Windows Forensics Project: Automated Memory Analyzer

Project by: Reut Abergel

Table of Contents

1. Introduction

This document serves as a technical guide for the Automated Memory Analyzer. The primary function of this script is to perform "dead box" forensics on raw memory dumps without requiring manual interaction or complex syntax memorization.The script combines two major forensic approaches into a single workflow:

- **File Carving:** Utilizes tools like **Binwalk**, **Foremost**, and **Bulk Extractor** to recover deleted files and hidden artifacts from the raw data.
- **Memory Analysis:** Integrates **Volatility** to parse RAM dumps, enabling the extraction of running processes, network connections, and registry keys.
- **Data Collection** The tool aggregates findings into a timestamped directory (forensics_output_YYYY...), generating a comprehensive report that includes:
- **Recovered Files:** Images, documents, and executables.
- **Network Artifacts:** .pcap files containing reconstructed network traffic.
- **System State:** Process lists (pslist) and connection logs (connscan).

2. Target Audience

This guide and tool are designed for:

- **Forensic Analysts:** Professionals seeking to automate the initial "triage" phase of a memory investigation.
- **Incident Responders:** Users needing a rapid method to extract evidence from a suspect machine's RAM.
- **Cybersecurity Students:** Individuals learning the mechanics of memory forensics and tool chaining in Linux.

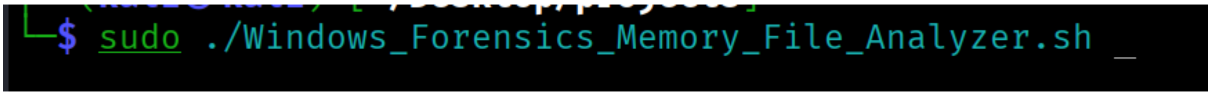
3. Requirements

To successfully execute this script, the following requirements must be met:

1. **Operating System:** Kali Linux (Native or VM).
2. **Privileges:** Root privileges are mandatory to install system tools and modify configuration files (/etc/scalpel/scalpel.conf).
3. **Connectivity:** Active internet connection for downloading Volatility standalone binaries and dependencies.
4. **Input Data:** A valid raw memory dump file (memdump.raw or image.bin).

4. Operational Guide

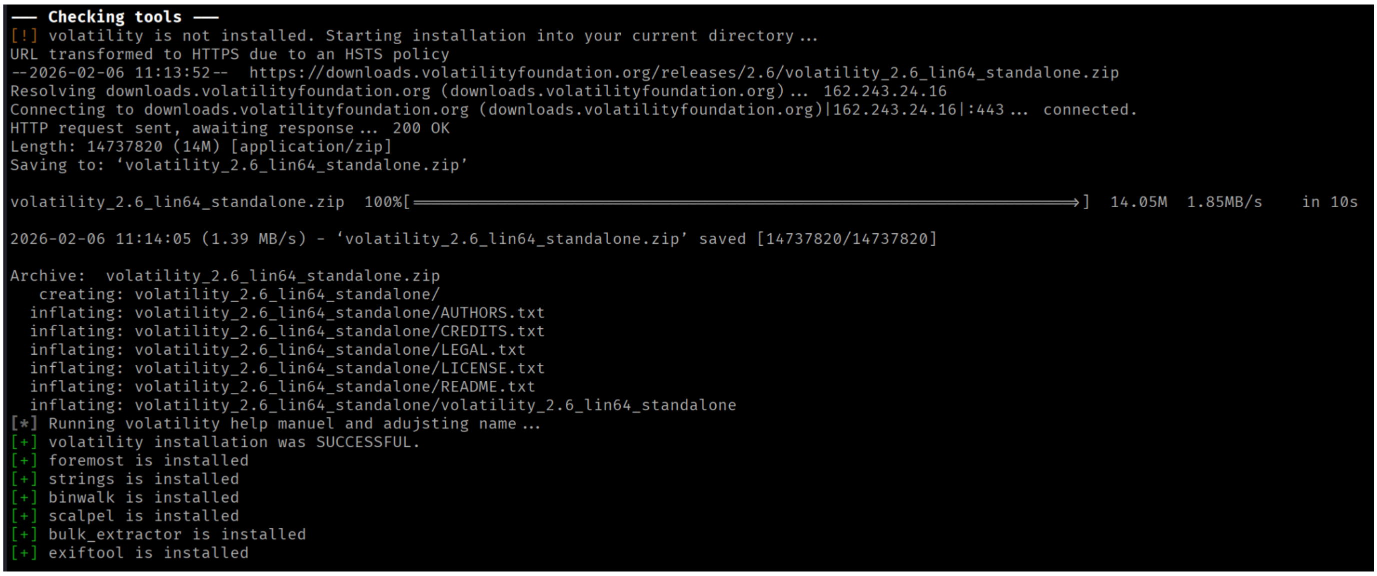
Step 1 Execution & Initialization: The script must be initiated with elevated privileges. Upon start, it captures the system time (to calculate total analysis duration later) and creates a unique workspace. Command:



```
└─$ sudo ./Windows_Forensics_Memory_File_Analyzer.sh _
```

Step 2 Automated Dependency Management: The script performs a "Health Check" on the environment.

