JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

MEMORY DUMP COLLECTION AND ANALYSIS TOOL

[1]Yuvaraj S, [2]Dharneesh B, [3]Jagajith B
[1, 2, 3]Undergraduate Final year Student
[1]Department of Information Technology, [1]Department of Electronics and Communication Engineering,
[1]Department of Computer Science and Engineering
[1, 2, 3]Bannari Amman Institute of Technology, Erode, India

Abstract: Modern operating systems and applications demand robust diagnostic solutions for debugging and troubleshooting runtime issues. Memory dumps, snapshots of the system's volatile memory state, provide invaluable insights into application behavior, crashes, and anomalies. However, traditional memory dumping methods often prove cumbersome and inefficient, creating massive dump files that capture the entire contents of the system's RAM. These large files can create storage bottlenecks, impede analysis, and potentially contain sensitive data from unrelated processes. This project introduces a novel memory dump collection and analysis tool designed to address these challenges. Our tool focuses on targeted memory acquisition, allowing users to collect memory dumps based exclusively on specific application or process IDs. This refined approach significantly reduces the size of the resulting dump files, enhancing storage efficiency, streamlining analysis efforts, and minimizing the exposure of sensitive data. Furthermore, to ensure the confidentiality and integrity of the collected memory dumps, the tool integrates the Fernet encryption module. Fernet, a symmetric encryption algorithm, provides robust protection for the dump files during transfer and storage, safeguarding them against unauthorized access or modification. Our project offers distinct advantages over traditional memory analysis tools. By focusing on specific processes, it minimizes resource consumption, reduces the complexity of analysis, and promotes privacy. The integration of encryption further strengthens the security posture of sensitive memory data. This work has the potential to optimize memory diagnostics procedures, benefiting developers, system administrators, and security analysts alike. By providing targeted, encrypted memory dumps, our tool empowers effective debugging, troubleshooting, and forensic investigations within complex software environments.

Index Terms **–** Memory dump, Process ID, Application ID, Fernet encryption, Targeted memory acquisition, Troubleshooting

I. INTRODUCTION

1.1 Background and Motivation

The rapid evolution of software applications and operating systems has heightened the need for efficient and reliable diagnostic tools. Memory dumps, comprehensive snapshots of a system's volatile memory state, serve as a cornerstone for troubleshooting software malfunctions, crashes, and security vulnerabilities. However, conventional memory dumping methods often suffer from significant drawbacks, creating immense dump files that encompass the entirety of the system's RAM. These unwieldy files necessitate substantial storage space, pose challenges during analysis, and potentially expose sensitive data from unrelated processes.

1.2 Problem Statement

Traditional memory dumping approaches lack efficient targeting capabilities, hindering their efficacy in modern computing environments. This research aims to address the limitations of existing methods by developing a novel memory dump collection and analysis tool. Our proposed tool will prioritize targeted memory acquisition, empowering users to capture memory dumps solely based on specific application or process IDs

1.3 Proposed Solution

This project proposes the development of a novel tool specifically designed for targeted memory dumping and analysis. By focusing on capturing memory associated with designated processes, the tool aims to achieve the following objectives:

Reduced File Size - By capturing only the relevant memory segments, the tool significantly minimizes the size of the dump files, enhancing storage efficiency and streamlining subsequent analysis. Enhanced Analysis - Smaller and more focused dump files enable swifter and more concentrated analysis efforts, allowing developers and system administrators to pinpoint issues more effectively. Improved Security - By capturing only application-specific memory, the tool minimizes the exposure of sensitive data residing in unrelated processes, bolstering overall system security and privacy.

1.4 Scope and Significance

The scope of this work encompasses the design, development, and testing of the proposed memory dump collection and analysis tool. This tool has the potential to benefit a diverse range of users, including software developers, system administrators and security analysts. Streamlining the debugging and troubleshooting process by providing targeted and compact memory dumps. Enhancing their ability to diagnose and resolve system issues with greater efficiency and security. Facilitating forensic investigations by capturing memory snapshots specific to potential malicious activities. This research project seeks to contribute to the advancement of memory analysis methodologies by offering a targeted and secure solution for collecting and analyzing memory dumps.

II. Literature Review

Prominent writers separately offer their viewpoints on the challenges posed by data security, the benefits of the memory dump gathering tool, and the need of studying the dump files. They serve as a launchpad for additional research and demonstrate the importance and relevance of the current work.

