Scope, Planning, and Design Artefacts (SPDA) Report

**Project - Memory Forensics and Volatile Artefact Analysis**

**Project Team #9**

|  |  |
| --- | --- |
| **Team Member** | **Student ID** |
| Aashish Anand | 645876 |
| Joshua Crisford | 574082 |
| Emily Tinsley | 652275 |
| Poorvika Kamberkavi | 641799 |

Table of Contents

[1. Introduction 3](#_Toc209783388)

[2. Project Scope 4](#_Toc209783389)

[2.1 Scope Statement 4](#_Toc209783390)

[2.2 Scope of Work 4](#_Toc209783391)

[3. Planning & Timeline 6](#_Toc209783392)

[3.1 High-Level Timeline (Gantt Chart) 6](#_Toc209783393)

[3.2 Task Allocation 7](#_Toc209783394)

[3.3 Project-specific deliverables 7](#_Toc209783395)

[4. Design Artefacts 9](#_Toc209783396)

[4.1 Component Diagram 9](#_Toc209783397)

[4.2 Sequence Diagrams 10](#_Toc209783398)

[5. Scope Improvement Ideas 11](#_Toc209783399)

[6. Conclusion 12](#_Toc209783400)

[7. References 13](#_Toc209783401)

## 1. Introduction

Volatile memory is the computer memory that requires a continual power supply to retain information. The common example of volatile memory is Random Access Memory (RAM). When a computer is shut down or the power supply is otherwise cut off, the information held in the volatile memory is lost, meaning the data must be extracted from the device before the device is shut down. Volatile memory allows a computer to operate at higher speeds as it reads the desired data from the hard drive and temporarily stores it in a smaller, more accessible location, where alterations to the data can be made before saving the data to the hard drive. However, this memory provides forensic evidence on what documents a user had open, what software was running and other recently accessed information, all of which may build a case for or against a person.

This document will provide detailed information on the project objective, scope, planning, and deliverables for the memory forensics and volatile artifact analysis project. This project investigates the reasons for analysing volatile memory, the methods and software required to successfully analyse volatile memory without compromising the integrity of evidence and demonstrates the extraction of volatile memory. All information discovered and achievements completed during this project will be presented at the end of this project.

## 2. Project Scope

### 2.1 Scope Statement

This project aims to design and implement a controlled memory forensics workflow using open-source software such as Volatility 3 or Rekall. The particular focus of the project is on acquiring and analysing volatile memory images from virtualised test systems to identify forensic artefacts. These artefacts may include running processes, user sessions, network connections, and other signs of malicious activity. The project will be limited to safe, ethical, and controlled environments; therefore, no live systems will be used. The final results will include a forensic analysis report, a memory artefact cheat sheet, and a presentation on findings.

### 2.2 Scope of Work

|  |  |  |
| --- | --- | --- |
| Requirement | Description | Possible Test Case |
| Configure VM environment | Build a VM dedicated to RAM acquisition and analysis (ensure isolation from the host system). | Confirm VM can be reset without affecting the host. |
| Install/ validate acquisition software | Install and configure DumpIt, Belkasoft RAM Capturer, or AVML, depending on the OS and verify functionality. | Run a test capture and confirm that an output file is created, and verify the output file is readable in Volatility/Rekall |
| Log acquisition metadata | Record the tool used, version, system info, timestamp, and hash in the acquisition log. | Verify metadata completeness and reproducibility between multiple dumps. |
| Inject harmless malware into VM memory | Inject harmless payloads into memory to simulate threats in the real world and test detection accuracy. | Confirm injected artefacts are picked up and logged without disrupting the system, and confirm it does not alter unrelated system functionality |
| Acquire memory image | Perform a RAM dump from the VM and generate a cryptographic hash to preserve integrity. | Recalculate the hash and confirm it matches the original. |
| Compare multiple memory snapshots | Add logic to compare memory dumps and highlight variation in processes, modules, and artefacts. | Take two RAM snapshots (baseline vs after malware injection) and confirm differences in running processes (e.g., new process malware.exe), loaded modules, or network connections are correctly identified and annotated. |
| Analyse memory image | Run plugins (pslist, netscan, cmdline, clipboard) to extract system data. | Confirm expected processes, connections, and commands appear in plugin outputs |
| Identify suspicious artefacts | Interpret plugin results to highlight artefacts. | Check suspicious processes against the baseline to confirm the artefact. |
| Correlate findings into a timeline | Order artefacts chronologically to create a forensic narrative of the system’s state. | Verify timeline consistency. |
| Automate analysis with Python script | Develop a script to parse snapshots, run plugins, extract artefacts, and generate reports. | Confirm script produces reproducible output consistently and reduces analysis time compared to manual steps |
| Validate Forensic Tool Outputs | Cross-check plugin outputs for consistency and accuracy (e.g., compare results from pslist vs psscan, or netscan results against system logs) to ensure forensic reliability. | Run two plugins that return overlapping information (such as pslist and psscan) and confirm that process IDs, names, and parent-child relationships match across outputs. |
| Develop an artefact cheat sheet | Produce a quick reference mapping of Volatility/Rekall commands to outputs and their usage. | Check the cheat sheet covers all plugins/tools used in analysis. |
| Produce forensic analysis report | Summarise findings, suspicious activity and artefacts, and conclusions, including supporting screenshots. | Confirm report includes all required sections and references artefacts. |
| Create presentation slides | Design stakeholder presentation that highlights project workflow, artefacts, and outcomes. | Validate slides, include workflow diagrams, screenshots, and key findings. |