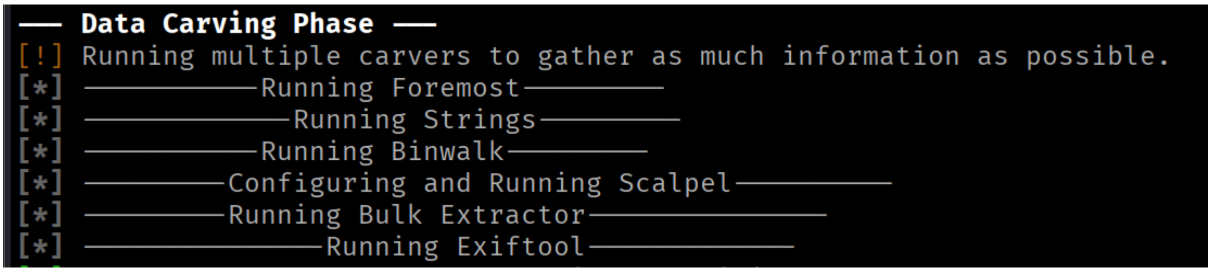
- **System Tools:** It verifies the installation of foremost, strings, binwalk, scalpel, and exiftool. Missing tools are installed via apt-get.
- **Volatility:** It checks for a local vol executable. If missing, it automatically fetches the standalone 2.6 version from the Volatility Foundation, unzips it, and configures it for immediate use.



```
── Checking tools ──
[!] volatility is not installed. Starting installation into your current directory ...
URL transformed to HTTPS due to an HSTS policy
--2026-02-06 11:13:52--  https://downloads.volatilityfoundation.org/releases/2.6/volatility_2.6_lin64_standalone.zip
Resolving downloads.volatilityfoundation.org (downloads.volatilityfoundation.org) ... 162.243.24.16
Connecting to downloads.volatilityfoundation.org (downloads.volatilityfoundation.org)|162.243.24.16|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 14737820 (14M) [application/zip]
Saving to: ‘volatility_2.6_lin64_standalone.zip’

volatility_2.6_lin64_standalone.zip  100%[=====================================================>]  14.05M  1.85MB/s    in 10s

2026-02-06 11:14:05 (1.39 MB/s) - ‘volatility_2.6_lin64_standalone.zip’ saved [14737820/14737820]

Archive:  volatility_2.6_lin64_standalone.zip
   creating: volatility_2.6_lin64_standalone/
  inflating: volatility_2.6_lin64_standalone/AUTHORS.txt
  inflating: volatility_2.6_lin64_standalone/CREDITS.txt
  inflating: volatility_2.6_lin64_standalone/LEGAL.txt
  inflating: volatility_2.6_lin64_standalone/LICENSE.txt
  inflating: volatility_2.6_lin64_standalone/README.txt
  inflating: volatility_2.6_lin64_standalone/volatility_2.6_lin64_standalone
[*] Running volatility help manuel and adujsting name ...
[+] volatility installation was SUCCESSFUL.
[+] foremost is installed
[+] strings is installed
[+] binwalk is installed
[+] scalpel is installed
[+] bulk_extractor is installed
[+] exiftool is installed
```

Step 3 File Carving & Extraction: Once the environment is ready, the script executes multiple carvers in parallel against the memory file.

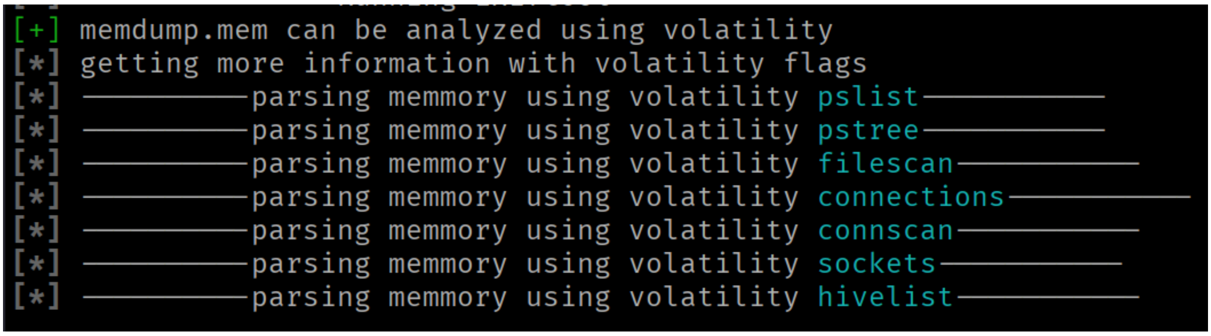
- **Scalpel Configuration:** It dynamically modifies the scalpel.conf file using sed, uncommenting specific file types (jpg, pdf, doc) to target relevant evidence.
- **Binwalk & Foremost:** These tools recursively extract embedded files and signatures.



```
—— Data Carving Phase ——
[!] Running multiple carvers to gather as much information as possible.
[*] ————————————Running Foremost——————————
[*] ——————————————Running Strings——————————
[*] ————————————Running Binwalk——————————
[*] ——————————Configuring and Running Scalpel——————————
[*] ——————————Running Bulk Extractor——————————————
[*] —————————————————Running Exiftool———————————————
```

