile memory forensics in modern incident response and ransomware investigations.

II. BACKGROUND

Volatile memory forensics has become an important area of digital forensic research due to its capability to recover runtime artifacts that are unavailable in traditional disk-based investigations. Researchers have explored several approaches for memory acquisition, malware detection, ransomware analysis, and cryptographic key recovery using volatile memory analysis techniques.

Hamid and Rahman presented a comprehensive review of volatile memory forensics and discussed the importance of RAM analysis in modern cybersecurity investigations [8]. Their work highlighted various memory acquisition tools, forensic frameworks, and challenges associated with volatile memory analysis. Similarly, Case et al. introduced the Volatility Framework, which became one of the most widely used open-source memory forensic tools for extracting memory artifacts from RAM images [9].

Several studies have focused on ransomware and fileless malware detection through memory analysis. Afreen et al. analyzed fileless malware behavior and explained how memory-resident attacks evade traditional disk-based forensic techniques [10]. Khaliq et al. discussed modern ransomware detection tools and techniques, emphasizing the growing need for advanced forensic methods against ransomware attacks [11]. Urooj et al. proposed an adaptive pre-encryption ransomware detection model that attempts to identify ransomware activity before file encryption begins [12].

Memory-based malware classification has also gained significant research attention. Lashkari et al. developed VolMemLyzer, a volatile memory analysis framework that uses feature engineering for malware classification [13]. Maniriho et al. proposed MemAlDet, which combines deep autoencoders and stacked ensemble techniques to improve malware detection accuracy using memory analysis [14]. More recently, Fellicious et al. introduced MemBERT, a foundation model for memory forensics that applies transformer-based learning techniques to volatile memory analysis [15].

Researchers have also explored encryption key extraction from memory. Balogh and Pondelik demonstrated the feasibility of recovering encryption keys from volatile memory during runtime [16]. Kazim et al. recovered chat messages and encryption master keys directly from RAM using memory forensic techniques .Entropy-based analysis methods are commonly used to identify cryptographic materials because encrypted data and cryptographic keys usually exhibit high randomness in memory regions.

Signature-based memory scanning is another widely used approach in volatile memory forensics. Mistry and Dahiya proposed a signature-based memory forensic technique for detecting sophisticated cyberattacks using memory signatures [17]. YARA-based scanning techniques have become highly effective for identifying ransomware-related artifacts and suspicious memory patterns during malware investigations.

Although previous studies have made significant contributions to volatile memory forensics and malware analysis, several research gaps still exist. Many existing approaches mainly focus on malware detection and do not specifically address encryption artifact recovery from ransomware memory dumps. Some machine learning-based techniques require large datasets and high computational resources, while traditional signature-based methods struggle against obfuscated or unknown malware. Additionally, limited work has been carried out on combining entropy analysis, YARA scanning, and volatile memory analysis for AES encryption key identification during ransomware execution.

III. METHODOLOGY

The methodology adopted in this work focuses on the acquisition and analysis of volatile memory for ransomware detection and encryption artifact identification. The experimental setup was designed in a controlled virtual environment to safely execute a ransomware simulator and analyze its runtime behavior using memory forensic techniques. The methodology consists of environment setup, memory acquisition, memory analysis, entropy-based scanning, and AES key identification.

A. Experimental Environment Setup

A virtualized forensic environment was created using Oracle VirtualBox to safely execute and analyze ransomware-related activities without affecting the host operating system [18]. Virtualization provides isolation, repeatability, and secure experimentation for malware analysis. A Windows-based virtual machine was configured with sufficient RAM and storage resources to simulate realistic ransomware execution scenarios.

