# Practica 1

# Análisis forense de sistemas Linux

El objetivo de esta práctica es desarrollar un script personalizado que pueda ejecutarse desde una unidad USB externa conectada a la computadora. Este script tendrá la capacidad de realizar tareas como la copia de registros a la unidad USB externa y recopilar información relevante, como la fecha, hora, usuarios registrados, árbol de procesos y tiempo de actividad del sistema, entre otros.

La primera función del script, denominada **'starting'**, se encarga de crear un directorio con el nombre especificado y la fecha actual en el momento en que se ejecuta el script.

```
function starting() {
    echo ""
    echo "Iniciando la recopilación. Espere..."
    # Obtener el directorio del script
    SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
    # Solicitar el nombre al usuario
    read -p "Por favor, ingresa un nombre para el directorio: " USER_NAME
    # Crear un directorio con el nombre y la fecha actual
    TARGET_DIR="$USER_NAME-$(date +'%Y%m%d_%H%M%S')"
    FULL_TARGET_DIR="$SCRIPT_DIR/$TARGET_DIR"
    # Verificar si el directorio ya existe
    if [ -d "$FULL_TARGET_DIR" ]; then
        echo "El directorio $FULL_TARGET_DIR ya existe. Por favor, elige un nombre diferente."
        exit 1
    fi
    # Crear el directorio
    mkdir -p "$FULL_TARGET_DIR"
    echo ""
}
```

Esta función llamada "**logs**" realiza la tarea de copiar el contenido de las carpetas de registro (logs) a un directorio de destino específico.

```
function logs() {
    printf "Copiar el contenido de las carpetas de "registro" (logs)......"
    DESTINATION_DIR="$FULL_TARGET_DIR/registros" > /dev/null 2>&1
    LOGS_FOLDER="/var/log" > /dev/null 2>&1
    mkdir -p "$DESTINATION_DIR" > /dev/null 2>&1
    cp -r "$LOGS_FOLDER"/* "$DESTINATION_DIR" > /dev/null 2>&1
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_info**" obtiene información básica del sistema. Primero, establece la ruta y nombre de archivo para guardar la información en un archivo de texto. Luego, registra la fecha y hora actuales, así como el nombre del host en el archivo de información. Después, obtiene y guarda el tiempo de actividad del sistema, información del entorno y variables de entorno en el mismo archivo.

```
function system_info() {
    printf "Obteniendo información basica......"
    INFO_FILE="$FULL_TARGET_DIR/info.txt"
    current_date=$(date +"%Y-%m-%d %H:%M:%S")
    echo -e "-----Información del sistema-----\nFecha del sistema: $current_date" > "$INFO_FILE"
    host_name=$(hostname)
    echo "Nombre del host: $host_name" >> "$INFO_FILE"
    echo "-----ACTIVITY SYSTEM-----" >> "$INFO_FILE"
    uptime >> "$INFO_FILE"
    echo "-----ENTORNO-----" >> "$INFO_FILE"
    lsb_release -a >> "$INFO_FILE"
    uname -a >> "$INFO_FILE"
    echo "-----VARIABLES DE ENTORNO-----" >> "$INFO_FILE"
    env >> "$INFO_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_cpu**" se encarga de obtener información relacionada con la CPU del sistema. Primero, establece la ruta y nombre de archivo para guardar la información en un archivo de texto específico. Luego, registra en dicho archivo la información detallada de la CPU utilizando el comando "lscpu".

```
function system_cpu() {
    printf "Obteniendo información de la CPU......"
    CPU_FILE="$FULL_TARGET_DIR/cpu.txt"
    echo "----------CPU----------" >> "$CPU_FILE"
    lscpu >> "$CPU_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_users**" recopila información sobre los usuarios del sistema, incluyendo usuarios conectados, últimos inicios de sesión, detalles de los archivos de configuración de usuarios y grupos, el último inicio de sesión por usuario, el usuario actual, los grupos del usuario actual y el historial de comandos para cada usuario.

```
function system_users() {
    printf "Obteniendo información de los usuarios......"
    USERS_FILE="$FULL_TARGET_DIR/users.txt"
    echo "-----USERS-----" >> "$USERS_FILE"
    who >> "$USERS_FILE"
    echo "-----LAST LOGINS-----" >> "$USERS_FILE"
    last >> "$USERS_FILE"
    echo "-----/etc/passwd-----" >> "$USERS_FILE"
    cat /etc/passwd >> "$USERS_FILE"
    echo "-----/etc/group-----" >> "$USERS_FILE"
    cat /etc/group >> "$USERS_FILE"
    echo "-----LAST LOGIN BY USER-----" >> "$USERS_FILE"
    last | awk '!seen[$1]++' >> "$USERS_FILE"
    echo "-----WHOAMI-----" >> "$USERS_FILE"
    logname >> "$USERS_FILE"
    echo "-----GROUPS BY USER-----" >> "$USERS_FILE"
    id -nG >> "$USERS_FILE"
    echo "-----HISTORY TERM-----" >> "$USERS_FILE"
    for user in $(awk -F: '{print $1}' /etc/passwd); do
        history_file="/home/$user/"

        # Determinar la shell del usuario y ajustar el archivo de historial correspondiente
        shell=$(awk -F: -v user="$user" '$1 == user {print $7}' /etc/passwd)
        case "$shell" in
            */bash) history_file+="bash_history" ;;
            */zsh)  history_file+=".zsh_history" ;;
            *)      history_file+="unknown_shell_history" ;;
        esac

        # Verificar si el archivo de historial existe y agregarlo al archivo info.txt
        if [ -f "$history_file" ]; then
             echo -e "\nHistorial de comandos para el usuario $user:\n$(sudo -u $user cat $history_file
2>/dev/null)" >> "$USERS_FILE"
        fi
    done
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_process**" recopila información sobre los procesos en ejecución, incluyendo una lista detallada de los procesos y la estructura jerárquica de los mismos.

```
function system_process() {
    printf "Obteniendo información de los procesos......"
    PROCESS_FILE="$FULL_TARGET_DIR/process.txt"
    echo "-----PROCESS PSLIST-----" >> "$PROCESS_FILE"
    ps aux >> "$PROCESS_FILE"
    echo "-----PROCESS PSTREE-----" >> "$PROCESS_FILE"
    pstree -p >> "$PROCESS_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_disk**" obtiene información sobre los discos del sistema, incluyendo el espacio utilizado y disponible en los sistemas de archivos montados, información de particionamiento de disco y estadísticas de uso del disco.