|  |  |  |  |
| --- | --- | --- | --- |
| Stage/Category Key | | | |
| **Setup** | **Acquisition** | **Analysis** | **Documentation** |

## 3. Planning & Timeline

### A screenshot of a computer AI-generated content may be incorrect.3.1 High-Level Timeline (Gantt Chart)

The following high-level timeline describes the planned sequence for the Memory Forensics project from week 7 to week 13. It links the requirement identified in the Scope of Work and confirms that all deliverables take place in a structured and sequential approach. It will guide task scheduling and provide a benchmark of project progress in upcoming stakeholder meetings.

#### 3.1.1 Explanation:

The tasks are divided into Setup, Acquisition, Analysis, and Documentation stages, each marked with milestones.

**Setup Stage (Weeks 7-8)**

The virtual environment is configured, and acquisition tools are validated. Completion of this stage ensures a stable baseline for controlled memory capture.

**Acquisition Stage (Weeks 8-9)**

Memory images are collected from the VM, with metadata logged and integrity validated using cryptographic hashing. This ensures the forensic soundness of collected evidence.

**Analysis Stage (Weeks 9-12)**

Extracted memory artefacts (processes, clipboard, command history, network connections) are analysed. Suspicious artefacts are identified, compared against baseline data, and correlated into a timeline to simulate an incident narrative.

**Documentation Stage (Weeks 11-13)**

Project outcomes are consolidated into a forensic report, a memory artefact cheat sheet, and stakeholder presentation slides. This ensures findings are communicated clearly and professionally.

### 3.2 Task Allocation

Work is split evenly amongst team members and in a way where no group member is continually doing work independently for more than two weeks, no group member is managing more than two independent tasks in any given week, and no group member is left without work for more than one week. The collaborative tasks are tasks that require creative analysis of information, and would thus benefit from multiple interpretations and perspectives, or require continuous communication, including formulating the report, preparing presentation slides, and rehearsing the presentation. This task allocation is subject to change as an unpredicted situation may arise for a team member, or a team member may experience an increase in workload from other subjects.

|  |  |
| --- | --- |
| **Team Member Name** | **Allocated Task** |
| **Josh** | Configure VM Environment |
| Install/Validate Acquisition Software |
| Extract Command History and Clipboard Data |
| Correlate Findings into Timeline |
| **Aashish** | Acquire Memory Image |
| Validate Forensic Tool Outputs |
| Draft Forensics Analysis Report (Introduction and Methodology) |  |
|  | Draft Memory Artifact Cheat Sheet |
| **Emily** | Inject Harmless Malware into Memory |
|  | Log Metadata and Hash Verification |
| Run Artifact Extraction Software |
| Refine Cheat Sheet |
| **Poorvika** | Compare Multiple Memory Snapshots |
| Stretch Objective Justification |
| Write Python Script to Automate Analysis of Memory |
| **Collaborative Work** | Identify Suspicious Artifacts |
| Draft Forensics Analysis Report (Findings and Screenshots) |
| Prepare Presentation Slides |
| Prepare Presentation Script |
| Finalise Report and Rehearse Presentation |

|  |  |  |  |
| --- | --- | --- | --- |
| Stage/Category Key | | | |
| **Setup** | **Acquisition** | **Analysis** | **Documentation** |

