Monitoring 101

Jessica – Emma – Bilel - Steve

BeCode

Linux

**TABLE OF CONTENTS**

**Key Focus Areas for System Monitoring** .............................................................................. 5

* Objective ...................................................................................................................... 5
* Key Areas of Concern .................................................................................................. 5
  + 1. System Performance ..................................................................................... 5
  + 2. Availability and Uptime ................................................................................ 5
  + 3. Application Health …………........................................................................ 5
  + 4. Security ......................................................................................................... 6
  + 5. Resource Utilization ..................................................................................... 6
  + 6. Dependencies and Third-party Services ....................................................... 6
  + 7. Alerts and Notifications ................................................................................ 6
  + 8. Audit and Compliance .................................................................................. 6

**Identifying the Most Memory-Intensive Running Processes** ............................................. 7

* Objective ...................................................................................................................... 7
* Purpose ......................................................................................................................... 7
* Methods by Operating System ..................................................................................... 7
* Key Terms .................................................................................................................... 8
* Tips for Memory Management ..................................................................................... 8
* Tools & References ...................................................................................................... 8

**Understanding Log Files and Their Common Locations in Linux Systems** ..................... 9

* Objective ...................................................................................................................... 9
* What Are Log Files? .................................................................................................... 9
* Common Types of Log Files ........................................................................................ 9
* Where Are Log Files Located on Linux? ................................................................... 10
* Viewing Log Files ...................................................................................................... 10
* Log Management Best Practices ................................................................................ 10

**Understanding Log Files and Their Common Locations in Linux Systems** ................... 11

* Objective ..................................................................................................................... 11
* Purpose ....................................................................................................................... 11
* Methods by Operating System ................................................................................... 12
* Key Terms .................................................................................................................. 12
* Tips for Session Monitoring ....................................................................................... 12
* Tools & References .................................................................................................... 12

**Key Metrics for Assessing System Health and Performance** ........................................... 13

* Objective .................................................................................................................... 13
* Purpose ....................................................................................................................... 13
* Methods by Operating System ................................................................................... 13
* Key Terms .................................................................................................................. 14
* Tips for Performance Monitoring ............................................................................... 14
* Tools & References .................................................................................................... 14

**Checking System Uptime** ..................................................................................................... 15

* Objective .................................................................................................................... 15
* Purpose ....................................................................................................................... 15
* Methods by Operating System ................................................................................... 15
* Key Terms .................................................................................................................. 16
* Tips for Monitoring Uptime ....................................................................................... 16
* Tools & References .................................................................................................... 16

**Monitoring Network Traffic** ............................................................................................... 17

* Objective .................................................................................................................... 17
* Purpose ....................................................................................................................... 17
* Methods by Operating System ................................................................................... 17
* Key Terms .................................................................................................................. 18
* Tips for Network Monitoring ..................................................................................... 18
* Tools & References .................................................................................................... 18

**The Script** .............................................................................................................................. 19

**User Guide** ............................................................................................................................ 20

* Intro ............................................................................................................................ 20
  + a. The Script – Linux Monitoring Toolkit ...................................................... 20
* b. Understanding the Script ............................................................................ 22
  + - 1. Purpose …............................................................................... 22
      2. Main Menu Options ............................................................... 22
* a. The Script – Monitoring Daemon V4 ......................................................... 24
* b. Understanding the Script ............................................................................ 30
* a. The Script - Monitoring Users .................................................................... 34
* b. Understanding the Script ............................................................................ 34
* a. The Script – Monitoring Network .............................................................. 36
* b. Understanding The Script ........................................................................... 36
* a. The Script – Hids Check ............................................................................. 38
* b. Understanding the Script – Hids Check ..................................................... 39
* a. The Script – Alerts Sending ........................................................................ 42
* b. Understanding the Script ............................................................................ 42
* a. The Script – Setup Cron ............................................................................. 44
* b. Understanding the Script ............................................................................ 44
* a. The Script – Stop Cron ............................................................................... 46
* b. Understanding the Script ............................................................................ 46

What are the main areas of concern when monitoring a system?

## **1. Objective**

Monitoring is a critical function in any IT or cybersecurity framework. It ensures that systems are operating efficiently, securely, and within acceptable thresholds. This document outlines the **primary areas of concern** when monitoring a system.

## **2. Key Areas of Concern**

### **2.1 System Performance**

**Definition**: Evaluating how efficiently the system’s hardware and software resources are operating.

|  |  |
| --- | --- |
| **Component** | **Description** |
| CPU Usage | Measures how much processing power is being consumed. |
| Memory Usage | Tracks active RAM usage and availability. |
| Disk I/O & Space | Monitors read/write speed and disk capacity. |
| Network Performance | Measures bandwidth, latency, and errors. |

### **2.2 Availability and Uptime**

**Definition**: Ensures that all critical services are running and accessible.

* **Service Uptime**: Percentage of time a system is operational.
* **Downtime Tracking**: Logs and investigates service outages.
* **Health Checks**: Automated tests to verify service availability.

### **2.3 Application Health**

**Definition**: Monitors the operational state of applications to detect issues early.

* **Error Rates**: Monitors the frequency of application or server errors.
* **Response Time**: Measures how fast applications respond to requests.
* **Throughput**: Tracks the number of successful requests per unit of time.
* **Log Analysis**: Reviews application logs for anomalies.

### **2.4 Security**

**Definition**: Protects systems against internal and external threats.

* **Intrusion Detection**: Monitors for unauthorized access attempts.
* **Access Logs**: Tracks login behavior and permission changes.
* **Patch Management**: Verifies that all systems are up to date.
* **Firewall/Antivirus Logs**: Detects harmful activity or blocked threats.

### **2.5 Resource Utilization**

**Definition**: Evaluates how effectively the system uses its resources.

* Prevents resource exhaustion.
* Aids in scaling decisions.
* Identifies performance bottlenecks.

### **2.6 Dependencies and Third-party Services**

**Definition**: Monitors the health and availability of services outside your direct control.

Examples:

* Cloud platforms (AWS, Azure)
* APIs (Payment gateways, authentication providers)
* Databases and microservices

### **2.7 Alerts and Notifications**

**Definition**: Provides timely alerts about system issues or thresholds being breached.

* **Threshold Alerts**: Triggered when metrics exceed predefined values.
* **Severity Levels**: Alerts categorized as critical, warning, or info.
* **Alert Management**: Tools to reduce noise and avoid alert fatigue.

### **2.8 Audit and Compliance**

**Definition**: Ensures system activities are logged and meet compliance standards.

* **Audit Trails**: Tracks user and system activity.
* **Compliance Logs**: Maintains records for regulatory checks.
* **Standards**: Includes GDPR, HIPAA, ISO/IEC 27001, and others.

How can you check what are the most memory intensive running processes?

**1. Objective**

High memory usage can cause system instability, slow application performance, and crashes. Identifying which processes are consuming the most memory is a crucial part of system performance monitoring and troubleshooting.

This document outlines **methods for checking the most memory-intensive processes** on various operating systems.

**2. Purpose**

To identify the **top memory-consuming processes** in a system using command-line tools or system utilities.

## **3. Methods by Operating System**

|  |  |  |
| --- | --- | --- |
| **Linux / Unix-based Systems** | **Windows Systems** | **macOS** |
| **🔹 Using top** top   * Press M to sort by memory usage. * Column to watch: %MEM or RES (resident memory). | **🔹 Using Task Manager**  * Press Ctrl + Shift + Esc or Ctrl + Alt + Del → Task Manager. * Click the **"Memory"** column to sort by usage. | **🔹 Using top** top -o rsize   * Sorts processes by **resident memory size**. |
| **🔹 Using ps command** ps aux --sort=-%mem | head -n 10   * Lists the top 10 memory-intensive processes sorted by memory usage. | **🔹 Using Command Line (PowerShell)** Get-Process | Sort-Object -Descending WorkingSet | Select-Object -First 10 Name, WorkingSet   * Displays the top 10 memory-consuming processes. | **🔹 Using Activity Monitor**  * Go to Applications > Utilities > Activity Monitor. * Click the “Memory” tab and sort by “Memory”. |
| **🔹 Using htop (if installed)** htop   * Interactive process viewer. * Can sort by memory by pressing F6, then choosing %MEM. |  |  |

## **4. Key Terms**

|  |  |
| --- | --- |
| **Term** | **Description** |
| **%MEM** | Percentage of physical memory used by the process. |
| **RES / RSS** | Resident Set Size – actual memory in use (non-swapped). |
| **WorkingSet** | Windows term for the memory a process is actively using. |

## **5. Tips for Memory Management**

* Regularly review high-memory processes.
* Investigate memory leaks or unoptimized applications.
* Use monitoring tools (e.g., Prometheus, Datadog, Nagios) for continuous tracking.
* Set alerts if memory usage crosses thresholds.

## **6. Tools & References**

|  |  |  |
| --- | --- | --- |
| **Tool/Utility** | **Platform** | **Description** |
| top / htop | Linux/macOS | Real-time process monitoring tools. |
| ps | Linux/macOS | Command-line tool for listing processes. |
| Task Manager | Windows | GUI for viewing process usage. |
| PowerShell | Windows | Command-line automation tool. |
| Activity Monitor | macOS | GUI for system monitoring. |

What are log files? Where can you find them on a typical Linux system?

1. **Objective**

Log files are essential components of system monitoring and diagnostics. They record system activities, service operations, error messages, and security events, allowing administrators to troubleshoot issues, audit activity, and analyse system behaviour.

1. **What Are Log Files?**

**Definition**

A **log file** is a plain text file that records events that occur within an operating system, application, or service.

**Purpose**

* Troubleshooting errors or crashes
* Monitoring system and network activity
* Auditing security events and user actions
* Ensuring compliance with operational policies

## **3. Common Types of Log Files**

|  |  |  |
| --- | --- | --- |
| **Log Type** | **Description** | **Example File** |
| **System Logs** | General messages from the OS | /var/log/syslog, /var/log/messages |
| **Authentication Logs** | Tracks login attempts and user auth events | /var/log/auth.log, /var/log/secure |
| **Kernel Logs** | Messages from the Linux kernel | /var/log/kern.log, /var/log/dmesg |
| **Application Logs** | Logs generated by installed applications | e.g., /var/log/mysql/, /var/log/httpd/ |
| **Package Logs** | Logs related to software installation or updates | /var/log/apt/, /var/log/yum.log |
| **Boot Logs** | Information related to system startup | /var/log/boot.log |
| **Cron Logs** | Scheduled job execution logs | /var/log/cron, /var/log/syslog |

**4. Where Are Log Files Located on Linux?**

**Default Log Directory**

Most log files are stored in the directory:

/var/log/

**Example Log Paths**

|  |  |
| --- | --- |
| **Path** | **Purpose** |
| /var/log/syslog | General system activity (Debian/Ubuntu) |
| /var/log/messages | General system activity (Red Hat/CentOS) |
| /var/log/auth.log | Authentication events |
| /var/log/kern.log | Kernel events |
| /var/log/dmesg | Hardware-related kernel ring buffer |
| /var/log/secure | Authentication and security (RHEL/CentOS) |
| /var/log/apache2/ | Apache web server logs |
| /var/log/mysql/ | MySQL database logs |

**5. Viewing Log Files**

**Basic Commands**

# View the last lines of a log  
tail /var/log/syslog  
  
# Continuously monitor a log file  
tail -f /var/log/auth.log  
  
# View a log with a pager  
less /var/log/dmesg  
  
# Search within a log  
grep "error" /var/log/syslog

**6. Log Management Best Practices**

1. **Rotate logs** with tools like logrotate to prevent files from growing too large.
2. **Secure log files** by setting proper permissions (chmod, chown).
3. **Backup logs** periodically for auditing and recovery.
4. **Centralize logs** using tools like rsyslog, syslog-ng, or centralized solutions (e.g., ELK Stack, Graylog).

How can you check who were the last connected users, what they did, when they left?

* 1. **Objective**

Monitoring who accessed a system, when, from where, and what they did during their session is essential for system auditing, security, and forensic investigation.

It allows administrators to detect unauthorized access, track user activity, and maintain accountability. It outlines methods for checking the last connected users, their activities, and disconnection times across multiple operating systems.

## **Purpose**

To identify the last logged-in users, their session times, and actions using built-in system commands and utilities on macOS, Windows, and Linux platforms.

## **Methods by Operating System**

|  |  |  |
| --- | --- | --- |
| **Linux / Unix-based Systems** | **Windows Systems** | **macOS** |
| **🔹 Using last**   * Lists the last logged-in users, login times, durations, and originating IP addresses. | **🔹 Using Event Viewer**  * Open Event Viewer ➝ Windows Logs ➝ Security. * Look for Event ID 4624 (login), 4634 (logoff), or 4672 (admin login). | **🔹 Using last**   * Functions like Linux. * