Windows Forensics Analysis Script Report

Script Explanation

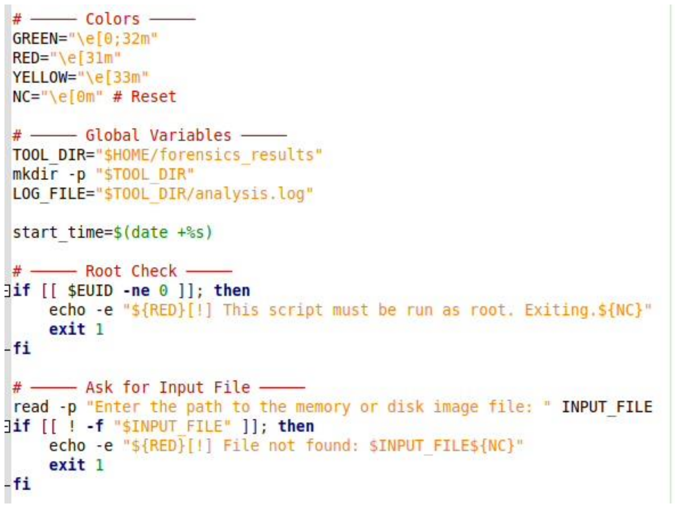
Windows Forensics Analysis Script Explanation

This bash script automates the forensic analysis of memory or disk image files on a Linux system.

It installs necessary tools, performs data carving, extracts readable data, conducts memory analysis with Volatility.

detects PCAP files, and generates a comprehensive forensic report.

Overview of Functions:



```
# ───── Colors ─────
GREEN="\e[0;32m"
RED="\e[31m"
YELLOW="\e[33m"
NC="\e[0m" # Reset

# ───── Global Variables ─────
TOOL_DIR="$HOME/forensics_results"
mkdir -p "$TOOL_DIR"
LOG_FILE="$TOOL_DIR/analysis.log"

start_time=$(date +%s)

# ───── Root Check ─────
if [[ $EUID -ne 0 ]]; then
    echo -e "${RED}[!] This script must be run as root. Exiting.${NC}"
    exit 1
fi

# ───── Ask for Input File ─────
read -p "Enter the path to the memory or disk image file: " INPUT_FILE
if [[ ! -f "$INPUT_FILE" ]]; then
    echo -e "${RED}[!] File not found: $INPUT_FILE${NC}"
    exit 1
fi
```

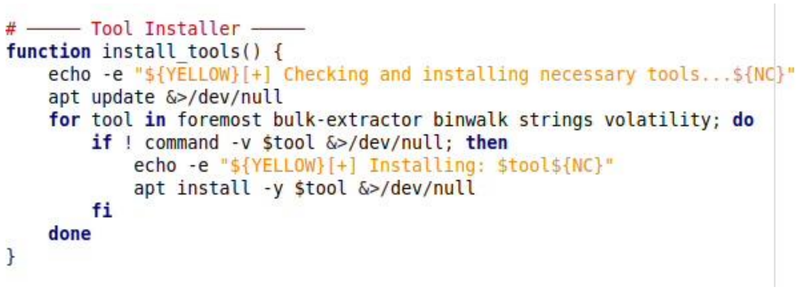
1. **Defines color variables** for pretty terminal output.
2. **Creates a directory** to store results and sets a log file.
3. **Starts a timer** to measure how long the script runs.
4. **Verifies root access**, since this is required for most forensic tasks.
5. **Prompts the user** for the path to a disk/memory image and **validates** the input.

----

1. install tools:

- Checks and installs the required tools: foremost, bulk-extractor, binwalk, strings, and volatility.

- Ensures the environment is ready before any analysis begins.



```
# ----- Tool Installer -----
function install_tools() {
    echo -e "${YELLOW}[+] Checking and installing necessary tools...${NC}"
    apt update &>/dev/null
    for tool in foremost bulk-extractor binwalk strings volatility; do
        if ! command -v $tool &>/dev/null; then
            echo -e "${YELLOW}[+] Installing: $tool${NC}"
            apt install -y $tool &>/dev/null
        fi
    done
}
```

1. Shows a message that tools are being checked.
2. Silently updates the apt package index.
3. Loops through a list of forensic tools.
4. If a tool is **not already installed**:
   - Shows a message in yellow.
   - Silently installs the tool.

This makes sure the script works on **any system** without requiring the user to manually install every tool.