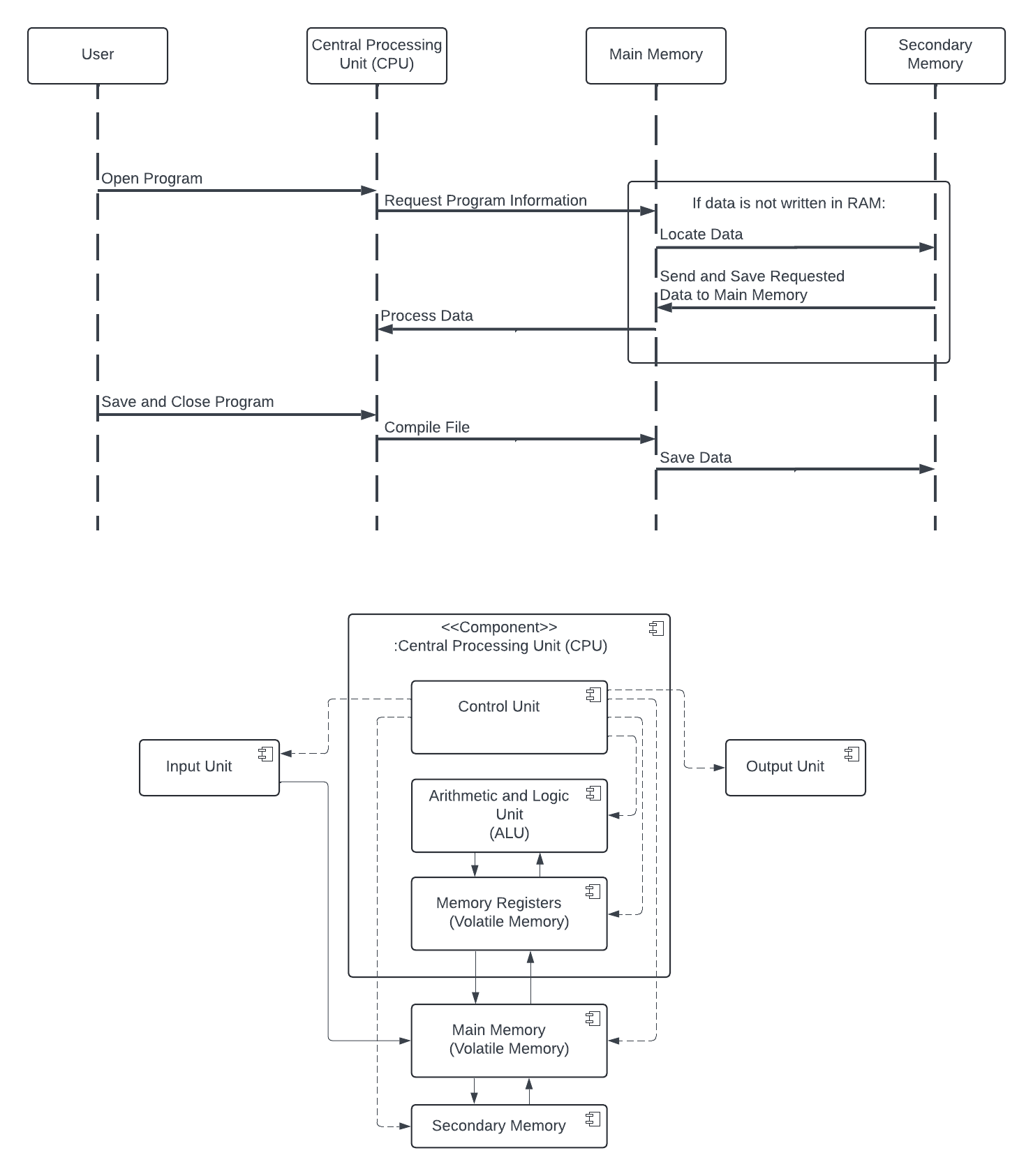
### 3.3 Project-specific deliverables

The deliverables are tailored to this memory forensics project and reflect industry-style DFIR outputs:

|  |  |  |
| --- | --- | --- |
| Deliverable | Description | Linked Stage |
| Validated Virtual Environment | Configured VM with acquisition tools installed and tested for stability and isolation. | Setup |
| Acquisition Logs & RAM Images | Memory dumps captured from VM, with metadata including hash values, tool details, and timestamps. | Acquisition |
| Analysis Outputs | Volatility/Rekall plugin results (e.g., pslist, netscan, cmdline, clipboard) showing processes, users, network sessions, and other artefacts. | Analysis |
| Forensic Timeline | Chronological narrative of system activity built from extracted artefacts, highlighting suspicious events. | Analysis |
| Memory Artefact Cheat Sheet | Quick-reference mapping of Volatility/Rekall commands to artefact types and investigative value. | Documentation |
| Forensic Analysis Report | Comprehensive report including introduction, methodology, findings, screenshots, suspicious activity analysis, and conclusions. | Documentation |
| Stakeholder Presentation Package | Slide deck and script summarising workflow, artefacts, findings, and project outcomes. | Documentation |
| Comparative Snapshot Analysis Module | Side-by-side comparison script or tool for several memory dumps with annotated differences and anomaly indicators. | Analysis |
| Malware Injection Test Suite | Controlled introduction of harmless malware samples to memory with payload behaviour records and detection results. | Acquisition / Analysis |
| Python Automation Script | Snapshot parsing, plugin execution, anomaly discovery, and report script with guide and source code annotated. | Analysis / Documentation |
| Updated Audit Logs | Stretch objective component execution logs with script execution, injection actions, and comparative analysis operations. | Documentation |

## 4. Design Artefacts

### 4.1 Component Diagram



The diagram above is a high-level component diagram of a computer that shows where volatile memory can be found. Before any memory is stored in the long-term secondary memory, it is created, read, and altered in the internal and main memory. When a computer is shut down and the volatile memory components stop receiving power, all information stored in volatile memory is destroyed (GeeksforGeeks, 2025). In the case that a computer enters sleep mode, all information stored in volatile memory is saved in a hibernation file. This file is persistent after power is off but will be overwritten the next time the device enters sleep mode (Microsoft, 2025).

Volatile memory holds memory that has been recently accessed and is predicted to be required again soon. By reading requested data and storing the memory address or actual data in a quicker to read and access location, the computer does not need to read the entire hard drive each time it must check information, nor does it have to permanently save a variable that is only required for a short period of time (GeeksforGeeks, 2025). This memory allows digital forensics to construct a timeline of the most recent activity that a given computer did and potentially form a case for or against a person.

### 4.2 Sequence Diagrams

As per the project specification, sequence diagrams are required to be submitted individually. Each member will develop sequence diagrams for their given requirements, including standard workflows and rainy-day scenarios, as well as possible test cases.

To minimise duplication, these diagrams will not be included in this submission. Rather, each member will submit their own diagrams separately as a part of their individual SPDA deliverable.

## 5. Scope Improvement Ideas

To enhance the scope beyond the base specification, the following improvements are proposed:

* **Comparative Analysis:** Capture memory at different points (baseline vs simulated incident) to strengthen findings and illustrate differences.
* **Advanced Artefact Correlation:** Use multiple Volatility plugins together (e.g., pslist + netscan + malfind) to detect stealthier threats.
* **Automation Extension (*Optional*):** Develop simple Python scripts to batch-run plugins and filter anomalies, simulating SOC automation practices.
* **Admissibility Focus:** Incorporate chain of custody considerations in acquisition logs to strengthen legal defensibility of evidence.
* **Extract volatile memory from hibernation file:** when a device enters sleep mode, volatile memory is stored in a hibernation file. Demonstrate that it is possible to extract volatile memory from this hibernation file.

#### Post-Stakeholder Meeting Outcome:

The team will implement all improvements except chain of custody considerations, as agreed with the stakeholder. These enhancements aim to provide more comprehensive analysis, increase detection accuracy, and improve the efficiency of memory forensics workflows.

## 6. Conclusion

This project aims to demonstrate the importance of extracting and analysing volatile memory and to follow a proper procedure with the correct software. By following the correct procedure, it is ensured that the integrity of evidence is maintained and that no live systems are unintentionally involved in memory extraction. The timeline of this project is split into 4 major parts, segments being setup, acquisition, analysis, and documentation. Finally, all findings and documentation will be presented at the end of the project, alongside the deliverables, to demonstrate the level of success this project had and to review areas of improvement.

## 7. References

GeeksforGeeks, 2025. Components of a Computer. [Online]   
Available at: <https://www.geeksforgeeks.org/computer-organization-architecture/computer-and-its-components/>   
 [Accessed 10 September 2025].

GeeksforGeeks, 2025. Difference Between Register and Memory. [Online]   
Available at: <https://www.geeksforgeeks.org/computer-organization-architecture/difference-between-register-and-memory/>   
 [Accessed 10 September 2025].

Microsoft, 2025. System power states. [Online]   
Available at: <https://learn.microsoft.com/en-us/windows/win32/power/system-power-states>   
 [Accessed 10 September 2025].