Lab-Manual

Digital Forensics

CY-334L

Prepared By: Ms.Memoona Sadaf

Instructor: Ms.Memoona Sadaf

Lab/Teaching Assistant: Muhammad Ahmad Ali Qureshi

Air University Islamabad

AIR UNIVERSITY

Department of Cyber Security

Lab Schedule

Week	Lab Topics
Week 11	**Memory Forensics**

🧠 Lab Manual: Memory Forensics (Complete Guide for Students)

---

1. What is Memory Forensics?

Memory forensics is a part of digital forensics that focuses on examining the **RAM (Random Access Memory)** of a computer. RAM is the temporary memory where a system stores data while it is running. It includes real-time information about running programs, user activities, system processes, open network connections, and sometimes even malware.

When a cyber attack happens, RAM can contain evidence of the attack *while it's still happening* or right after. Unlike data stored on a hard drive, RAM data is **volatile**, meaning it disappears when the system is turned off. This makes it extremely valuable for **live investigations**.

---

2. What is Volatile Data?

Volatile data is any data that gets lost when a system is powered off. This includes:

- **Running processes** (e.g., open programs)
- **Active network connections**
- **Clipboard content**
- **Command-line history**
- **Passwords in memory**
- **Encryption keys**
- **Unwritten data (unsaved documents)**
- **Malware running in memory**

Because it can vanish instantly, memory must be captured *before* turning off the system. That's why **memory forensics is time-sensitive**.

---

3. Understanding Processes, Threads, and DLLs

To understand memory, you need to understand some key terms:

✅ Process

A process is a running instance of a program. For example, if you open Microsoft Word, it runs as a process. Each process is assigned a **PID (Process ID)**.

✅ Thread

A thread is a smaller unit of a process. A single process can run multiple threads to perform different tasks at the same time.

✅ DLL (Dynamic Link Library)

These are shared files used by processes to perform certain tasks, like printing or networking. DLLs can be exploited by malware to hide code inside legitimate programs.

---

4. Why Memory Forensics is Important

Here's why memory forensics is so useful:

- Detects malware not found on disk
- Helps find rootkits and stealth processes
- Recovers sensitive data like passwords or documents
- Shows real-time activity during an attack
- Provides proof of user actions

Even if attackers delete logs or uninstall tools, memory may still contain traces of what they did.

---

5. Tools for Memory Acquisition

Before analyzing memory, you must **capture** it using safe tools. Below are some common tools for memory capture:

🛠 Magnet RAM Capture

- Lightweight and easy to use
- Great for incident response
- Saves memory as `.raw` or `.bin`

🛠 Belkasoft RAM Capturer

- Free tool
- Captures full memory dump from Windows systems

🛠 FTK Imager

- Professional forensic tool
- Can capture memory and create disk images

🛠 DumpIt

- One-click memory dump tool
- Simple for quick use in the field

🛠 WinPmem

- Part of the Rekall project
- Works with command-line interface
- Great for Linux and Windows memory capture

🔒 **Note**: Always hash the memory dump (MD5 or SHA256) to prove integrity.

---

6. Tools for Memory Forensics Analysis

Once the memory is captured, you need tools to examine it. Here are the most commonly used:

🔍 Volatility 2

- Most popular open-source memory analysis tool
- Works with Windows, Linux, and macOS memory
- Plugin-based (each plugin extracts a specific type of data)

🔍 Rekall

- Originally a fork of Volatility
- Supports more modern OS versions
- Slightly more automated than Volatility

🔍 Redline

- Developed by FireEye
- Graphical interface for timeline and malware analysis
- Useful for non-technical investigators

🔍 LiME (Linux Memory Extractor)

- For acquiring memory on Linux systems

- Works well for cloud or embedded devices

---

7. Volatility 2: Overview and Features

🧰 What is Volatility 2?

Volatility 2 is a powerful memory forensics framework written in Python. It works by reading raw memory dumps and using **plugins** to extract meaningful data like process lists, network info, and malware indicators.

🔍 Key Features of Volatility 2

1. Cross-Platform Support

🖥️ **What it means:** Volatility works with memory from different types of systems—not just Windows!

💡 **In simple words:** You can use it to examine memory from **Windows**, **Linux**, and even **Mac** computers. It's useful no matter what kind of system you're investigating.

---

2. Plugin-Based Architecture

🔌 **What it means:** Volatility uses **plugins**, which are like mini-tools built into the main tool.

💡 **In simple words:** Each plugin does a specific job—like showing running processes, listing open network connections, or checking for malware. There are **over 100 plugins**, and you can even make your own if you know Python!

---

3. Supports Many Memory File Types

📁 **What it means:** It can open different kinds of memory dumps.

💡 **In simple words:** Volatility can read memory files from:

- Real computers (RAW memory dumps)
- Virtual machines (like VirtualBox or VMware)
- Crash dumps
- Hibernation files (when a computer goes to sleep)

No matter how the memory was captured, Volatility can usually handle it.

---

4. No Installation Needed

🚫 **What it means:** You don't need to install anything on your computer.

💡 **In simple words:** Just download the tool, run it using Python, and you're ready to go. This is great for using it on USB drives or in a forensic lab without changing the system.

---

5. Rootkit Detection

🦠 **What it means:** Volatility can find hidden or suspicious software that tries to avoid being seen.

💡 **In simple words:** Some malware hides itself from the Task Manager. Volatility can still **spot these sneaky programs** by directly looking into memory.

---

6. Malware Investigation

🔍 **What it means:** It can show signs of malicious software running in memory.

💡 **In simple words:** You can use Volatility to:

- See malware that only runs in RAM (and leaves no file on disk)
- Find weird code injected into real programs
- Analyze suspicious files without needing antivirus

It's perfect for **malware hunting**.

---

7. YARA Integration

🧬 **What it means:** Volatility works with **YARA rules**, which are like search patterns for malware.

💡 **In simple words:** If you have a rule that describes a certain virus, you can tell Volatility to **search memory** for anything that matches it. It's like using a **sniffer dog** trained to find certain smells—but for malware.

---

8. Registry Analysis

🗂️ **What it means:** Volatility can extract Windows registry data from memory.

💡 **In simple words:** The Windows registry stores settings for users and programs. Volatility can **pull out registry keys** to show things like:

- What programs run at startup
- Which USB devices were plugged in
- What a user recently accessed

Even if the hard drive is wiped, this data might still be in memory!

---

9. Timeline Creation

🕒 **What it means:** You can build a timeline of what happened on the system.

💡 **In simple words:** Volatility can show you:

- When files were created or opened
- When a process started or ended

This helps you understand **what happened and in what order**, which is super helpful during an investigation.

---

10. Open-Source and Free

💸 **What it means:** Anyone can use or improve Volatility without paying.

💡 **In simple words:** It's totally free to download, use, and even modify. Because it's open-source, the community can add features and fix bugs. That means it keeps getting better!

---

8. Common Volatility 2 Commands (with Explanations)

Let's say your memory image is named `memory.raw` and the OS profile is `Win7SP1x64`. Here's how to analyze it:

🧩 1. Identify Profile

```
python vol.py -f memory.raw imageinfo
```

📌 **Use it to:** Suggest correct OS profile for analysis.

📋 2. View Running Processes

```
python vol.py -f memory.raw --profile=Win7SP1x64 pslist
```

📌 **Use it to:** List all active programs when memory was captured.

🕵️‍♀️ 3. Scan for Hidden Processes

```
python vol.py -f memory.raw --profile=Win7SP1x64 psscan
```

📌 **Use it to:** Find processes hidden by malware (not in `pslist`).

🌐 4. List Network Connections

```
python vol.py -f memory.raw --profile=Win7SP1x64 netscan
```

📌 **Use it to:** Show open network connections and IP addresses.

⌨️ 5. Show Command History

```
python vol.py -f memory.raw --profile=Win7SP1x64 consoles
```

📌 **Use it to:** See what was typed in cmd or PowerShell.

💉 6. Detect Injected Code

```
python vol.py -f memory.raw --profile=Win7SP1x64 malfind
```

📌 **Use it to:** Find code injected by malware into legitimate processes.

📁 7. View DLLs of a Process

```
python vol.py -f memory.raw --profile=Win7SP1x64 dlllist -p <PID>
```

📌 **Use it to:** List DLL files loaded by a specific process.

💾 8. Dump Process Executable

```
python vol.py -f memory.raw --profile=Win7SP1x64 procdump -p <PID> --dump-dir=output/
```

📌 **Use it to:** Save a copy of a process to analyze further.

🔍 9. YARA Rule Scan

```
python vol.py -f memory.raw --profile=Win7SP1x64 yarascan -Y "MZ"
```

📌 **Use it to:** Look for suspicious or known malware patterns in memory.

---

9. Tips for Students Doing Memory Forensics

- 🔄 **Cross-check your results** from different plugins.
- 📂 **Save your commands and outputs** for documentation.
- 📌 Focus on **strange process names** or **network activity**.
- 🕒 Always note the **time and date** of analysis.
- 🧪 **Practice** with clean and infected memory images.
- 🔐 Never modify the original memory dump—**work on a copy**.

10. Final Thoughts

Memory forensics is one of the most powerful skills in digital forensics and cybersecurity. It gives investigators access to **live system data**, helping to detect attacks as they happen, or even **uncover threats that leave no trace on disk**.

With tools like **Volatility 2**, students can:

- Investigate live malware
- Extract user activity

- Detect hidden processes
- Analyze network behavior

As you practice, remember to approach memory analysis like a detective—always looking for connections, patterns, and anomalies.