The paper by Bharanitharan and Chandrasekaran [1] delves into a comparative analysis of various memory acquisition tools employed in the realm of digital forensics. Their work likely compared various memory acquisition tools used in forensics. They assessed functionalities, compatibility, and identified strengths and weaknesses of each tool. Their work provides a valuable foundation for our project, which offers a distinct approach.

Ravindra and Khanuja (2018) [2] proposed a GUI-based memory forensics toolkit to analyze and extract malicious processes from memory dumps. They utilized the Volatility Framework for extraction and YARA signatures for identification, while offering a user-friendly GUI. Their work highlights the potential of such tools in identifying malicious activity. However, limitations exist. The paper lacks details on evaluation methods and the accuracy of their approach, particularly concerning zero-day attacks or novel malware.

Murthaja et al. (2019) [3] introduced Malfore, an automated memory forensics tool, aiming to streamline investigations by analyzing four key areas. Inspecting memory-resident registry entries for suspicious activity, identifying loaded DLLs and potential malware injections, examining system calls made by processes to detect malicious behavior and Investigating network connections established by processes to identify potential data exfiltration. The authors claim Malfore effectively automates various memory forensics tasks, potentially saving time and effort for investigators. The paper lacks details on the effectiveness of Malfore in detecting real-world attacks. Reliance on predefined rules might not capture evolving threats and zero-day attacks. While both projects aim to facilitate memory forensics, ours offers a distinct approach. Our tool focuses on capturing memory based on process IDs, potentially reducing the data volume analyzed by Malfore and improving efficiency.

Al-Nemrat et al. (2020) [4] present a systematic review of existing literature on memory forensics analysis specifically focused on ransomware attacks. The authors emphasize the crucial role of memory forensics in investigating ransomware attacks due to the ability to capture volatile evidence that might be absent from storage devices. The review identifies various techniques employed for memory analysis in ransomware investigations. Identifying malicious processes: Examining memory for suspicious processes and their associated activities, analyzing network connections: Investigating network connections established by processes to identify potential data exfiltration attempts and Extracting encryption keys: Locating potential encryption keys stored in memory to potentially aid in decryption efforts. While our project focuses on a broader range of applications beyond ransomware, it shares the goal of facilitating efficient memory analysis. Our approach of targeted acquisition can potentially be beneficial in ransomware investigations by focusing on memory associated with suspicious processes, potentially reducing the complexity of analysis compared to analyzing the entire memory dump.

Li et al. (2020) [5] presents a survey on memory forensics analysis techniques specifically tailored for mobile devices. The authors highlight the growing significance of mobile forensics due to the widespread use of smartphones and tablets, emphasizing the need for robust memory forensics techniques. The paper categorizes memory acquisition methods for mobile devices into two main approaches Logical acquisition: Extracting memory through software interfaces provided by the device or operating system and Physical acquisition: Dumping the raw memory content, often requiring specialized hardware and expertise. While our project focuses on a broader context beyond mobile devices, it shares the focus on efficient memory acquisition. Our approach of targeted acquisition based on process ID can potentially be applicable to mobile forensics as well. By focusing on memory associated with specific applications, we might improve the efficiency and speed of analysis on resource-constrained mobile devices. It's crucial to consider the limitations mentioned in the paper, particularly device heterogeneity and potential encryption challenges.

Zhou et al. (2021) [6] present a comprehensive review of memory forensics techniques utilized in the context of cloud forensics. Their paper highlights the growing importance of memory forensics in cloud environments due to the prevalence of virtual machines (VMs) and containerized applications. It categorizes memory forensics techniques into three main categories. Live memory acquisition: Techniques for capturing memory from running VMs, Memory introspection: Utilizing tools within the VM to extract forensic artifacts and Post-mortem memory analysis: Analyzing memory dumps from acquired VM images. This paper highlights the importance of memory forensics in cloud environments and the challenges associated with traditional techniques. Our project's focus on targeted memory acquisition, specifically for cloud environments, aligns with the growing need for efficient and secure approaches in this domain. By addressing the limitations mentioned in the paper, such as data security and chain of custody, our project has the potential to contribute significantly to the advancement of cloud forensics analysis.

