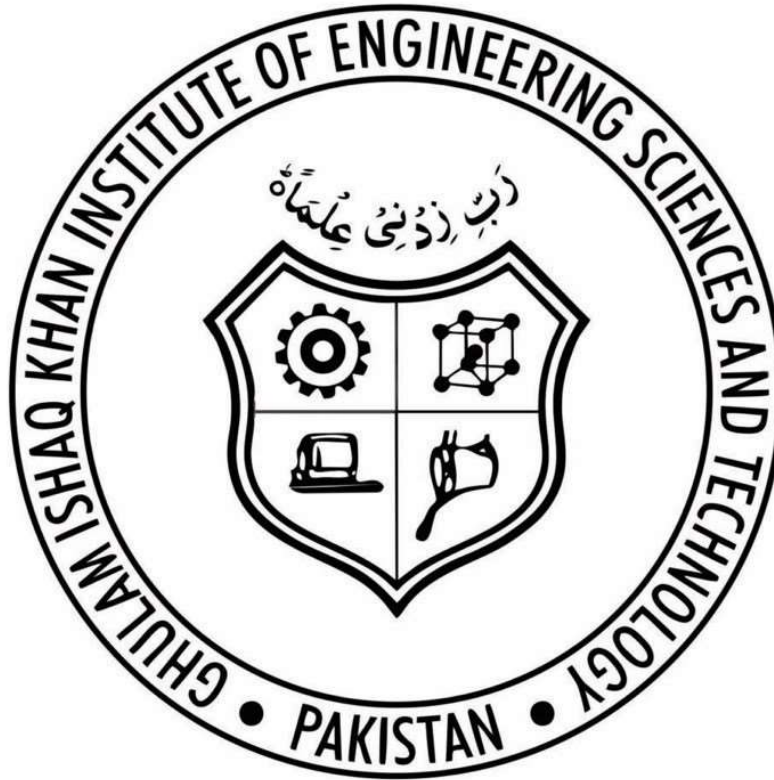


**Ghulam Ishaq Khan Institute of Engineering Sciences and  
Technology  
Digital Forensics-Spring 2025 CY341**



**By**

**Ayesha Kashif – 2022132**

**Ayela Israr Haqani-2022130**

**Noor ul Ain – 2022485**

**Submitted to:**

**Dr. Shahab Haider Assistant Professor FCS**

**Title: Live Memory Forensic Tool**

## 1. Introduction

This project presents a memory forensics framework for both **Linux** and **Windows** systems using open-source tools to detect malware and analyze volatile data. The aim was to design a forensics pipeline that performs memory acquisition, scanning, and malware detection with a focus on cross-platform capabilities.

Two independent workflows were developed:

- A Linux model using **LiME**, **Volatility**, and **YARA**.
- A Windows model using **WinPmem**, **Volatility**, and **YARA**.

Our approach enables analysts to extract and analyse key digital items such as running processes, network connections, and malicious indicators embedded in memory.

## 2. Problem Statement

Memory-resident malware poses a severe threat to digital infrastructures due to its stealthy nature and fileless execution. Detecting such threats requires real-time memory acquisition and deep forensics analysis.

This project addresses the following challenges:

- I. How to reliably acquire memory dumps in both Linux and Windows environments.
- II. How to analyse volatile memory for malicious processes or code injections.
- III. How to apply YARA rules effectively to detect signatures of malware in memory dumps.

## 3. System Design & Architecture

The system architecture is divided into two components:

### **Linux Workflow:**

- **Acquisition:** LiME kernel module
- **Analysis:** Volatility (pslist, netscan, malfind)
- **Detection:** YARA scanning on memory images

### **Windows Workflow:**

- **Acquisition:** WinPmem
- **Analysis:** Volatility using appropriate Windows profile
- **Detection:** YARA rules specific to Windows malware signatures

Scripts and automation are provided in Python to streamline these steps. All YARA rules are stored under `yara_rules/`.

## 4. Objectives

### 4.1 Cross-Platform Memory Acquisition

- Implement and use LiME on Linux to acquire [.lime](#) memory dumps.
- Use WinPmem for [.raw](#) memory dumps on Windows machines.

### 4.2 Volatility-Based Analysis

- Detect processes, network connections, and injected memory pages.
- Identify signs of malware such as suspicious processes or DLLs.

### 4.3 Malware Detection with YARA

- Develop and apply custom YARA rules.
- Detect known malware indicators using pattern matching.

### 4.4 Report Generation and Automation

- Generate HTML-based forensic reports.
- Provide modular code for scalability and automation.

## 5. Methodology

### Phase 1: Memory Acquisition

#### *Linux (LiME)*

- Compiled and loaded LiME module using:

```
bash
CopyEdit
insmod lime.ko "path=/root/dump.lime format=lime"
```

- Captured live memory in `.lime` format for analysis.

#### *Windows (WinPmem)*

- Ran winpmem.exe with the following:

```
cmd
CopyEdit
winpmem.exe -o memory.raw
```

- Successfully acquired a physical memory dump.

## Phase 2: Volatility Analysis

- Used the following Volatility plugins:
  - pslist: View active processes
  - netscan: Discover network connections
  - malfind: Detect injected or suspicious code

## Phase 3: YARA Detection

- Applied YARA rules to both .raw and .lime dumps.

### keylogger\_rules.yar

```
rule keylogger_sample {  
  
  strings:  
  
    $c = "keylogger_string"  
  
  condition:  
  
    $c}
```

Detected suspicious patterns in both environments.

## 6. Key Findings

- i. Volatility analysis showed processes in memory with anomalies.
- ii. YARA rules matched several indicative strings related to malware.
- iii. All results were compiled into an HTML-based forensic report.

## 7. Conclusion

This project successfully demonstrated the implementation of a cross-platform memory forensics solution. Using open-source tools, we were able to:

- Acquire volatile memory from both Linux and Windows.
- Extract and analyze system artifacts with Volatility.
- Detect malicious indicators using YARA rules.

The flexibility of the framework allows it to be easily extended for real-world scenarios involving advanced persistent threats or rootkit detection.