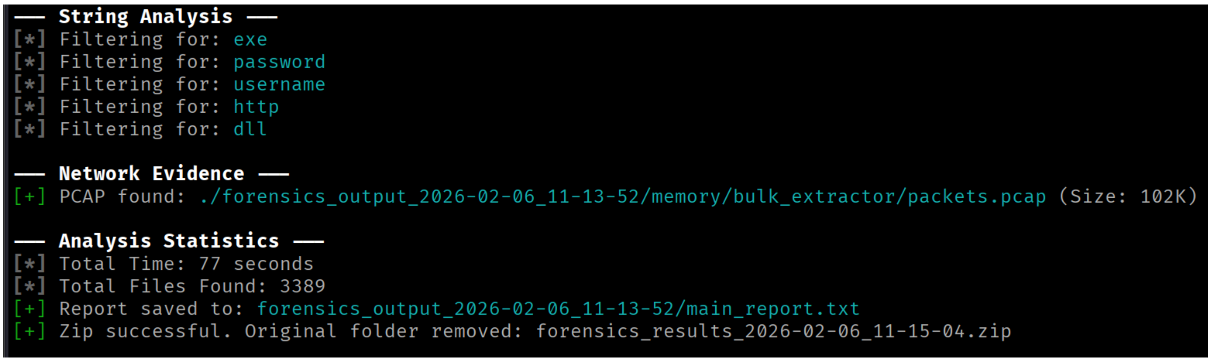
Step 4 Volatility Memory Analysis: The script runs imageinfo to identify the operating system profile. It then automatically extracts this profile name and loops through critical plugins:

- pslist / pstree (Running Processes)
- connscan / sockets (Network Activity)
- hivelist (Registry Hives)



```
[+] memdump.mem can be analyzed using volatility
[*] getting more information with volatility flags
[*] ————————————parsing memmory using volatility pslist —————————————
[*] ————————————parsing memmory using volatility pstree —————————————
[*] ————————————parsing memmory using volatility filescan —————————————
[*] ————————————parsing memmory using volatility connections ————————————
[*] ————————————parsing memmory using volatility connscan —————————————
[*] ————————————parsing memmory using volatility sockets —————————————
[*] ————————————parsing memmory using volatility hivelist —————————————
```

Step 5 Network Evidence & Reporting: The tool automatically scans for network packet captures (.pcap files) recovered during the extraction phase and reports their file size to the analyst, ensuring network artifacts are not overlooked. Simultaneously, it performs a targeted string search for high-value keywords (e.g., "password", "http", "dll"). Finally, it calculates execution statistics and compresses the entire evidence folder into a ZIP archive for secure transport.



```
—— String Analysis ——
[*] Filtering for: exe
[*] Filtering for: password
[*] Filtering for: username
[*] Filtering for: http
[*] Filtering for: dll

—— Network Evidence ——
[+] PCAP found: ./forensics_output_2026-02-06_11-13-52/memory/bulk_extractor/packets.pcap (Size: 102K)

—— Analysis Statistics ——
[*] Total Time: 77 seconds
[*] Total Files Found: 3389
[+] Report saved to: forensics_output_2026-02-06_11-13-52/main_report.txt
[+] Zip successful. Original folder removed: forensics_results_2026-02-06_11-15-04.zip
```