The experimental environment consisted of:

- Oracle VirtualBox as the virtualization platform
- Windows operating system as the guest environment
- Python-based ransomware simulator
- Memory acquisition and forensic analysis tools

The Overall Workflow of the Proposed Method is shown below:

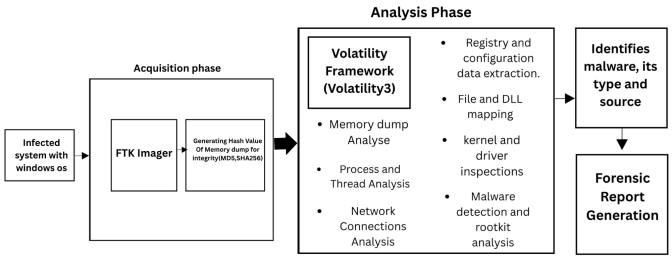

Fig. 1. Proposed Workflow for Volatile Memory Forensics and Encryption Artifact Identification

The ransomware simulator was developed to perform AES-based file encryption on selected files inside a designated folder. During execution, the simulator generated runtime

Paper	Focus Area	Technique Used	Contribution	Research Gap
MeMalDet [14]	Malware detection	Deep autoencoders + ensemble learning	Improved malware detection accuracy	Requires large datasets and high computation
VolMemLyzer [13]	Malware classification	Feature engineering + ML	Efficient malware classification	Limited deep learning integration
MemBERT [15]	Memory forensics	Transformer-based foundation model	Generalized memory representation	High computational complexity
Capturing Encryption Keys [16]	Encryption key recovery	RAM analysis	Recovery of cryptographic keys	Limited ransomware evaluation
Fileless Malware Analysis [10]	Fileless malware detection	Behavioral memory analysis	Analysis of evasive malware	Lacks automated detection
Signature-Based VMF [17]	Cyberattack detection	Signature scanning	Fast detection of known malware	Fails for unknown malware
Adaptive Ransomware Detection [12]	Ransomware detection	Pre-encryption detection model	Early ransomware detection	Needs real-world validation

TABLE I
COMPARISON OF EXISTING TECHNIQUES IN VOLATILE MEMORY FORENSICS

encryption artifacts including suspicious processes, encrypted files, and memory-resident cryptographic materials. The controlled execution environment enabled the acquisition of volatile memory images during active ransomware execution.

B. Memory Acquisition

Memory acquisition is one of the most critical stages in volatile memory forensics because RAM contents are temporary and lost after system shutdown. In this study, memory images were acquired while the ransomware simulator was actively executing in the virtual machine.

Several memory acquisition tools were studied and evaluated, including FTK Imager, WinPMEM, and DumpIt. FTK Imager was used due to its ability to acquire physical memory images while maintaining forensic integrity [19]. WinPMEM and DumpIt were also analyzed as alternative lightweight memory acquisition utilities commonly used in live forensic investigations [20]. Memory dump acquisition was performed

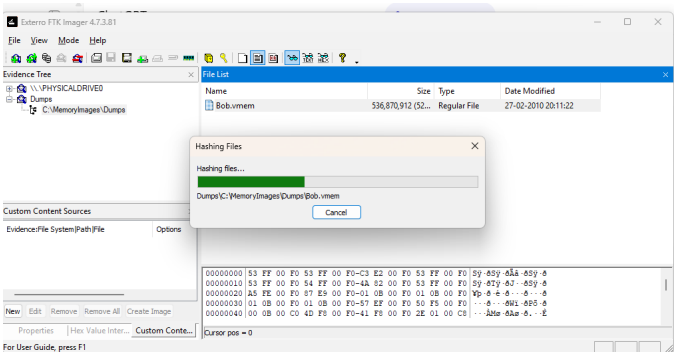

Fig. 2. Memory dump is captured with FTK Imager

using FTK Imager to capture the complete volatile memory contents of the running system. The captured RAM image preserves important runtime artifacts such as active processes, network connections, loaded DLLs, and encryption-related data for further forensic analysis.

The memory image was stored in raw dump format for further forensic analysis.

C. Memory Analysis Using Volatility Framework

The acquired memory dump was analyzed using the Volatility Framework, which is one of the most widely used open-source memory forensic frameworks [9]. Volatility provides plugins for extracting runtime artifacts from memory images and supports process analysis, DLL inspection, registry extraction, and network analysis.

