intelligent_system_hardening_audit_tool.sh

A Rare, Advanced Bash Script for Automated System Security Auditing, Data Integrity Enforcement, and Compliance Reporting (SEO Keywords: Bash security script, advanced bash programming, Linux hardening, system audit automation, DevSecOps bash tool)

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

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

# Title   : Intelligent System Hardening & Audit Tool for Linux Servers

# Author  : Maria Sultana

# Date    : 2025-07-19

# Version : v3.7.9

# Usage   : ./intelligent_system_hardening_audit_tool.sh

# Purpose : An extremely advanced and rare Bash script to automate Linux

#           server hardening, security auditing, compliance enforcement,

#           and real-time log monitoring with AI-inspired intelligence.

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


set -euo pipefail

IFS=$'\n\t'


#============[ SEO Keywords Included ]=========================

# High-ranked keywords used: bash system audit script, DevSecOps bash tool,
```

```
# bash for security automation, Linux hardening bash script,
# intelligent bash logging, advanced bash monitoring tool,
# bash log parsing AI, audit tool for compliance Linux

#============[ Global Constants & Rare Techniques ]=========================

readonly SCRIPT_VERSION="3.7.9"
readonly LOG_DIR="/var/log/secure-audit"
readonly SECURE_BACKUP="/var/backups/secure-data"
readonly AUDIT_REPORT="/var/log/secure-audit/report-$(date +%F).html"
readonly TMPDIR=$(mktemp -d /tmp/audit.XXXXXX)

trap 'rm -rf "$TMPDIR"' EXIT

#============[ Colorized Logging (AI-Inspired Logging) ]=====================

function log_info()    { echo -e "\033[1;34m[INFO]\033[0m    $*"; }
function log_warn()    { echo -e "\033[1;33m[WARNING]\033[0m $*"; }
function log_error()   { echo -e "\033[1;31m[ERROR]\033[0m   $*"; }
function log_success() { echo -e "\033[1;32m[SUCCESS]\033[0m $*"; }

#============[ Rare Function: Check Kernel Runtime Mitigations ]============

check_kernel_mitigations() {
  log_info "Checking Kernel Runtime Mitigations..."
```

```
  grep -E '^kernel.randomize_va_space|kernel.kptr_restrict|kernel.dmesg_restrict' /etc/sysctl.conf
|| true
  for sysctl in kernel.randomize_va_space kernel.kptr_restrict kernel.dmesg_restrict; do
    value=$(sysctl -n "$sysctl")
    if [[ "$value" -eq 0 ]]; then
      log_warn "$sysctl is not hardened (value: $value)"
    else
      log_success "$sysctl is hardened (value: $value)"
    fi
  done
}

#============[ Rare Function: Detect Hidden Processes Using /proc ]=========

detect_hidden_processes() {
  log_info "Scanning for hidden processes (Rootkit detection)..."
  for pid in $(ls -1 /proc/ | grep -E '^[0-9]+$'); do
    if [[ ! -e /proc/$pid/exe ]]; then
      log_warn "Process $pid has no executable path (might be hidden)"
    fi
  done
}

#============[ Rare Function: Check Filesystem Integrity using SHA-512 ]====
```

```
audit_integrity() {
  log_info "Performing filesystem integrity check (SHA-512)..."
  local db_file="$SECURE_BACKUP/fs_integrity.db"
  mkdir -p "$SECURE_BACKUP"

  if [[ -f "$db_file" ]]; then
    log_info "Comparing checksums against baseline..."
    while IFS= read -r line; do
      filepath=$(echo "$line" | cut -d':' -f1)
      old_hash=$(echo "$line" | cut -d':' -f2)
      if [[ -f "$filepath" ]]; then
        new_hash=$(sha512sum "$filepath" | awk '{print $1}')
        if [[ "$new_hash" != "$old_hash" ]]; then
          log_warn "File modified: $filepath"
        fi
      fi
    done < "$db_file"
  else
    log_info "Creating new baseline integrity database..."
    find /etc /bin /sbin /usr/bin /usr/sbin -type f -exec sha512sum {} + |
      awk '{print $2 ":" $1}' > "$db_file"
    log_success "Baseline created."
  fi
}
```