```
function system_disk() {
    printf "Obteniendo información sobre los discos......"
    DISK_FILE="$FULL_TARGET_DIR/disk.txt"
    echo "-----DISK MOUNT-----" >> "$DISK_FILE"
    df -h >> "$DISK_FILE"
    echo "-----DISK PARTITION-----" >> "$DISK_FILE"
    sudo fdisk -l >> "$DISK_FILE"
    echo "-----DISK USE-----" >> "$DISK_FILE"
    iostat >> "$DISK_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_kernel**" obtiene información sobre el kernel del sistema, incluyendo parámetros y configuraciones, así como información almacenada en "/proc/cmdline".

```
function system_kernel() {
    printf "Obteniendo información del kernel......"
    KERNEL_FILE="$FULL_TARGET_DIR/kernal.txt"
    echo "-----KERNEL-----" >> "$KERNEL_FILE"
    sudo sysctl -a >> "$KERNEL_FILE"
    echo "-----KERNEL PARAMETERS-----" >> "$KERNEL_FILE"
    cat /proc/cmdline >> "$KERNEL_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_memory**" obtiene información sobre el uso de memoria del sistema, mostrando los dos procesos principales en términos de uso de memoria.

```
function system_memory() {
    printf "Obteniendo información de la memoria......"
    MEMORY_FILE="$FULL_TARGET_DIR/memory.txt"
    echo "-----MEMORY-----" >> "$MEMORY_FILE"
    ps -o pid,%mem,vsz,rss,cmd --sort=-%mem | head -n 2 >> "$MEMORY_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_services**" obtiene información sobre los servicios en ejecución del sistema, mostrando una lista de los servicios activos.

```
function system_services() {
    printf "Obteniendo información de los servicios......"
    SERVICES_FILE="$FULL_TARGET_DIR/services.txt"
    echo "-----SERVICES-----" >> "$SERVICES_FILE"
    systemctl list-units --type=service --state=running >> "$SERVICES_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**system_modules**" obtiene información sobre los módulos del sistema, mostrando una lista de los módulos cargados en el kernel.

```
function system_modules() {
    printf "Obteniendo información de los modulos......"
    MODULES_FILE="$FULL_TARGET_DIR/modules.txt"
    echo "-----MODULES-----" >> "$MODULES_FILE"
    lsmod >> "$MODULES_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "s**ystem_network**" recopila información de red del sistema, incluyendo conexiones, interfaces, puertos, enrutamiento, ARP, hosts permitidos y denegados, resolución DNS y configuraciones de puerta de enlace y DNS, guardando los resultados en un archivo específico.

```
function system_modules() {
    printf "Obteniendo información de la red......"
    RED_FILE="$FULL_TARGET_DIR/red.txt"
    echo "-----RED-----" >> "$RED_FILE"
    sudo netstat -anp >> "$RED_FILE"
    echo "-----INTERFACE-----" >> "$RED_FILE"
    ip a >> "$RED_FILE"
    echo "-----SOCKETS-----" >> "$RED_FILE"
    ss -s >> "$RED_FILE"
    echo "-----PORTS-----" >> "$RED_FILE"
    ss -tuln >> "$RED_FILE"
    echo "-----ROUTING TABLE-----" >> "$RED_FILE"
    ip route show >> "$RED_FILE"
    echo "-----ARP TABLE-----" >> "$RED_FILE"
    arp -a >> "$RED_FILE"
    echo "-----INTERFACE INFO-----" >> "$RED_FILE"
    ip -s link show >> "$RED_FILE"
    echo "-----ALLOWED HOSTS-----" >> "$RED_FILE"
    cat /etc/hosts.allow >> "$RED_FILE"
    echo "-----DENY HOSTS-----" >> "$RED_FILE"
    cat /etc/hosts.deny >> "$RED_FILE"
    echo "-----DNS RESOLUTION-----" >> "$RED_FILE"
    cat /etc/resolv.conf >> "$RED_FILE"
    echo "-----GATEWAY & DNS DYNAMIC-----" >> "$RED_FILE"
    nmcli dev show | grep -E 'IP4\.GATEWAY|IP4\.DNS' >> "$RED_FILE"
    echo -e "\e[32mOK\e[0m"
}
```

La función "**suid_guid_files**" busca archivos con permisos SUID o GUID en el sistema y los copia a un directorio específico.

```
function suid_guid_files() {
    printf "Buscando archivos con permisos SUID o GUID..."
    SUGU_DIR="$FULL_TARGET_DIR/suid_guid_files"
    mkdir -p "$SUGU_DIR"
    find / -type f \( -perm -2000 -o -perm -4000 \) -exec cp -p {} "$SUGU_DIR/" \; 2>/dev/null
    echo -e "\e[32mOK\e[0m"
}
```

El resultado obtenido al ejecutar el script se vería algo así:


Se genera un archivo por función que contiene la información recopilada.