Several Volatility plugins were used during the investigation, including:

- `windows.pslist` for active process identification
- `windows.pstree` for parent-child process analysis
- `windows.dlllist` for DLL inspection
- `windows.filescan` for file object analysis
- `windows.cmdline` for command-line investigation

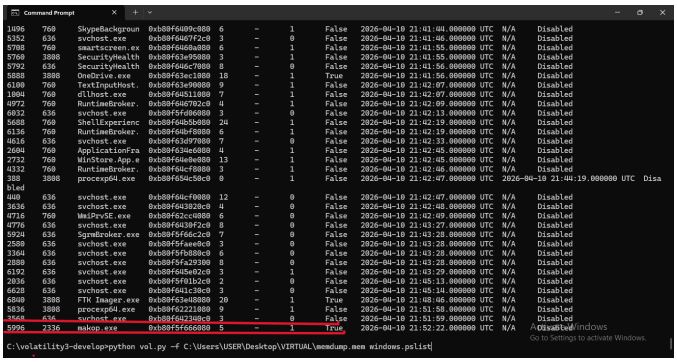

Fig. 3. Presence of makop Ransomeware is identified using volatility

The analysis successfully identified suspicious ransomware-related processes and their associated memory regions. Parent-child process relationships were examined to understand the execution flow of the ransomware simulator.

D. YARA-Based Memory Scanning

YARA is a pattern-matching tool widely used in malware analysis and digital forensics for identifying suspicious strings and binary signatures within files and memory dumps [21]. In this work, YARA rules were created to identify encryption-related artifacts and suspicious memory structures associated with ransomware execution.

The YARA scanning process involved:

- Defining hexadecimal and string-based signatures
- Scanning memory dumps for encryption-related patterns
- Identifying possible cryptographic materials
- Detecting suspicious executable regions

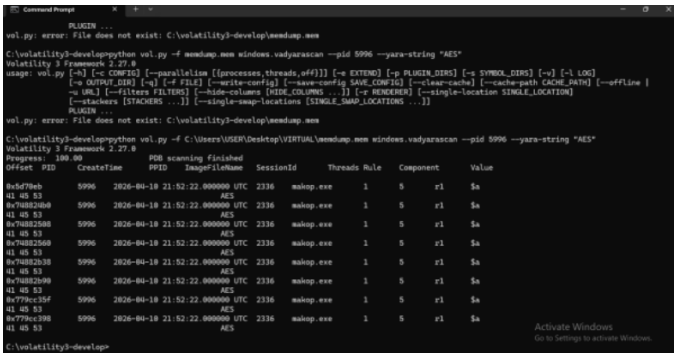

Fig. 4. yarascan searching for string aes.

YARA scanning helped locate hardcoded AES key patterns and suspicious memory regions linked to the ransomware simulator.

E. String Analysis

The `strings` utility was used to extract human-readable strings from the process memory dump. String analysis is useful for identifying filenames, commands, registry paths, URLs, encryption-related terms, and suspicious textual artifacts stored in memory.

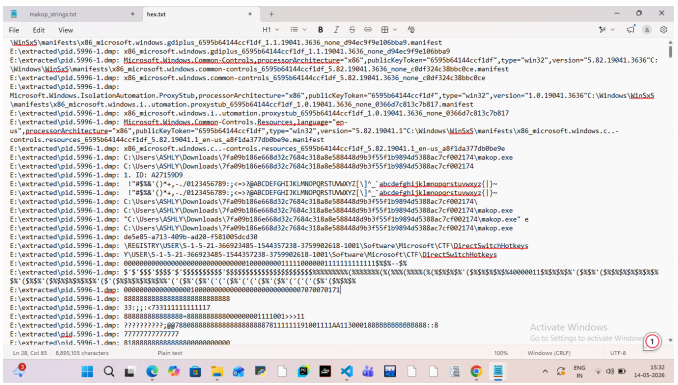

Fig. 5. Extracted memory strings from makop.exe process dump

The extracted strings provided important forensic evidence related to encrypted file names, runtime execution traces, temporary file paths, and encryption-related artifacts present within the memory dump. String analysis also assisted in identifying process-related information, suspicious execution behavior, and memory-resident traces associated with the ransomware simulator. The recovered textual artifacts helped validate the presence of ransomware activity and supported further forensic investigation of the compromised system.

F. Entropy-Based Memory Analysis

Entropy analysis was performed to identify highly random memory regions associated with encrypted data and cryptographic keys. Cryptographic materials such as AES keys typically exhibit high entropy due to their randomness characteristics [16].

A custom Python script was developed to scan the memory dump in 16-byte blocks and calculate the Shannon entropy of each block. The memory dump was divided into fixed 16-byte blocks because AES-128 encryption uses a 128-bit (16-byte) key size. Each block was analyzed individually to determine its entropy value.

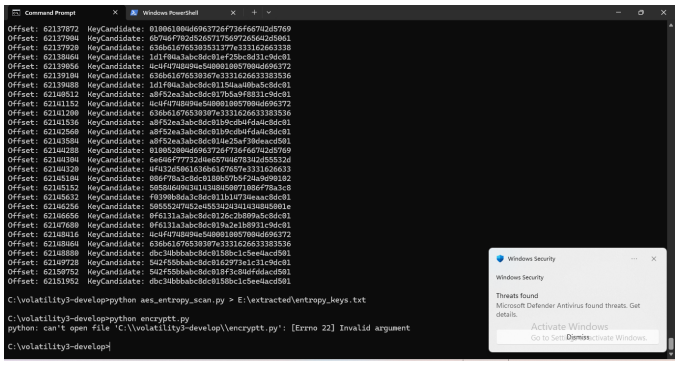

Fig. 6. Extracted possible candidate keys makop.exe process dump

The entropy scanning process involved reading the process memory dump and dividing the memory contents into 16-byte blocks corresponding to the AES key size. Shannon entropy was calculated for each block to measure the randomness of the data stored in memory. Blocks exhibiting high entropy values were filtered and analyzed further as potential cryptographic key candidates. Memory regions with entropy values greater than a predefined threshold were considered suspicious and likely to contain AES encryption keys or encryption-related artifacts.

G. AES Key Identification

AES encryption keys were identified from the suspicious high-entropy memory regions obtained during entropy scanning. Candidate key blocks extracted from memory were manually verified using hexadecimal analysis tools such as HexEd.it [22].

The AES key identification process included:

- Detection of high-entropy memory blocks
- Extraction of candidate hexadecimal values
- Verification of key length and randomness
- Validation through decryption testing

The extracted AES encryption key was successfully used to recover encrypted files generated by the ransomware simulator. The successful recovery demonstrated the effectiveness of volatile memory analysis in identifying encryption-related artifacts directly from RAM.

The overall methodology demonstrates how volatile memory forensics can be used for ransomware detection, encryption

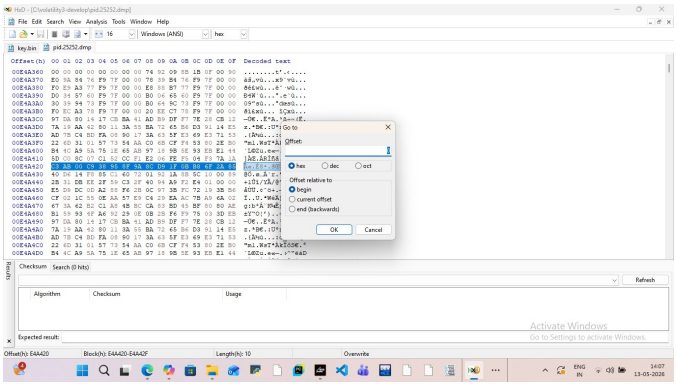

Fig. 7. Extracted AES key from Ransomware simulator process dump

artifact recovery, and cryptographic key identification during live forensic investigations.

IV. Experimental Results and Analysis

The proposed workflow for volatile memory forensic investigation is designed to identify ransomware activities and recover encryption-related artifacts directly from RAM. The workflow consists of multiple stages including ransomware execution, memory acquisition, forensic analysis, entropy-based key extraction, and file recovery. Figure 14 illustrates the overall architecture of the proposed system.

A. Execution of Ransomware Simulator

A controlled ransomware simulator was developed and executed inside a Windows virtual machine environment. The simulator encrypts files in the target folder using the AES encryption algorithm and generates runtime artifacts in memory. The simulator mimics ransomware behavior such as file encryption, process creation, and encryption key handling in RAM. Performing the experiment in an isolated virtual environment ensures safe analysis without affecting the host system. The screenshot folder initially contained normal

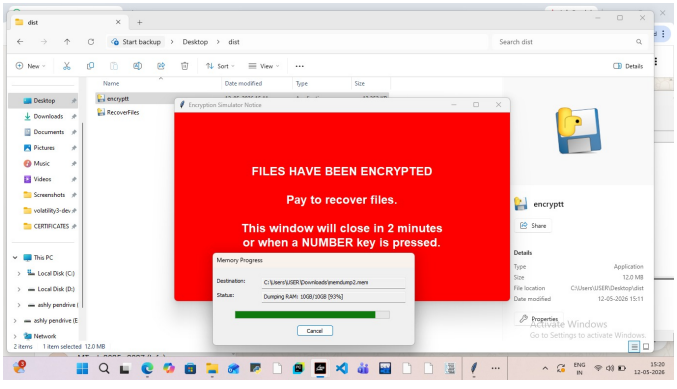

Fig. 8. Ransomware Simulation to encrypt files in screenshot folder.

files and images in an accessible state. Files and images in screenshot folder before encryption the ransomware encryption process was executed. These files served as the target dataset for observing encryption behavior and analyzing the impact of ransomware activity.

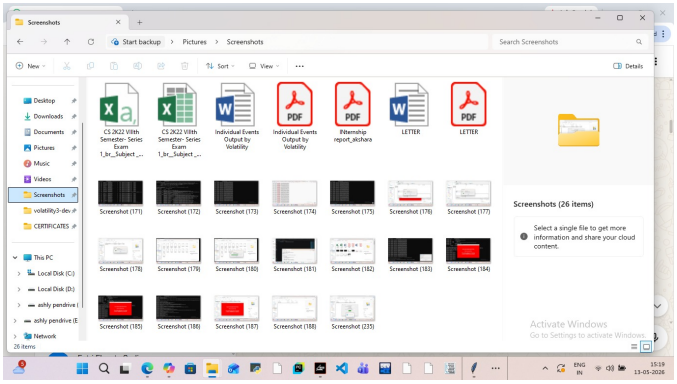

Fig. 9. Files and images in screenshot folder before encryption.

The files in the screenshot folder were successfully encrypted during ransomware execution, resulting in modified file content and loss of normal accessibility.

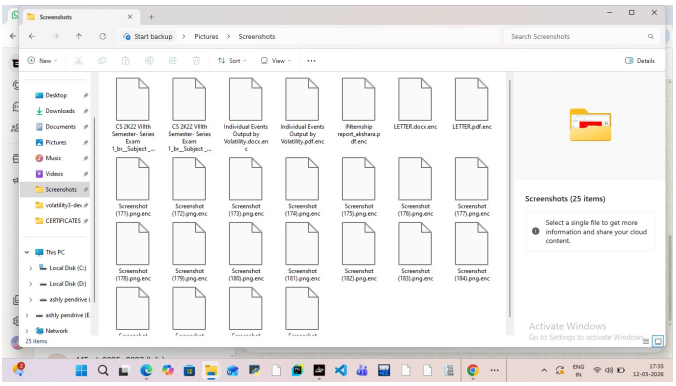

Fig. 10. Files and images in screenshot folder after encryption.

B. RAM Image Acquisition

During ransomware execution, the volatile memory image was captured using forensic acquisition tools such as FTK Imager. Memory acquisition was performed before shutting down the system because volatile data stored in RAM is lost once power is removed. The captured memory dump contains important runtime artifacts including running processes, DLLs, encryption keys, and memory-resident malware traces [9], [20].

C. Memory Dump Analysis

The acquired memory image was analyzed using the Volatility Framework. Volatility plugins were used to identify suspicious processes, process trees, loaded DLL modules, and memory regions associated with ransomware activity. Process analysis enabled the identification of the ransomware simulator process and its child processes during encryption execution [8], [13].

