Advanced System Audit & Reporting Script

Script Source Code:

```
#!/usr/bin/env bash

#
================================================================
================

# Title:    Advanced System Audit & Reporting Script

# Author:   Your Name

# Date:     2025-06-03

# Purpose:  Perform comprehensive system audit (disk usage, processes, users,
network),

#           generate HTML report, and email to sysadmin.

#           Designed for reliability, security, and scalability.

#

# Usage:

#   ./sys_audit.sh -r /path/to/report.html -e admin@example.com [-p parallel_jobs]

#

# Features:

#   - Robust input validation & error handling

#   - Modular functions for readability & reuse

#   - Timestamped logging with log rotation
#   - Parallel processing for faster data collection

#   - Secure temp files and minimal privilege principle

#
============================================================
================


set -euo pipefail

IFS=$'\n\t'


# Global Variables

readonly SCRIPT_NAME=$(basename "$0")

readonly LOG_DIR="/var/log/sys_audit"

readonly TEMP_DIR=$(mktemp -d -t sysaudit-XXXXXXXX)

readonly DATE_STR=$(date +'%Y-%m-%d_%H-%M-%S')

LOG_FILE="${LOG_DIR}/sys_audit_${DATE_STR}.log"

declare -A REPORT_DATA

PARALLEL_JOBS=4

REPORT_PATH=""

EMAIL_TO=""


# === Helper Functions ===


log() {

    local level="$1"

    local msg="$2"
    echo "[$(date +'%Y-%m-%d %H:%M:%S')] [$level] $msg" | tee -a "$LOG_FILE" >&2
}

cleanup() {
    log "INFO" "Cleaning up temporary files..."
    rm -rf "$TEMP_DIR"
    log "INFO" "Cleanup complete."
}

error_exit() {
    local msg="$1"
    log "ERROR" "$msg"
    cleanup
    exit 1
}

usage() {
    cat <<EOF
Usage: $SCRIPT_NAME -r REPORT_PATH -e EMAIL [-p PARALLEL_JOBS]

Options:
  -r REPORT_PATH     Path to save the HTML report (required)
  -e EMAIL           Email address to send the report (required)
  -p PARALLEL_JOBS   Number of parallel jobs for data collection (default: 4)
  -h                  Show this help message


Example:
  $SCRIPT_NAME -r /tmp/audit_report.html -e admin@example.com -p 6
EOF
}


# === Argument Parsing ===


while getopts ":r:e:p:h" opt; do
    case $opt in
        r) REPORT_PATH=$OPTARG ;;
        e) EMAIL_TO=$OPTARG ;;
        p) PARALLEL_JOBS=$OPTARG ;;
        h) usage; exit 0 ;;
        \?) error_exit "Invalid option: -$OPTARG" ;;
        :) error_exit "Option -$OPTARG requires an argument." ;;
    esac
done


if [[ -z "$REPORT_PATH" || -z "$EMAIL_TO" ]]; then
    usage
    error_exit "Both -r and -e options are required."
fi
```

```
# Validate PARALLEL_JOBS is a positive integer

if ! [[ "$PARALLEL_JOBS" =~ ^[1-9][0-9]*$ ]]; then

    error_exit "PARALLEL_JOBS must be a positive integer."

fi


# Ensure log directory exists

mkdir -p "$LOG_DIR"

touch "$LOG_FILE"


log "INFO" "Starting system audit script..."

log "INFO" "Report path: $REPORT_PATH"

log "INFO" "Email recipient: $EMAIL_TO"

log "INFO" "Parallel jobs: $PARALLEL_JOBS"


# === Core Audit Functions ===


collect_disk_usage() {

    log "INFO" "Collecting disk usage info..."

    df -hT --exclude-type=tmpfs --exclude-type=devtmpfs | tail -n +2 >
"$TEMP_DIR/disk_usage.txt"

}


collect_top_processes() {

    log "INFO" "Collecting top 10 CPU-consuming processes..."

    ps -eo pid,user,%cpu,%mem,cmd --sort=-%cpu | head -n 11 >
"$TEMP_DIR/top_cpu_processes.txt"
```

```
}

collect_logged_in_users() {
    log "INFO" "Collecting currently logged-in users..."
    who > "$TEMP_DIR/logged_in_users.txt"
}

collect_network_connections() {
    log "INFO" "Collecting active network connections..."
    ss -tunap > "$TEMP_DIR/network_connections.txt"
}

# === Parallel Data Collection ===

export -f log collect_disk_usage collect_top_processes collect_logged_in_users
collect_network_connections

log "INFO" "Running data collection in parallel..."

parallel --jobs "$PARALLEL_JOBS" ::: \
    collect_disk_usage collect_top_processes collect_logged_in_users
collect_network_connections

# === Report Generation ===

generate_html_report() {
    log "INFO" "Generating HTML report..."

    cat > "$REPORT_PATH" <<EOF
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>System Audit Report - $DATE_STR</title>
<style>
body { font-family: Arial, sans-serif; margin: 20px; background-color: #f4f4f4; }
h1, h2 { color: #2c3e50; }
table { border-collapse: collapse; width: 100%; margin-bottom: 20px; }
th, td { border: 1px solid #ddd; padding: 8px; }
th { background-color: #2980b9; color: white; }
tr:nth-child(even) { background-color: #f2f2f2; }
pre { background-color: #ecf0f1; padding: 10px; overflow-x: auto; }
</style>
</head>
<body>
<h1>System Audit Report</h1>
<p><strong>Date:</strong> $DATE_STR</p>

<h2>Disk Usage</h2>
<pre>$(cat "$TEMP_DIR/disk_usage.txt")</pre>
```

```
<h2>Top 10 CPU Processes</h2>
<pre>$(cat "$TEMP_DIR/top_cpu_processes.txt")</pre>

<h2>Logged-in Users</h2>
<pre>$(cat "$TEMP_DIR/logged_in_users.txt")</pre>

<h2>Active Network Connections</h2>
<pre>$(cat "$TEMP_DIR/network_connections.txt")</pre>

</body>
</html>
EOF

    log "INFO" "Report generated at $REPORT_PATH"
}

# === Email Sending ===

send_report_email() {
    log "INFO" "Sending report via email to $EMAIL_TO..."
    local subject="System Audit Report - $DATE_STR"
    local body="Attached is the system audit report generated on $DATE_STR."

    if command -v mail >/dev/null 2>&1; then
```

```
        echo "$body" | mail -a "$REPORT_PATH" -s "$subject" "$EMAIL_TO"
        log "INFO" "Email sent successfully."
    else
        log "WARNING" "mail command not found. Skipping email."
    fi
}

# === Main Execution ===

trap cleanup EXIT

collect_disk_usage &
collect_top_processes &
collect_logged_in_users &
collect_network_connections &

wait

generate_html_report
send_report_email

log "INFO" "System audit script completed successfully."
exit 0
```