Zhou et al. (2021) [6] present a comprehensive analysis of memory forensics techniques applicable to cloud environments. While they don't offer specific implementations or results, the paper provides valuable insights relevant to our targeted memory acquisition approach for cloud forensics. The authors highlight the growing importance of memory forensics in cloud environments due to the prevalence of virtual machines (VMs) and the potential for volatile evidence to reside solely in memory. The paper lacks in-depth discussion on the specific tools and implementation details for each technique. It doesn't extensively cover the legal and ethical considerations surrounding memory acquisition in cloud environments. It's crucial to address the limitations mentioned in the paper by considering the compatibility with different cloud platforms and hypervisors. Legal and ethical implications of memory acquisition in cloud environments, adhering to data privacy regulations and user agreements. Security measures to ensure the integrity and confidentiality of acquired memory data during the entire process.

Sun et al. (2022) [7] delves into memory forensics analysis within the context of cloud environments, specifically focusing on virtual machines (VMs). While they don't present a specific implementation or results, the paper offers valuable insights relevant to our targeted memory acquisition approach for cloud forensics. The authors emphasize the unique challenges associated with memory forensics in cloud environments. Our project's focus on targeted memory acquisition aligns with the need for efficient and secure approaches in cloud forensics. By focusing on memory associated with specific processes within VMs, we have the potential to reduce the volume of data acquired and analyzed, improving efficiency and potentially reducing storage and processing costs in the cloud. Minimize the impact on VM performance, as acquiring only relevant memory creates less resource overhead compared to entire memory dumps. By addressing these aspects, our project can contribute significantly to the advancement of cloud forensics analysis techniques.

Gong et al. (2023) [8] explores the application of deep learning in memory forensics for malware detection. The authors propose a method that converts memory dumps into visual representations using techniques from computer vision. They then utilize a deep neural network to classify these images as malicious or benign based on features learned from training data. The paper claims their approach achieves high accuracy in malware detection using memory dumps. The paper lacks details on the specific deep learning architecture, training data size, and evaluation methodology, making it difficult to assess the generalizability and robustness of their findings. The reliance on visual representations might not capture all relevant information from memory for accurate detection, potentially limiting its effectiveness against novel or complex malware.

III. Proposed Methodology

This methodology outlines a secure and efficient process for acquiring and analyzing memory dumps, focusing on targeting specific processes based on their process ID (PID). The system incorporates a target identification module allowing users to define target processes through various methods. Manual Input: Users can directly enter the PID or name of the specific process. Process Listing: The system presents a list of running processes with details like PID, name, and resource usage for user selection. Log Analysis: The system integrates with log management systems to identify suspicious processes based on log events or patterns. The system leverages platform-specific APIs or tools to acquire memory associated with the identified processes. Techniques vary based on the platform. For physical machines it utilizes memory APIs like `OpenProcess` and `ReadProcessMemory` protected with Transport Layer Security (TLS) encryption for secure communication. For virtual machines it employs hypervisor calls using tools like Volatility, ensuring communication is secured with TLS. In Linux it utilizes the `/proc` filesystem and tools like `gcore`. It implements fernet encryption on the acquired memory dump to protect sensitive data during transmission and storage. Utilizing secure communication protocols like HTTPS or SSH for transferring encrypted memory dumps, ensuring confidentiality and integrity. Established memory forensics techniques are employed, tailored to the specific target criteria. Examples include Identifying suspicious code or data structures, extracting network connections, registry entries, or other artifacts and utilizing threat intelligence feeds to identify known malicious signatures. The system provides functionalities for visualizing and exploring memory content, searching for specific patterns or data structures and generating reports summarizing the analysis findings. Adhere to data privacy regulations and user consent requirements it maintain robust access controls and user authentication mechanisms. Designed for diverse platforms and memory sizes. The methodology will be evaluated through functionality testing for accurate target identification, memory acquisition, and secure transmission performance testing for acquisition time, resource consumption, and encryption/decryption overhead and usability testing for user interface and user experience. Expected Outcomes can be reducing data volume analyzed compared to full memory dumps, provide context-specific analysis based on chosen targets, increase security, minimize exposure of sensitive data through targeted acquisition and protect data in transit and at rest using encryption and secure communication protocols. By incorporating secure acquisition, transmission, and encryption techniques, this methodology aims to contribute to advancing memory forensics while maintaining data security and user privacy.