5. Purpose & Automation

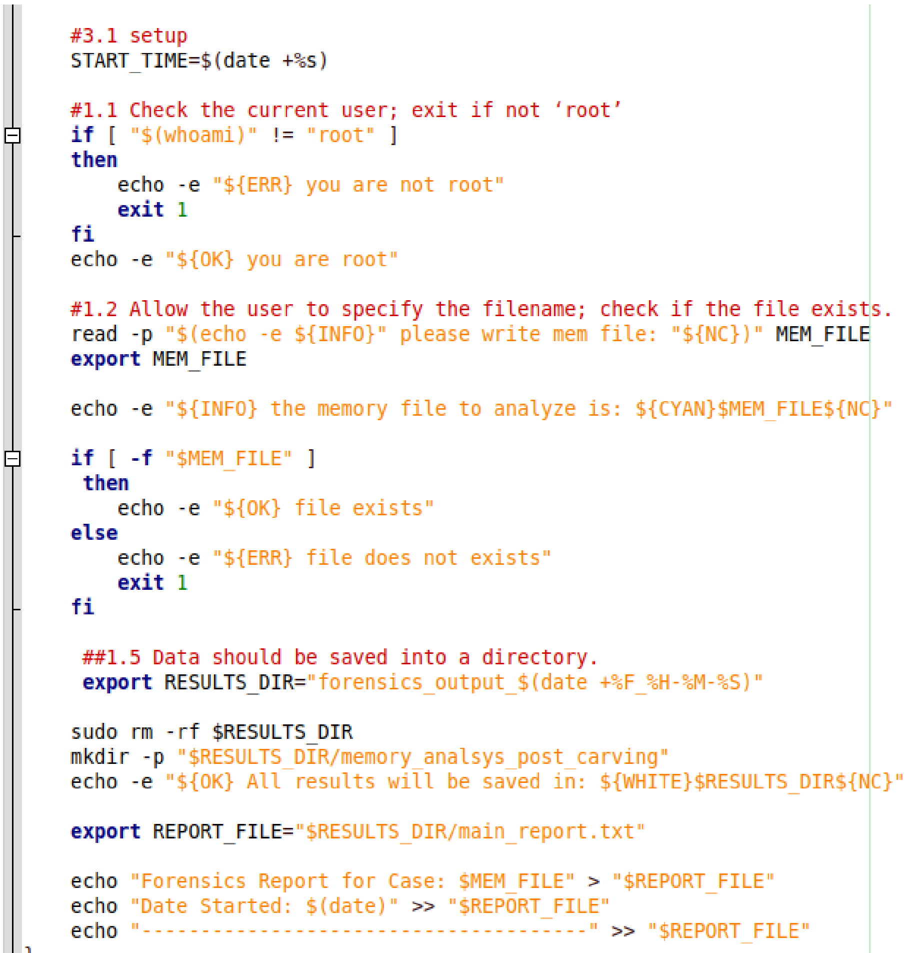
The goal is to perform a deep forensic investigation while eliminating the tedious manual setup often required in Linux forensics.

1. **Dynamic Configuration:** Instead of asking the user to manually edit configuration files (like /etc/scalpel/scalpel.conf), the script programmatically configures them using stream editors (sed) to enable specific file headers.
2. **Intelligent Profiling:** The script automates the hardest part of Volatility usage—identifying the correct profile. It parses the imageinfo output and feeds the result into subsequent commands, creating a seamless loop.
3. **Evidence Integrity:** By organizing output into structured, timestamped directories and zipping the final result, the tool ensures that evidence remains isolated and unmodified.

6. Technical Analysis

The script relies on six core functions to manage the workflow:

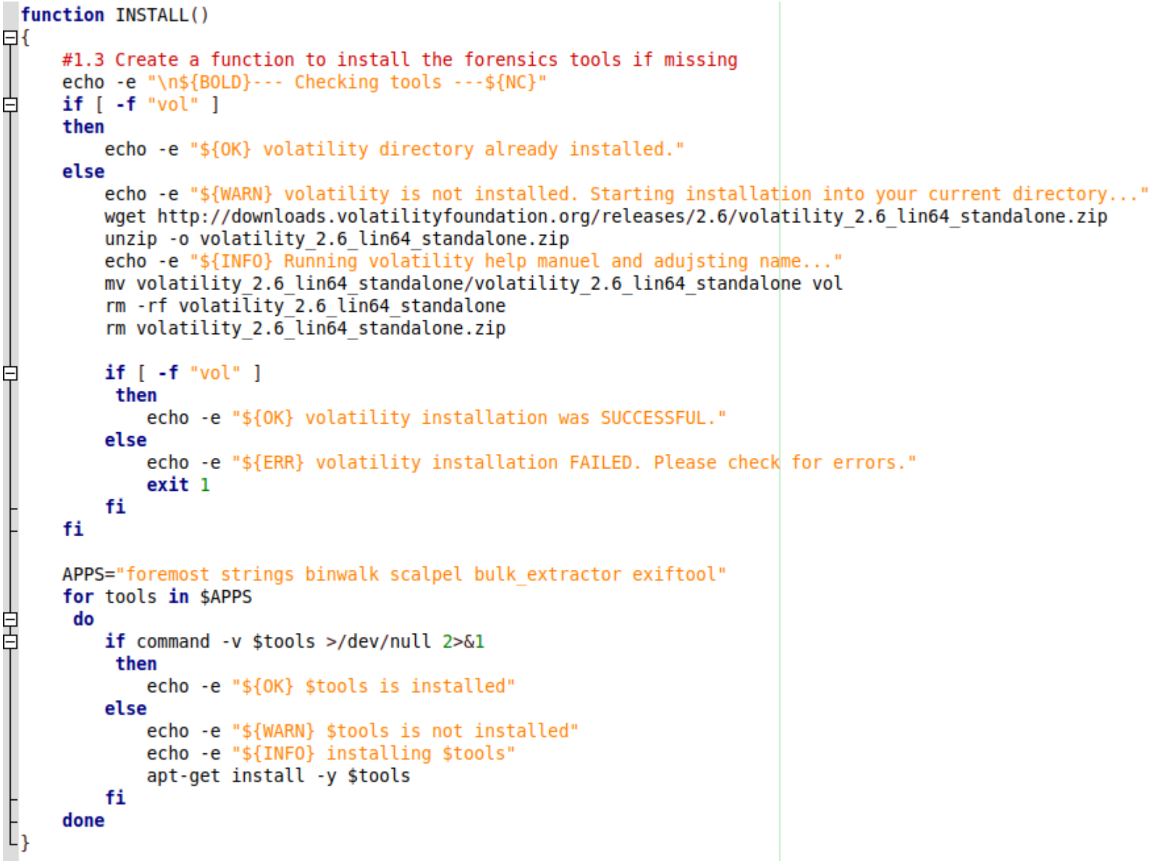
1. Function START (Environment Setup): Initializes the workspace and enforces privilege checks. It captures the START_TIME using date +%s. It then validates that the user is root and that the input file exists.



```
#3.1 setup
START_TIME=$(date +%s)

#1.1 Check the current user; exit if not ‘root’
if [ "$(whoami)" != "root" ]
then
    echo -e "${ERR} you are not root"
    exit 1
fi
echo -e "${OK} you are root"

#1.2 Allow the user to specify the filename; check if the file exists.
read -p "$(echo -e "${INFO}" please write mem file: "${NC})" MEM_FILE
export MEM_FILE

echo -e "${INFO} the memory file to analyze is: ${CYAN}$MEM_FILE${NC}"

if [ -f "$MEM_FILE" ]
 then
    echo -e "${OK} file exists"
else
    echo -e "${ERR} file does not exists"
    exit 1
fi

 ##1.5 Data should be saved into a directory.
 export RESULTS_DIR="forensics_output_$(date +%F_%H-%M-%S)"

sudo rm -rf $RESULTS_DIR
mkdir -p "$RESULTS_DIR/memory_analsys_post_carving"
echo -e "${OK} All results will be saved in: ${WHITE}$RESULTS_DIR${NC}"

export REPORT_FILE="$RESULTS_DIR/main_report.txt"

echo "Forensics Report for Case: $MEM_FILE" > "$REPORT_FILE"
echo "Date Started: $(date)" >> "$REPORT_FILE"
echo "--------------------------------------" >> "$REPORT_FILE"
}
```

2. Function INSTALL (Tool Retrieval): Handles hybrid installation methods (Package Manager + Direct Download). **Logic:**