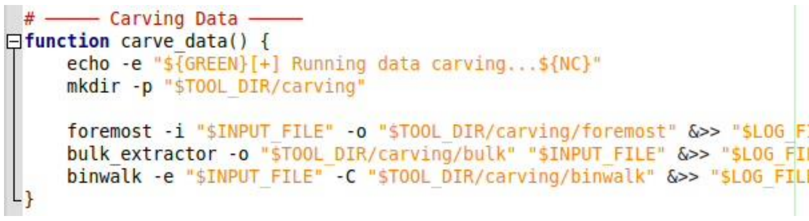
2. carve data:

- Uses three tools to extract artifacts from the image:

  - foremost: for file carving.

  - bulk_extractor: to extract emails, URLs, and other forensic artifacts.

  - binwalk: to extract embedded files and firmware contents.



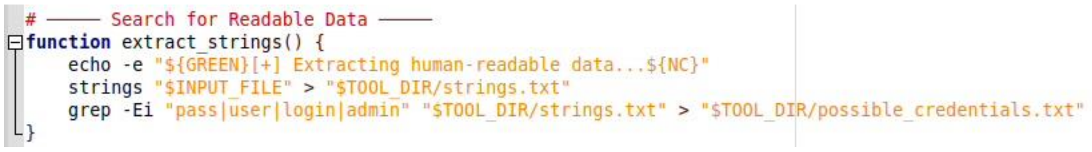
```
# ----- Carving Data -----
function carve_data() {
    echo -e "${GREEN}[+] Running data carving...${NC}"
    mkdir -p "$TOOL_DIR/carving"

    foremost -i "$INPUT_FILE" -o "$TOOL_DIR/carving/foremost" &>> "$LOG_FILE"
    bulk_extractor -o "$TOOL_DIR/carving/bulk" "$INPUT_FILE" &>> "$LOG_FILE"
    binwalk -e "$INPUT_FILE" -C "$TOOL_DIR/carving/binwalk" &>> "$LOG_FILE"
}
```

1. Prints a status message in green to notify the user.
2. Prepares a directory for carved data.
3. Uses **three tools** (foremost, bulk_extractor, binwalk) to analyze and extract files or patterns from the input disk/memory image.
4. Saves all carved data into separate subfolders.
5. **Appends all tool output into one central log file** (analysis.log).

3. extract strings:

- Extracts human-readable ASCII strings from the image using `strings`.

- Searches for potential credentials using grep with common keywords like "pass", "user", etc.



```
# ----- Search for Readable Data -----
function extract_strings() {
    echo -e "${GREEN}[+] Extracting human-readable data...${NC}"
    strings "$INPUT_FILE" > "$TOOL_DIR/strings.txt"
    grep -Ei "pass|user|login|admin" "$TOOL_DIR/strings.txt" > "$TOOL_DIR/possible_credentials.txt"
}
```

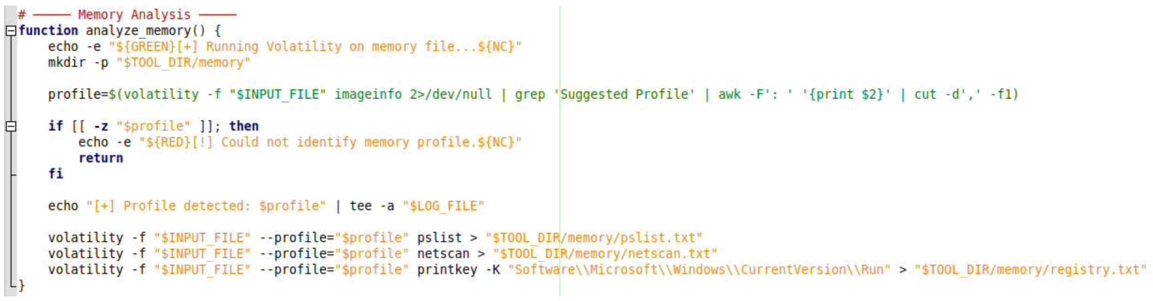
1. Extracts all readable text from the disk/memory image.
2. Saves that raw text into strings.txt.
3. Filters that text for any signs of credentials (like password, login, user, or admin).
4. Saves these suspicious lines into possible_credentials.txt.

4. analyze memory:

- If the file is a memory dump, this function uses Volatility to:

  - Identify the correct profile.

  - Extract running processes (`pslist`), network connections (`netscan`), and registry autostart keys.