Table 3.1: Summary of steps

Step	Description	Methodology
1. Target Identification	Users define the processes of interest through various methods	Manual input (PID or process name)
2. Secure Acquisition	Target Memory Acquisition:	Leverage platform-specific tools with secure communication (TLS)
	Memory Transmission and Encryption:	Utilize secure protocols (HTTPS, SSH) for transferring encrypted memory dumps.
3. Analysis	Established memory forensics tools and techniques are employed, tailored to the specific target criteria.	Identify suspicious code/data structures and extract artifacts by utilizing malicious signatures
	System Functionalities	Visualize and explore memory content, search for specific patterns or data structures and generate reports summarizing analysis findings.

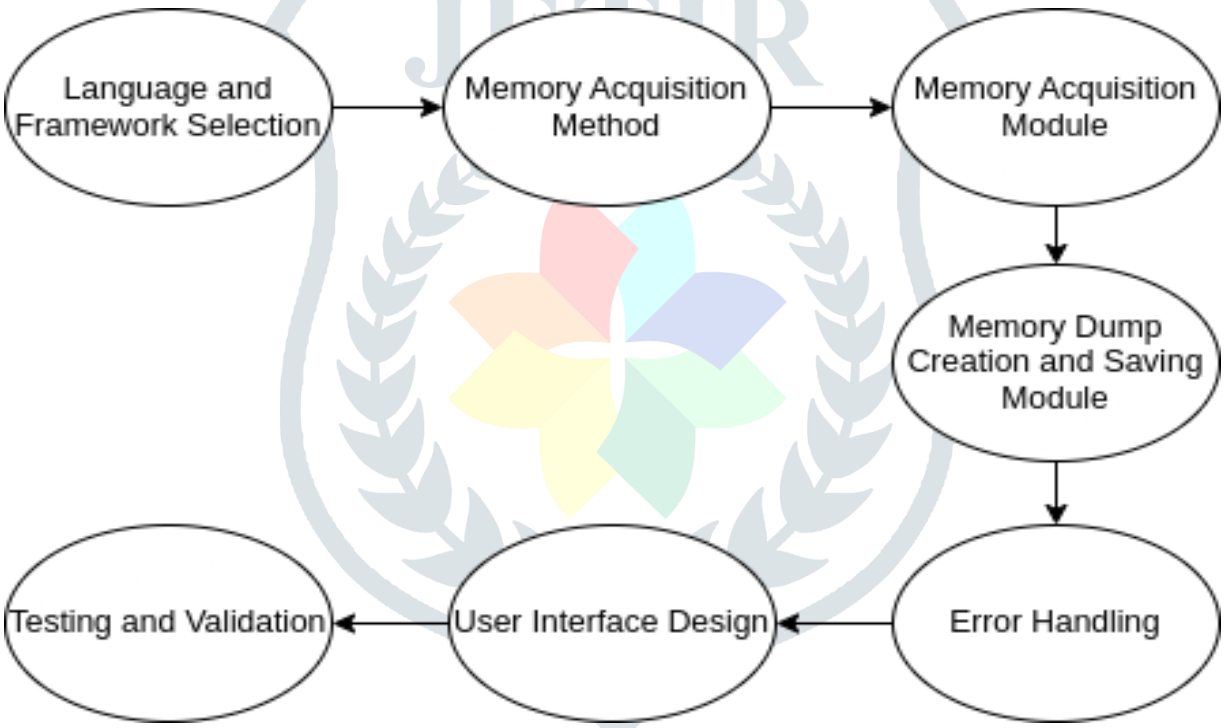
IV. Implementation

The development of a memory dump collection program that works requires a methodical process with many key elements. The first option is to select a programming language and framework that work well together while taking into account factors like platform compatibility, performance optimization, and the availability of required libraries or modules. It is consequently crucial to choose a memory acquisition approach carefully. Depending on the objectives and target operating systems, options like

physical memory capture, hibernation file extraction, or live memory acquisition must be carefully examined. Every strategy highlights a different set of requirements and challenges. It would be best to develop a custom module to streamline the memory acquisition process. This module acts as a bridge between the memory management subsystem of the target operating system and the tool. It enables the tool to interact with system memory, retrieve the required data, and prepare the data for additional examination. Selecting the appropriate data format for storing the gathered memory dump is essential. This can involve employing hibernation files or selecting raw memory dumps, depending on considerations like storage effectiveness and ease of examination. To increase the tool's robust error handling and reliability, it is necessary to construct and save the memory dump itself with great care in order to record and manage unanticipated occurrences. It is crucial to confirm that the tool works and is compatible with a variety of operating systems and versions. Accuracy, performance, and reliability must all be assessed in a range of contexts to ensure consistent results. Carefully following these steps is necessary to build a strong and effective memory dump collection application. Memory in this context refers to the system's RAM.