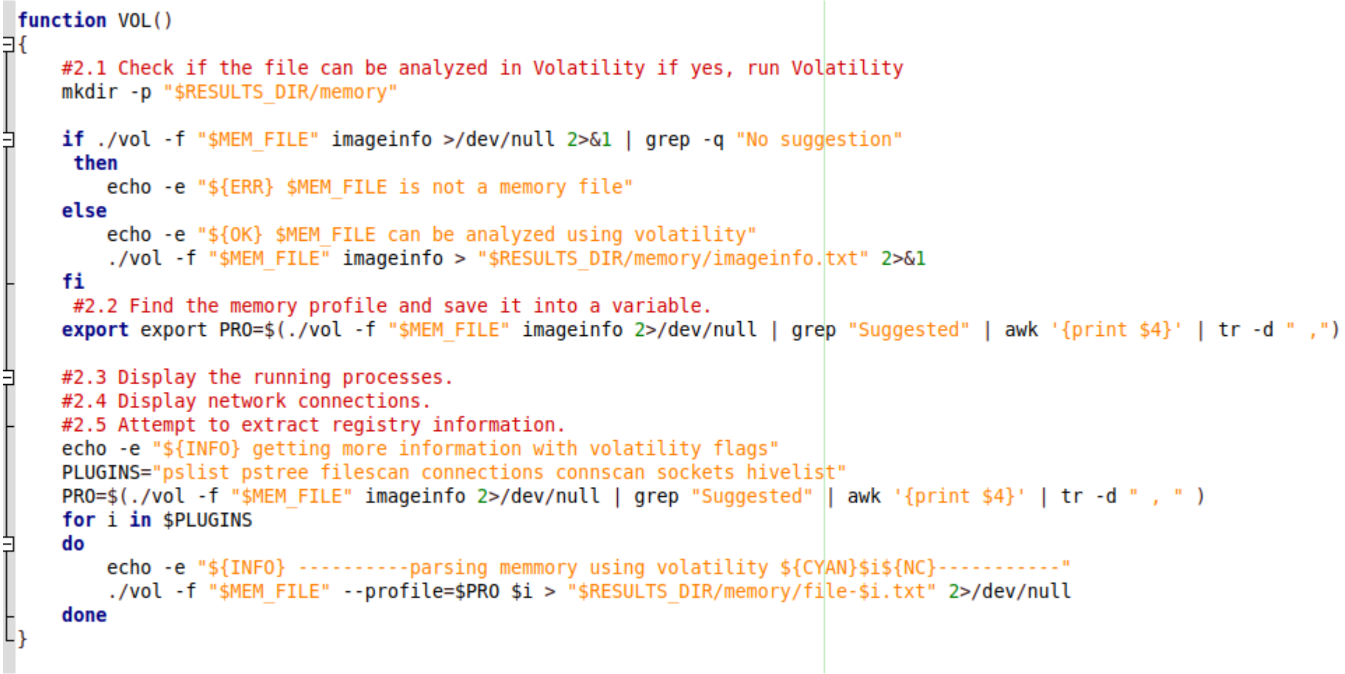
- **Loop:** Iterates through $APPS (foremost, binwalk, etc.) and installs them via apt.
- **Wget:** Downloads the Volatility standalone ZIP, unzips it, and renames the binary to ./vol for local execution, bypassing repository version issues.



```
function INSTALL()
{
    #1.3 Create a function to install the forensics tools if missing
    echo -e "\n${BOLD}--- Checking tools ---${NC}"
    if [ -f "vol" ]
    then
        echo -e "${OK} volatility directory already installed."
    else
        echo -e "${WARN} volatility is not installed. Starting installation into your current directory..."
        wget http://downloads.volatilityfoundation.org/releases/2.6/volatility_2.6_lin64_standalone.zip
        unzip -o volatility_2.6_lin64_standalone.zip
        echo -e "${INFO} Running volatility help manuel and adujsting name..."
        mv volatility_2.6_lin64_standalone/volatility_2.6_lin64_standalone vol
        rm -rf volatility_2.6_lin64_standalone
        rm volatility_2.6_lin64_standalone.zip

        if [ -f "vol" ]
         then
            echo -e "${OK} volatility installation was SUCCESSFUL."
        else
            echo -e "${ERR} volatility installation FAILED. Please check for errors."
            exit 1
        fi
    fi

    APPS="foremost strings binwalk scalpel bulk_extractor exiftool"
    for tools in $APPS
     do
        if command -v $tools >/dev/null 2>&1
         then
            echo -e "${OK} $tools is installed"
        else
            echo -e "${WARN} $tools is not installed"
            echo -e "${INFO} installing $tools"
            apt-get install -y $tools
        fi
    done
}
```