```
#============[ Advanced Feature: Real-Time Log Anomaly Detection ]==========

monitor_logs() {
  log_info "Monitoring /var/log/auth.log in real time for anomalies..."
  tail -Fn0 /var/log/auth.log | while read -r line; do
    if echo "$line" | grep -Ei "failure|invalid|denied|unauthorized"; then
      log_warn "Suspicious activity detected: $line"
    fi
  done
}

#============[ Rare Function: SSH Configuration Compliance ]================

check_ssh_hardening() {
  log_info "Checking SSHD configuration compliance..."
  local sshd_conf="/etc/ssh/sshd_config"

  declare -A checks=(
    ["PermitRootLogin"]="no"
    ["PasswordAuthentication"]="no"
    ["Protocol"]="2"
    ["PermitEmptyPasswords"]="no"
    ["MaxAuthTries"]="3"
  )
```

```
  for key in "${!checks[@]}"; do
    expected=${checks[$key]}
    actual=$(grep -E "^$key" "$sshd_conf" | awk '{print $2}')
    if [[ "$actual" != "$expected" ]]; then
      log_warn "$key is not compliant (Expected: $expected, Found: $actual)"
    else
      log_success "$key is compliant"
    fi
  done
}

#============[ Reporting System ]==========================

generate_report() {
  log_info "Generating HTML audit report at $AUDIT_REPORT"
  mkdir -p "$(dirname "$AUDIT_REPORT")"

  {
    echo "<html><head><title>System Audit Report</title></head><body>"
    echo "<h1>System Audit Report - $(hostname)</h1>"
    echo "<p><strong>Date:</strong> $(date)</p>"
    echo "<ul>"
    echo "<li>Kernel: $(uname -r)</li>"
    echo "<li>Hostname: $(hostname)</li>"
    echo "<li>User: $(whoami)</li>"
```

```
    echo "</ul>"
    echo "<pre>"
    uname -a
    echo ""
    df -h
    echo ""
    free -m
    echo "</pre>"
    echo "</body></html>"
  } > "$AUDIT_REPORT"

  log_success "Audit report generated: $AUDIT_REPORT"
}

#============[ Advanced Bash Flags Check (Uncommon Features) ]==============

check_bash_flags() {
  log_info "Verifying advanced bash flags (rare use)..."
  if [[ $- != *e* ]]; then
    log_warn "Bash 'set -e' not active — Script might ignore failures"
  fi
  if [[ $- != *u* ]]; then
    log_warn "Bash 'set -u' not active — Unbound variable issues possible"
  fi
  if [[ $- != *o* ]]; then
```

```
    log_warn "Bash 'set -o pipefail' not active — Piped errors might be missed"
  fi
  log_success "All necessary bash flags are correctly configured"
}

#============[ MAIN FUNCTION ENTRYPOINT ]==================================

main() {
  log_info "========== Starting Intelligent Bash System Audit =========="
  log_info "Version: $SCRIPT_VERSION"

  check_bash_flags
  check_kernel_mitigations
  detect_hidden_processes
  audit_integrity
  check_ssh_hardening
  generate_report

  log_info "========== Audit Completed =========="
  log_info "Use: tail -f $AUDIT_REPORT to view report in real-time."
  log_info "Or open $AUDIT_REPORT in a browser."
}

#============[ Background Monitor ]==========================
```

```
# Uncomment to enable real-time log monitor
# monitor_logs &

main "$@"
```

SEO Keyword Optimization Summary

Target Keywords included:

- advanced bash programming
- bash security script
- Linux hardening bash
- intelligent log monitoring bash
- DevSecOps bash automation
- rare bash functions
- hidden process detection Linux
- bash script for compliance audit
- AI-inspired bash audit tool
- advanced logging and reporting bash