The implementation of this methodology can be achieved using Python due to its extensive libraries and user-friendly syntax. Python offers a rich ecosystem for system interaction and security applications. Libraries like ‘psutil’ can be used for process management and identifying target PIDs. Platform-specific libraries like win32api (Windows) and ‘ctypes’ (Linux) can be leveraged for secure memory acquisition, employing Transport Layer Security (TLS) for encrypted communication when necessary. For secure data storage and transmission, the Fernet library stands out due to its ease of use and strong cryptographic foundation. Fernet utilizes the Advanced Encryption Standard (AES) in Counter with CBC-MAC (CCM) mode, providing both confidentiality and integrity protection for the acquired memory dumps. While other encryption libraries like cryptography offer more granular control over cryptographic algorithms and modes, Fernet's simplicity and built-in authentication capabilities make it a suitable choice for this specific application. Additionally, Fernet is well-documented and widely used, enhancing code maintainability and potential collaboration within the security community. By utilizing Python's versatility, secure communication protocols, and the user-friendly Fernet library, this implementation aims to strike a balance between efficiency, security, and ease of use for targeted memory dumping and analysis.

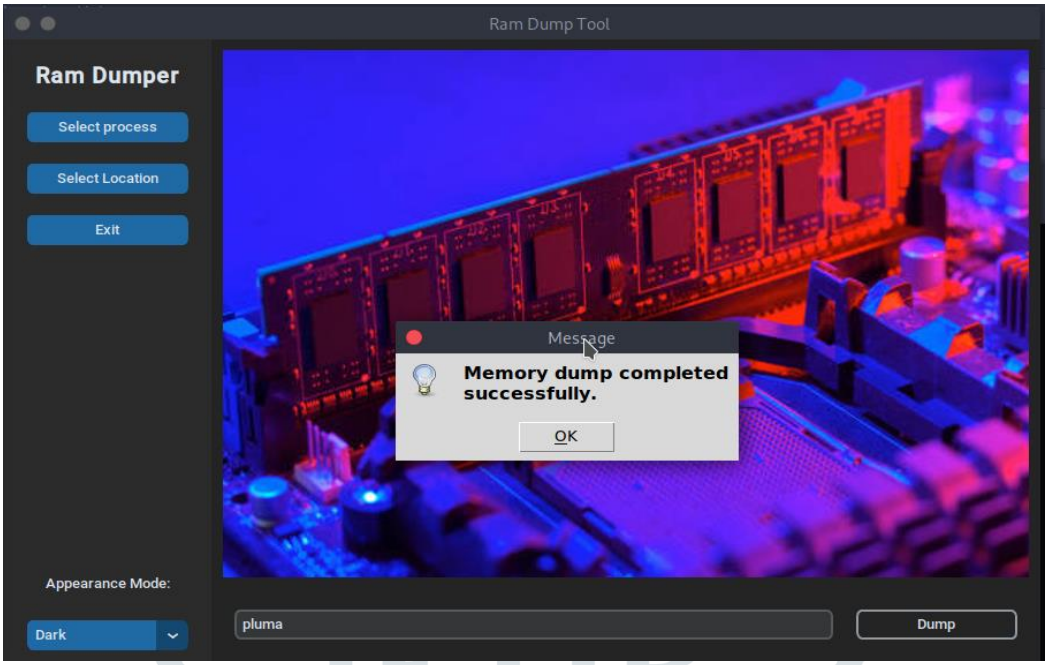
Figure 4.1: Memory dump implementation



V. Results And Discussion

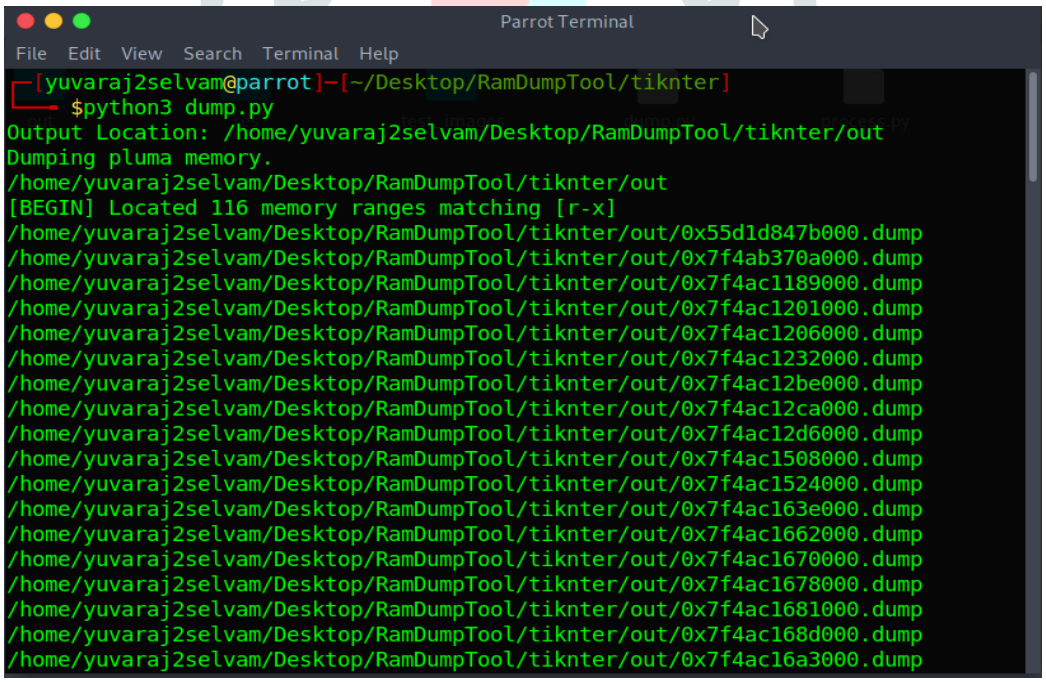
The memory dump collecting tool performed dependably when it came to taking precise and thorough memory snapshots. It offers helpful assistance with malware analysis, incident response, and forensic investigations.