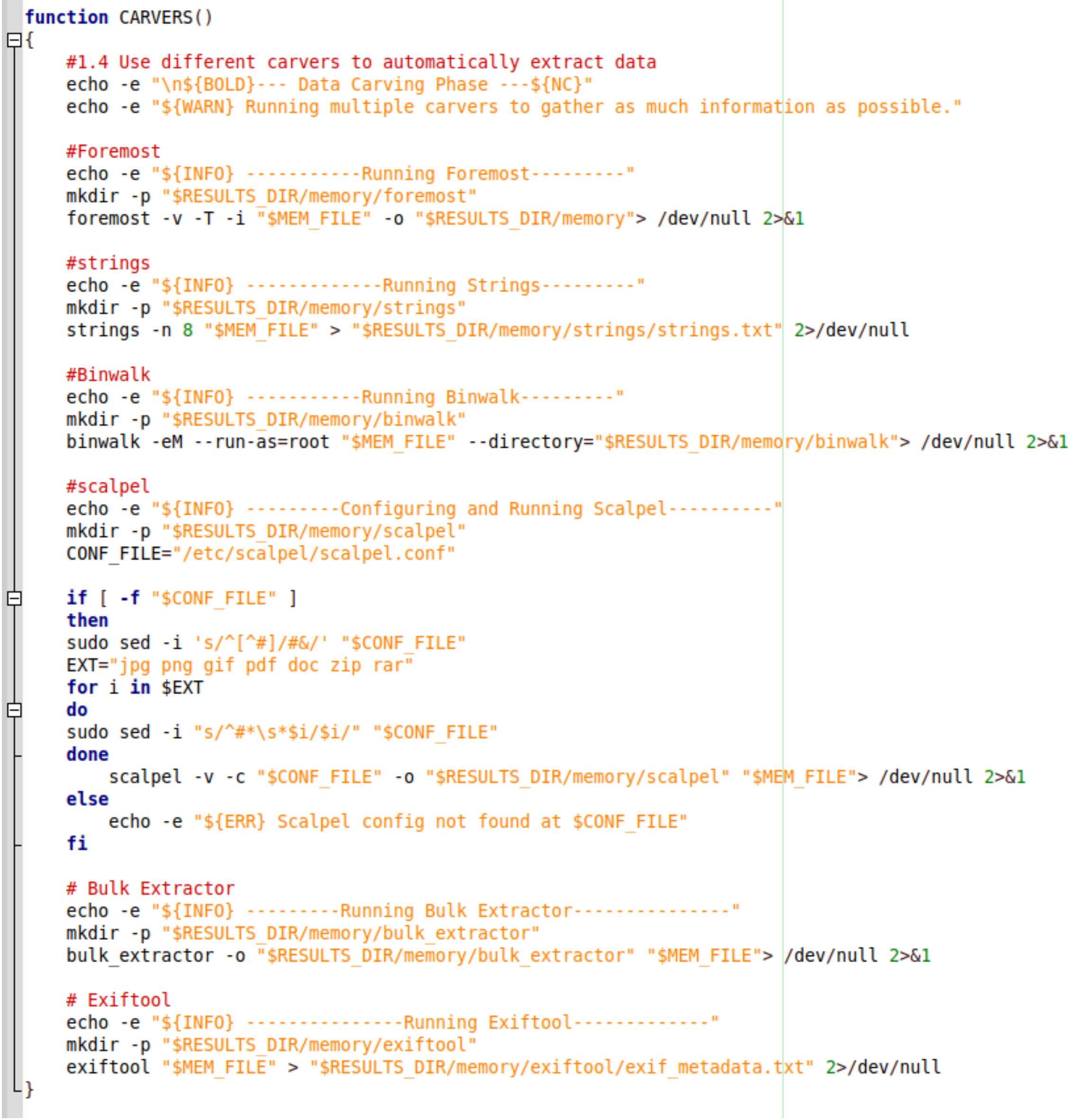
3. Function VOL (RAM Analysis): The core intelligence engine for memory analysis.

- **Profile Extraction:** Uses grep and awk to isolate the "Suggested Profile" from the imageinfo output.
- **Plugin Loop:** Iterates through a list of plugins (pslist, connscan) using the extracted $PRO variable.



```
function VOL()
{
    #2.1 Check if the file can be analyzed in Volatility if yes, run Volatility
    mkdir -p "$RESULTS_DIR/memory"

    if ./vol -f "$MEM_FILE" imageinfo >/dev/null 2>&1 | grep -q "No suggestion"
     then
        echo -e "${ERR} $MEM_FILE is not a memory file"
    else
        echo -e "${OK} $MEM_FILE can be analyzed using volatility"
        ./vol -f "$MEM_FILE" imageinfo > "$RESULTS_DIR/memory/imageinfo.txt" 2>&1
    fi
     #2.2 Find the memory profile and save it into a variable.
    export export PRO=$(./vol -f "$MEM_FILE" imageinfo 2>/dev/null | grep "Suggested" | awk '{print $4}' | tr -d " ,")

    #2.3 Display the running processes.
    #2.4 Display network connections.
    #2.5 Attempt to extract registry information.
    echo -e "${INFO} getting more information with volatility flags"
    PLUGINS="pslist pstree filescan connections connscan sockets hivelist"
    PRO=$(./vol -f "$MEM_FILE" imageinfo 2>/dev/null | grep "Suggested" | awk '{print $4}' | tr -d " , " )
    for i in $PLUGINS
    do
        echo -e "${INFO} ----------parsing memmory using volatility ${CYAN}$i${NC}-----------"
        ./vol -f "$MEM_FILE" --profile=$PRO $i > "$RESULTS_DIR/memory/file-$i.txt" 2>/dev/null
    done
}
```