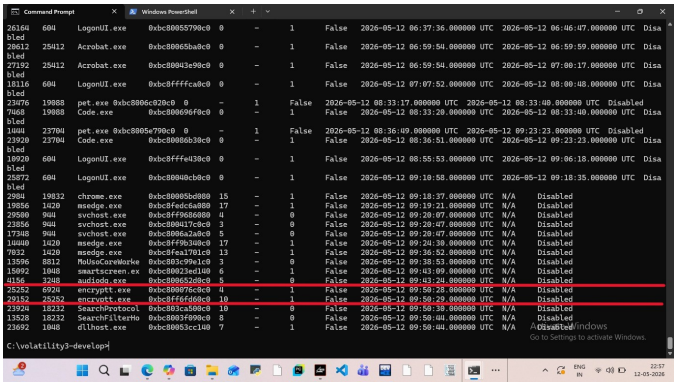

Fig. 11. Files and images in screenshot folder after encryption.

D. Detection of Suspicious Processes

Suspicious processes were identified using Volatility plugins such as `pslist`, `pstree`, and `malfind`. The analysis focused on detecting abnormal process behavior, hidden processes, injected code segments, and unauthorized memory regions. Process hierarchy analysis helped in tracing parent-child relationships associated with ransomware execution.

E. Recovery of Encryption Artifacts

Encryption-related artifacts were recovered from process memory using string extraction, YARA scanning, and manual forensic inspection. YARA rules were used to identify AES-related patterns and hardcoded key structures within memory dumps. Runtime traces such as encrypted file names, cryptographic buffers, and suspicious strings were successfully recovered from RAM [17], [21].

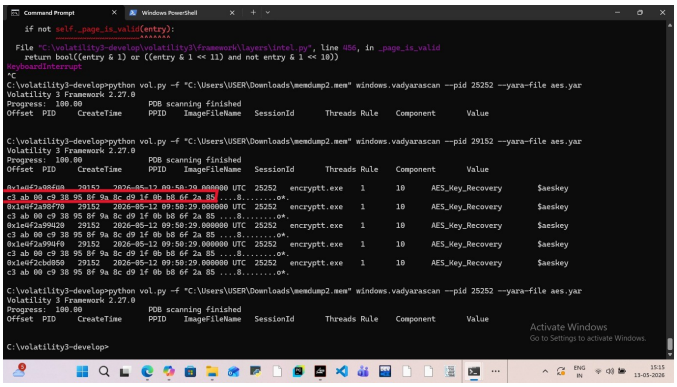

Fig. 12. Recovered Hardcoded Encryption Key from memory of child process.

F. AES Key Extraction Using Entropy Analysis

Entropy-based analysis was performed to identify potential AES encryption keys stored in memory. The process memory dump was divided into 16-byte blocks, corresponding to the AES key size. Each block was analyzed using Shannon entropy calculation to detect high-randomness regions associated with cryptographic material.

Blocks with entropy values above a predefined threshold were treated as potential AES key candidates. This approach enabled successful extraction of hardcoded encryption keys from the ransomware process memory [**?**], [16].

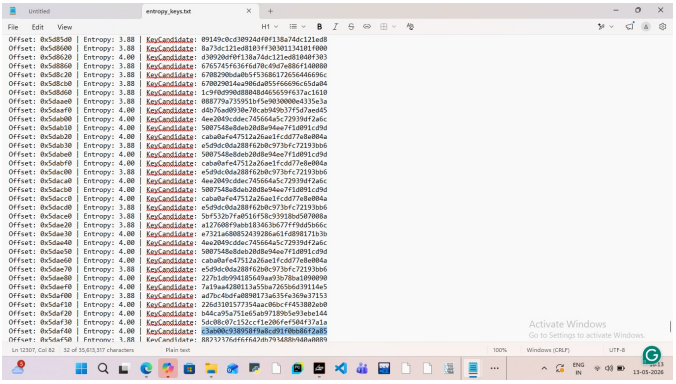

Fig. 13. Recovered Encryption Key through Entropyscan

G. Decryption and Recovery of Encrypted Files

The recovered AES key was used to decrypt encrypted files generated by the ransomware simulator. Successful recovery of encrypted files demonstrates the effectiveness of volatile memory forensics in ransomware investigations. The experiment proves that important forensic evidence such as encryption keys can remain in RAM during ransomware execution even when limited disk artifacts are available.