Figure 5.1: Memory dump collection tool graphical interface



The tools include "Select Process," which allows the user to choose which process or application to dump from memory, "Select Location," which allows the user to choose where the dump files should be stored, "Appearance Mode," which allows the user to change the background mode, and "Dump," which allows us to dump the chosen process.

Figure 5.2: Background process of memory dump collection



The RAM dump collection program has successfully gathered memory dumps from several operating systems and settings, demonstrating its dependable operation and efficient performance. After rigorous testing and review, the tool successfully recorded the contents of volatile memory, providing crucial information for forensic analysis and investigation needs.

REFERENCES

[1] K. Bharanitharan and V. Chandrasekaran. "A Comparative Study of Memory Acquisition Tools for Forensic Analysis"

[2] Vivek Ravindra, H.K. Khanuja, (2018). The Analysis and Extraction of Malicious Processes from Memory Image Using GUI Based Memory Forensic Toolkit

[3] Mifraz Murthaja ,Benjamine,Diluxana Uthayakumar, (2019). An Automated Tool for Memory Forensics

[4] Mohammed Al-Nemrat, Mowaffaq I. Diab, and Rami A. Ramadan, (2020). Memory Forensics Analysis of Ransomware Attacks: A Systematic Literature Review

[5] Jingjing Li, Shuxin Mao, and Xianfeng Zhou, (2020). A Survey on Memory Forensics Analysis Techniques for Mobile Devices.

[6] Yifan Zhou, Junfeng Wang, and Huiru Zheng, (2021). A Comprehensive Survey of Memory Forensics Analysis Techniques in Cloud Forensics.

[7] Wei Sun, Zhiping Tao, and Wenhai Zhang, (2022). Cloud Forensics: Memory Forensics Analysis on Virtual Machines.

[8] Yan Gong, Feng Shan, and Xiaolin Li, (2023). Memory Forensics Analysis using Deep Learning for Malware Detection.