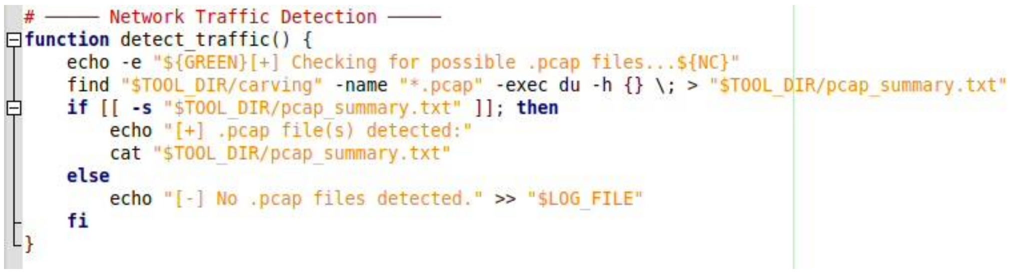
```
# ——— Memory Analysis ———
function analyze_memory() {
    echo -e "${GREEN}[+] Running Volatility on memory file...${NC}"
    mkdir -p "$TOOL_DIR/memory"

    profile=$(volatility -f "$INPUT_FILE" imageinfo 2>/dev/null | grep 'Suggested Profile' | awk -F': ' '{print $2}' | cut -d',' -f1)

    if [[ -z "$profile" ]]; then
        echo -e "${RED}[!] Could not identify memory profile.${NC}"
        return
    fi

    echo "[+] Profile detected: $profile" | tee -a "$LOG_FILE"

    volatility -f "$INPUT_FILE" --profile="$profile" pslist > "$TOOL_DIR/memory/pslist.txt"
    volatility -f "$INPUT_FILE" --profile="$profile" netscan > "$TOOL_DIR/memory/netscan.txt"
    volatility -f "$INPUT_FILE" --profile="$profile" printkey -K "Software\\Microsoft\\Windows\\CurrentVersion\\Run" > "$TOOL_DIR/memory/registry.txt"
}
```

1. Creates a folder for memory results.
2. Uses Volatility to detect the correct memory profile.
3. If a profile is found:
   - It logs the profile used.
   - It extracts:
     - Active processes list (pslist)
     - Open network connections (netscan)
     - Registry “Run” entries (printkey)

5. detect traffic:

- Searches for PCAP files generated from carving.

- Summarizes size and location for review.



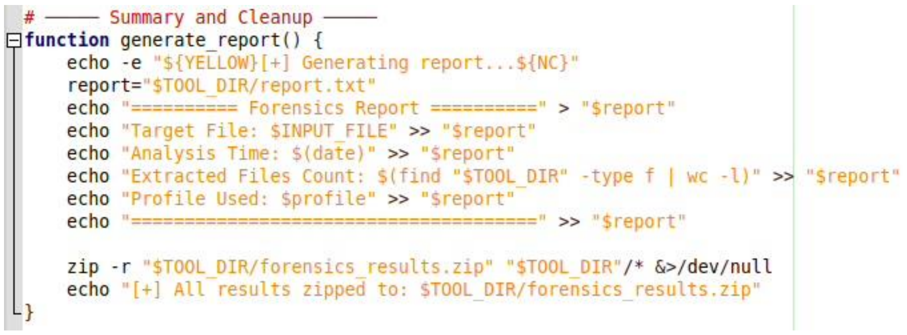
```
# ——— Network Traffic Detection ———
function detect_traffic() {
    echo -e "${GREEN}[+] Checking for possible .pcap files...${NC}"
    find "$TOOL_DIR/carving" -name "*.pcap" -exec du -h {} \; > "$TOOL_DIR/pcap_summary.txt"
    if [[ -s "$TOOL_DIR/pcap_summary.txt" ]]; then
        echo "[+] .pcap file(s) detected:"
        cat "$TOOL_DIR/pcap_summary.txt"
    else
        echo "[-] No .pcap files detected." >> "$LOG_FILE"
    fi
}
```

1. Searches for .pcap files inside the carving directory.
2. If it finds any, it:
   - Logs their size and location.
   - Displays the result to the user.
3. If no .pcap files are found, it logs a message indicating that.

6. generate report:

- Generates a report with the number of files extracted and the analysis profile used.

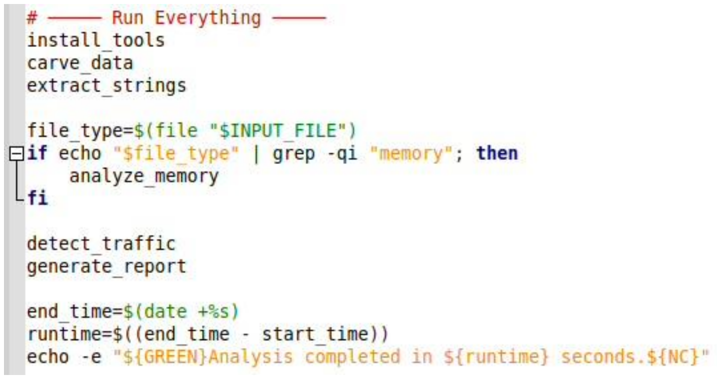
- Compresses all findings into a ZIP archive for submission or future analysis.



```
# ——— Summary and Cleanup ———
function generate_report() {
    echo -e "${YELLOW}[+] Generating report...${NC}"
    report="$TOOL_DIR/report.txt"
    echo "========== Forensics Report ==========" > "$report"
    echo "Target File: $INPUT_FILE" >> "$report"
    echo "Analysis Time: $(date)" >> "$report"
    echo "Extracted Files Count: $(find "$TOOL_DIR" -type f | wc -l)" >> "$report"
    echo "Profile Used: $profile" >> "$report"
    echo "=====================================" >> "$report"

    zip -r "$TOOL_DIR/forensics_results.zip" "$TOOL_DIR"/* &>/dev/null
    echo "[+] All results zipped to: $TOOL_DIR/forensics_results.zip"
}
```

1. Prepares a **summary report** with key details:
    - Input file analyzed
    - Date of analysis
    - Number of output files
    - Volatility profile used
2. Compresses all analysis output into one .zip file.
3. Displays a success message at the end.

Execution Flow:

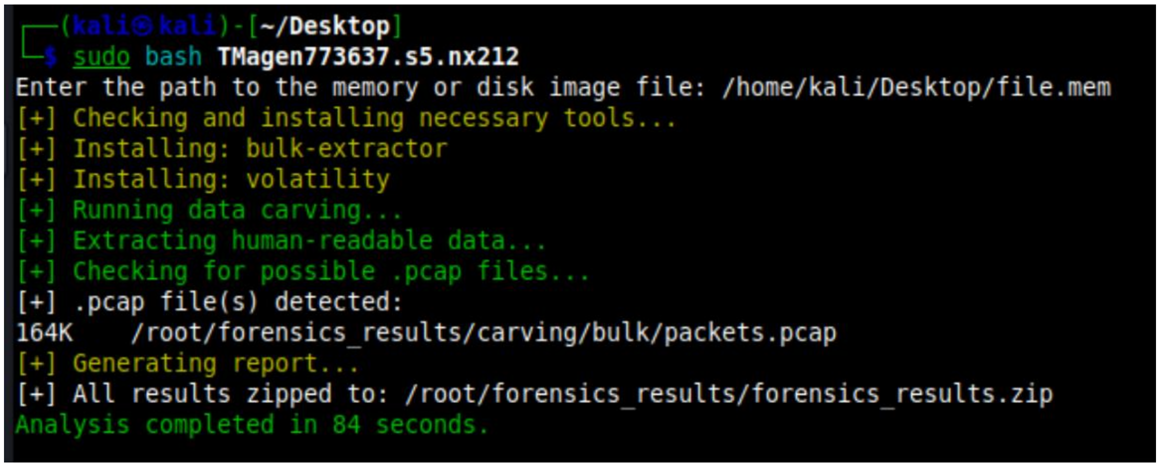


```
# ——— Run Everything ———
install_tools
carve_data
extract_strings

file_type=$(file "$INPUT_FILE")
if echo "$file_type" | grep -qi "memory"; then
    analyze_memory
fi

detect_traffic
generate_report

end_time=$(date +%s)
runtime=$((end_time - start_time))
echo -e "${GREEN}Analysis completed in ${runtime} seconds.${NC}"
```

1. Runs all major functions one after the other.
2. Detects the type of input file and decides whether to analyze memory.
3. Performs cleanup tasks (PCAP detection, report generation).
4. Calculates total runtime and prints a completion message.

Script Execution Screenshot

Below is a screenshot showing the successful execution of the script:



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo bash TMagen773637.s5.nx212
Enter the path to the memory or disk image file: /home/kali/Desktop/file.mem
[+] Checking and installing necessary tools...
[+] Installing: bulk-extractor
[+] Installing: volatility
[+] Running data carving...
[+] Extracting human-readable data...
[+] Checking for possible .pcap files...
[+] .pcap file(s) detected:
164K    /root/forensics_results/carving/bulk/packets.pcap
[+] Generating report...
[+] All results zipped to: /root/forensics_results/forensics_results.zip
Analysis completed in 84 seconds.
```