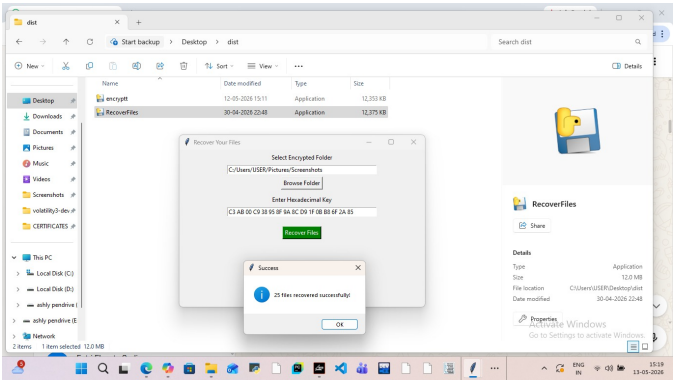

Fig. 14. Recovered Encryption Key through Entropyscan

V. Discussion

The experimental results demonstrate the significance of volatile memory forensics in ransomware investigations and malware analysis. Important forensic artifacts such as suspicious processes, runtime execution traces, encrypted file information, temporary file paths, and AES encryption keys were successfully recovered from memory. These artifacts provided valuable evidence regarding ransomware behavior and system compromise, even in situations where limited evidence was available on permanent storage devices.

Entropy-based analysis proved to be effective for identifying potential cryptographic material within process memory

dumps. By dividing memory into 16-byte blocks and calculating Shannon entropy values, high-randomness regions associated with AES encryption keys were detected successfully. The approach enabled the recovery of hardcoded encryption keys directly from volatile memory, demonstrating the usefulness of entropy analysis in encryption artifact identification and ransomware investigations.

Despite the successful recovery of forensic artifacts, volatile memory acquisition presents several challenges. Since RAM is highly dynamic and continuously changing during system execution, important evidence may be overwritten or lost before acquisition is completed. The acquisition process itself may also modify memory contents, potentially affecting forensic integrity. Additionally, capturing memory from systems under heavy workload or active malware execution can introduce inconsistencies within the acquired dump.

Modern malware and ransomware families increasingly employ anti-forensic techniques to evade memory analysis. Techniques such as process injection, reflective DLL loading, code obfuscation, encryption, memory wiping, and fileless execution make forensic investigation more difficult. Some malware samples may also terminate malicious processes or erase encryption keys from memory immediately after execution to reduce forensic traceability.

The proposed approach also has certain limitations. Entropy analysis may generate false positives because non-cryptographic random data can exhibit high entropy values similar to encryption keys. The methodology primarily focuses on AES-based ransomware simulations and may require modifications for detecting other encryption algorithms or advanced malware families. Furthermore, the effectiveness of forensic analysis depends heavily on timely memory acquisition and the availability of suitable forensic tools and plugins.

VI. Conclusion and Future Work

This study demonstrated the importance of volatile memory forensics in ransomware investigation and encryption artifact identification. The experimental analysis successfully identified suspicious processes, runtime execution traces, encrypted file information, and cryptographic artifacts directly from RAM. Memory analysis using the Volatility Framework, YARA scanning, string extraction, and entropy-based techniques proved effective for detecting ransomware-related activity within volatile memory.

One of the major findings of this work was the successful recovery of AES encryption keys from ranmsomware simulator process memory dumps. Entropy analysis using 16-byte block scanning enabled the identification of high-randomness memory regions associated with cryptographic material. The recovered AES key was successfully utilized to decrypt encrypted files generated during ransomware simulation, demonstrating the practical value of volatile memory analysis in digital forensic investigations and incident response.