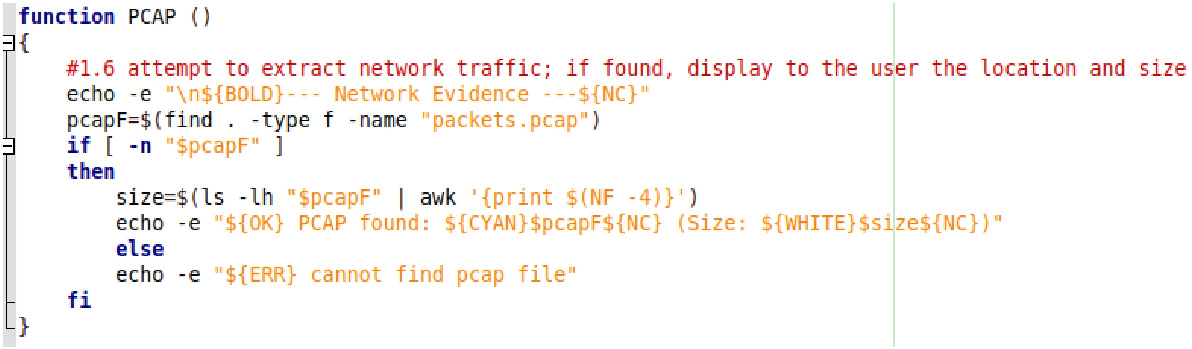
4. Function CARVERS (Artifact Extraction): Configures and executes file carving tools. Uses sed to manipulate the configuration file. It first comments out all lines



```
function CARVERS()
{
    #1.4 Use different carvers to automatically extract data
    echo -e "\n${BOLD}--- Data Carving Phase ---${NC}"
    echo -e "${WARN} Running multiple carvers to gather as much information as possible."

    #Foremost
    echo -e "${INFO} -----------Running Foremost---------"
    mkdir -p "$RESULTS_DIR/memory/foremost"
    foremost -v -T -i "$MEM_FILE" -o "$RESULTS_DIR/memory"> /dev/null 2>&1

    #strings
    echo -e "${INFO} -------------Running Strings---------"
    mkdir -p "$RESULTS_DIR/memory/strings"
    strings -n 8 "$MEM_FILE" > "$RESULTS_DIR/memory/strings/strings.txt" 2>/dev/null

    #Binwalk
    echo -e "${INFO} -----------Running Binwalk---------"
    mkdir -p "$RESULTS_DIR/memory/binwalk"
    binwalk -eM --run-as=root "$MEM_FILE" --directory="$RESULTS_DIR/memory/binwalk"> /dev/null 2>&1

    #scalpel
    echo -e "${INFO} ---------Configuring and Running Scalpel----------"
    mkdir -p "$RESULTS_DIR/memory/scalpel"
    CONF_FILE="/etc/scalpel/scalpel.conf"

    if [ -f "$CONF_FILE" ]
    then
    sudo sed -i 's/^[^#]/#&/' "$CONF_FILE"
    EXT="jpg png gif pdf doc zip rar"
    for i in $EXT
    do
    sudo sed -i "s/^#*\s*$i/$i/" "$CONF_FILE"
    done
        scalpel -v -c "$CONF_FILE" -o "$RESULTS_DIR/memory/scalpel" "$MEM_FILE"> /dev/null 2>&1
    else
        echo -e "${ERR} Scalpel config not found at $CONF_FILE"
    fi

    # Bulk Extractor
    echo -e "${INFO} ---------Running Bulk Extractor---------------"
    mkdir -p "$RESULTS_DIR/memory/bulk_extractor"
    bulk_extractor -o "$RESULTS_DIR/memory/bulk_extractor" "$MEM_FILE"> /dev/null 2>&1

    # Exiftool
    echo -e "${INFO} ---------------Running Exiftool-------------"
    mkdir -p "$RESULTS_DIR/memory/exiftool"
    exiftool "$MEM_FILE" > "$RESULTS_DIR/memory/exiftool/exif_metadata.txt" 2>/dev/null
}
```

5. Function PCAP & STR (Network & Keyword Analysis): targeted extraction of specific data types. **Logic:**

- **PCAP:** Uses find to locate any .pcap files recovered by Bulk Extractor and reports their size.
- **STR:** Loops through a list of keywords and uses strings | grep to find readable text artifacts.



```
function PCAP ()
{
    #1.6 attempt to extract network traffic; if found, display to the user the location and size
    echo -e "\n${BOLD}--- Network Evidence ---${NC}"
    pcapF=$(find . -type f -name "packets.pcap")
    if [ -n "$pcapF" ]
    then
        size=$(ls -lh "$pcapF" | awk '{print $(NF -4)}')
        echo -e "${OK} PCAP found: ${CYAN}$pcapF${NC} (Size: ${WHITE}$size${NC})"
        else
        echo -e "${ERR} cannot find pcap file"
    fi
}
```

7. Technical Skills Gained

Through the development of this project, I developed the following technical skills:

- **Advanced Text Manipulation:** Mastered the use of sed and awk to programmatically edit system configuration files and parse complex tool output.
- **Forensic Tool Chaining:** Learned to integrate distinct tools (Volatility, Binwalk, Foremost) into a unified pipeline.
- **System Resource Management:** Implemented logic to check for software dependencies and perform "hybrid" installations (apt + manual binary management).
- **Variable Scope Management:** Utilized export to ensure critical variables (like $MEM_FILE and $PRO) were accessible across different function blocks.

8. Summary

This project resulted in a fully automated forensic analyzer. By abstracting the complexity of tool installation, configuration, and syntax, the script allows an analyst to focus on the **evidence** rather than the **process**. It demonstrates a strong understanding of both Linux scripting and the principles of digital forensics.