

Name: reut abergel

SID :s8

Unit: TMagen7736.38

Program code:NX212

# Windows Forensics project

## Table of contents

1. Introduction

1.2 Who is this guide for

1.3 How this guide helps

4. What you'll need

5. How to Run

6. Purpose of the Tool

What I learned from building the script

Analysis Complex functions and commands

Summary

## 1.INTRODUCTION

Hi! This is a friendly guide for my script. The script's role is to act as an automated Analyzer for Windows memory dumps. It takes a raw file provided by the user and extracts hidden evidence automatically.

it combines two major forensic approaches:

1. It uses tools like Binwalk and Foremost to extract files and network traffic without needing to "run" the system.
2. It utilizes Volatility to analyze RAM dumps, finding running processes and registry keys.
3. The script collects information and saves it to a structured report folder

### **Who is this guide for?**

1. Students in a cyber course.
2. Anyone who wants to analyze a memory file but doesn't want to memorize complex carving commands.
3. Users who want to automate the installation of forensic tools (Bulk Extractor, Foremost, Volatility).

### **How the guide helps:**

1. Explains the forensic logic behind the tools used.
2. Shows exactly how to run the script step-by-step.
3. Describes what each main function in the script does (START, INSTALL, CARVERS,PCAP,STR ,VOL ) and analyzes the commands within it.

## What you'll need

1. **Kali Linux:** VM or operating system.
2. **Root permissions:** The script checks this automatically  
you must be root to install the carvers and volatility.
3. **Internet connection:** Required for downloading missing tools
4. **A Memory File:** A file to analyze (inputted by the user).

## How to Run

1. **Execution:** You must run the script with root permissions. sudo  
`./wfproj.sh`
2. **Automatic Setup:** The script starts by checking your User ID. If you are not root, it will exit also it defines a unique results directory based on the current time: `forensics_output_YYYY-MM-DD_HH-MM-SS`.
3. **Tool Check & Installation:**
  - **Volatility:** It checks if a local file named vol exists. If not, it uses wget to download the specific standalone zip, unzips it, and renames it to vol for easy usage.
  - **System Tools:** It loops through a list (foremost, strings, binwalk...) and installs any missing ones using apt-get.
4. **The Analysis:** The script asks for your memory file, runs all carvers, configures Scalpel, and then uses Volatility to find the OS profile and extract process lists and connections.

## Purpose of the Tool

The goal is to perform a deep forensic investigation while automating the tedious setup phase:

1. **Dynamic Configuration:** Instead of asking the user to manually edit /etc/scalpel/scalpel.conf, the script uses sed to uncomment image and changes file types automatically.
2. **Keyword Hunting:** It automatically runs strings looking for high-value targets like "password", "http", and "dll".
3. **Statistics:** It calculates the total execution time and total files found at the very end to give the analyst a quick summary.

## What I learned from building the script

- **Advanced Text Manipulation:** How to use sed to find and replace text in system configuration files.
- **Time Calculation:** How to use \$(date +%s) to measure exactly how long an analysis takes.
  - **Variable Exporting:** Using export to ensure variables like \$MEM\_FILE are available to all functions.

## Analysis of functions and complex commands

The script is divided into 6 main functions:

START, INSTALL, VOL, CARVERS, PCAP and STR

## Function: START

**What it does:** Sets up the environment. **Process:**

1. Captures START\_TIME.
2. Checks whoami. If not "root", it exits.
3. Asks the user for the filename and verifies it exists using [ -f "\$MEM\_FILE" ].
4. Creates a timestamped directory forensics\_output\_... to store all results.

```
function START ()  
{  
    figlet "welcome to WF proj"  
    #3.1 setup  
    START_TIME=$(date +%s)  
  
    #1.1 Check the current user; exit if not 'root'  
    if [ "$(whoami)" != "root" ]  
    then  
        echo "[():]you are not root"  
        exit 1  
    fi  
    echo "[():]you are root"  
  
    #1.2 Allow the user to specify the filename; check if the file exists.  
    read -p "[():]please write mem file: " MEM_FILE  
    # export makes sure that other functions can see the Variable  
    export MEM_FILE  
  
    echo "[!]the memory file to analyze is: $MEM_FILE"  
    if [ -f "$MEM_FILE" ]  
    then  
        echo "[():]file exists"  
    else  
        echo "[!!]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_analysys_post_carving"  
    echo "All results will be saved in: $RESULTS_DIR"  
}
```

## Function: INSTALL

**What it does:** Handles two types of installations.

1. **Manual:** Checks for the vol . If missing, it runs wget to fetch the specific zip from the Volatility Foundation, unzips it, and renames the tool to ./vol.

**2. Package Manager:** Loops through \$APPS ("foremost strings binwalk...") and checks them with command -v. If missing, it runs apt-get install -y.

```

function INSTALL()
{
    #1.3 Create a function to install the forensics tools if missing
    echo "... Checking tools ..."
    if [ -f "vol" ]
    then
        echo "[::] volatility directory already installed."
    else
        echo "[!] 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 "[*] Running volatility help manuel and adjustring 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 "[::] volatility installation was SUCCESSFUL.."
        else
            echo "[::] 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 "[::] $tools is installed"
        else
            echo "[!] $tools is not installed"
            echo "[#] installing $tools"
            apt-get install -y $tools
        fi
    done
}

```

## Complex Command (Volatility Setup):

```

echo "[!] 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 "[*] Running volatility help manuel and adjustring 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

```

**Explanation:** This sequence automates the retrieval of a tool that isn't always in the apt repository. It downloads, extracts, and renames the file so the rest of the script can just call ./vol.