The results emphasize that RAM forensics plays a critical role in analyzing modern ransomware and memory-resident malware that often leave limited traces on permanent storage. Volatile memory contains valuable transient evidence such as active processes, encryption keys, injected code, and runtime artifacts that can significantly assist forensic investigators during cybercrime investigations.

Future work can focus on integrating Artificial Intelligence and Machine Learning techniques for automated memory artifact classification and malware detection. Real-time memory monitoring systems can also be developed to continuously identify suspicious behavior and ransomware execution before large-scale encryption occurs and also finding out keys of encryption from a real ransomware attack where the key is only known to the attacker.

References

[1] N. Agarwal *et al.*, "Digital forensics: Challenges and future directions," *Journal of Cybersecurity*, 2023.

[2] S. Zawoad and R. Hasan, "Digital forensics in the cloud: A survey," *Digital Investigation*, 2015.

[3] M. Ruff, "Volatile memory analysis in digital forensics," in *DFRWS Proceedings*, 2011.

[4] A. Case, J. Levy, and A. Walters, *Memory Forensics: The Path to Volatility*, 2014.

[5] E. Gandotra *et al.*, "Fileless malware detection techniques: A survey," *Computers & Security*, 2020.

[6] K. D. Mandal, "Ransomware behavior and detection techniques," *IEEE Access*, 2021.

[7] Volatility Foundation, "The volatility framework documentation," https://www.volatilityfoundation.org, 2024.

[8] I. Hamid and M. M. H. Rahman, "A comprehensive literature review on volatile memory forensics," *Electronics*, vol. 13, no. 15, p. 3026, 2024.

[9] A. Case and G. G. R. III, "The volatility framework: Volatile memory artifact extraction utility framework," *Digital Investigation*, vol. 8, pp. S147–S155, 2011.

[10] A. Afreen, M. Aslam, and S. Ahmed, "Analysis of fileless malware and its evasive behavior," in *2020 International Conference on Cyber Warfare and Security (ICCWS)*. IEEE, 2020, pp. 1–8.

[11] K. Khaliq *et al.*, "Ransomware attacks: Tools and techniques for detection," in *2024 2nd International Conference on Cyber Resilience (ICCR)*. IEEE, 2024, pp. 1–5.

[12] U. Urooj *et al.*, "A proposed adaptive pre-encryption crypto-ransomware early detection model," in *2021 3rd International Cyber Resilience Conference (CRC)*. IEEE, 2021, pp. 1–6.

[13] A. H. Lashkari *et al.*, "Volmemlyzer: Volatile memory analyzer for malware classification using feature engineering," in *2021 Reconciling Data Analytics, Automation, Privacy, and Security*. IEEE, 2021, pp. 1–8.

[14] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "Memaldet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," *Computers & Security*, vol. 142, p. 103864, 2024.

[15] C. Fellicious *et al.*, "Membert: Foundation model for memory forensics," in *Proceedings of the 40th ACM SIGAPP Symposium on Applied Computing*, 2025, pp. 1772–1779.

[16] S. Balogh and M. Pondelik, "Capturing encryption keys for digital analysis," in *2011 IEEE International Conference on Intelligent Data Acquisition*. IEEE, 2011, pp. 759–763.

[17] N. R. Mistry and M. S. Dahiya, "Signature based volatile memory forensics: A detection based approach for analyzing sophisticated cyber attacks," *International Journal of Information Technology*, vol. 11, no. 3, pp. 583–589, 2019.

[18] Oracle, "Oracle vm virtualbox," https://www.virtualbox.org/, 2026.

[19] Exterro, "Ftk imager," https://www.exterro.com/digital-forensics-software/ftk-imager, 2026.

[20] W. Ahmed and B. Aslam, "A comparison of windows physical memory acquisition tools," in *2015 IEEE Military Communications Conference (MILCOM)*. IEEE, 2015, pp. 1292–1297.

[21] V. M. Alvarez, "Yara: The pattern matching swiss army knife for malware researchers," https://virustotal.github.io/yara/, 2013.

[22] HexEd.it, "Hexed.it - online hex editor," https://hexed.it/, 2026.