## Function: CARVERS

**What it does:** Runs file extraction tools.

**Process:** It runs Foremost, Strings, Binwalk, Bulk Extractor, and Exiftool. And it configures Scalpel before running it.

```

function CARVERS()
{

#1.4 Use different carvers to automatically extract data
echo "[!!]Running multiple carvers to gather as much information as possible."

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

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

#Binwalk
echo "-----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 "-----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 "[!] Scalpel config not found at $CONF_FILE"
fi

# Bulk Extractor
echo "-----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 "-----Running Exiftool-----"
mkdir -p "$RESULTS_DIR/memory/exiftool"
exiftool "$MEM_FILE" > "$RESULTS_DIR/memory/exiftool/exif_metadata.txt" 2>/dev/null
}

```

## Complex Command (Sed Configuration):

```

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 "[!] Scalpel config not found at $CONF_FILE"
fi

```

## **Explanation:**

The first sed comments out every line in the config file (disabling everything) to start fresh.

The loop then finds specific file extensions inside the file and removes the # (uncomments them), enabling only the file types we want to find.

## **Function: VOL**

**What it does:** Performs memory analysis. **Process:**

1. Runs imageinfo to determine if the file is valid memory.
2. Extracts the Suggested Profile and extracting with text manipulation the profile.
3. Loops through the “plugins” (pslist, connscan, hivelist...) using that

profile.

```
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 "[!] $MEM_FILE is not a memory file"
        else
            echo "[(:] $MEM_FILE can be analyzed using volatility"
            ./vol -f "$MEM_FILE" imageinfo > "$RESULTS_DIR/memory/imageinfo.txt" 2>&1
    fi
    #2.2 Find the memory profile and save it into a variable.
    export PRO=$(./vol -f "$MEM_FILE" imageinfo 2>/dev/null | grep "Suggested" | awk '{print $4}' | tr -d ",")
    #2.3 Display the running processes.
    #2.4 Display network connections.
    #2.5 Attempt to extract registry information.
    echo "[!] getting more information with volatility flags"
    PLUGINS="pslist pstree filescan connections connscan sockets hivelist"
    PRO=$(./vol -f "$MEM_FILE" imageinfo 2>/dev/null | grep "Suggested" | awk '{print $4}' | tr -d ", ")
    for i in $PLUGINS
    do
        echo "[*]-----parsing memory using volatility $i-----"
        ./vol -f "$MEM_FILE" --profile=$PRO $i > "$RESULTS_DIR/memory/file-$i.txt"2>/dev/null
    done
}
```

## **Complex Command (Profile Extraction):**

```
#2.2 Find the memory profile and save it into a variable.  
export PRO=$(./vol -f "$MEM_FILE" imageinfo 2>/dev/null | grep "Suggested" | awk '{print $4}' | tr -d ",")  
#3.2 Display the running processes
```

grep "Suggested": Isolates the line with the OS suggestions.

awk '{print \$4}': Grabs the fourth word (the first profile suggestion).

tr -d ",": Deletes any commas or spaces to ensure the variable \$PRO is clean.

## Function: PCAP

### **What it does:**

**PCAP:** Uses find . -type f -name "packets.pcap" to locate any network traffic extracted by Bulk Extractor and calculates its size.

```
function PCAP ()  
{  
    #1.6 attempt to extract network traffic; if found, display to the user the location and size  
    pcapF=$(find . -type f -name "packets.pcap")  
    if [ -n "$pcapF" ]  
    then  
        size=$(ls -lh "$pcapF" | awk '{print $(NF -4)})'  
        echo "PCAP found: $pcapF (Size: $size)"  
    fi
```

## Function: STR

### **What it does:**

- **STR:** Loops through a list of keywords (password, username, http) and runs strings | grep "\$i" to save lines containing those words to specific text files.

```
function STR()  
{  
    #1.7 Check for human-readable (exe files, passwords, usernames, etc.).  
    STRINGS_LIST="exe password username http dll"  
    for i in $STRINGS_LIST  
    do  
        strings "$MEM_FILE" | grep "$i" > "$RESULTS_DIR/file_$i.txt"  
    done  
}
```

## Statistics output

### **Command:**

```

echo "--- Analysis Statistics ---"
echo "Total Time: $TOTAL_TIME seconds"
echo "Total Files Found: $TOTAL_FILES"
echo "Report saved to: $REPORT_FILE"
-- -- -- -- -

```

- **Explanation:** By subtracting the time captured at the start from the time captured at the end (date +%s), the script calculates exactly how many seconds the analysis took. then uses the information and write's it to the report file

```

} {
    echo ""
    echo "--- Final Statistics ---"
    echo "Analysis Completed at: $(date)"
    echo "Total Time Taken: $TOTAL_TIME seconds"
    echo "Total Files Extracted: $TOTAL_FILES"
    echo "-----"
} >> "$REPORT_FILE"

```

## Zip command

### Command:

```

#3.3 Zip the extracted files and the report file
ZIP_NAME="forensics_results_${date +%F %H-%M-%S}.zip"
zip -r "$ZIP_NAME" "$RESULTS_DIR">> /dev/null 2>&1 && rm -rf "$RESULTS_DIR"
if [ -f "$ZIP_NAME" ]
then
    echo "[(:] Zip successful. Original folder removed."
else
    echo "[!] Error: Zip failed! I kept the folder '$RESULTS_DIR' so you don't lose data."
fi

```

- **Explanation:** By compressing the results directory into a timestamped ZIP file, the script organizes the output. It uses a safety check (&&) to delete the original folder only if the archiving was successful, ensuring no data is lost before verifying the file creation.

## Summary

The script is a combination of the most powerful Linux forensic tools. It creates a dedicated directory, cleans up previous runs , installs dependencies dynamically, and outputs organized folders for the tools results

Thank you for reading and using my script !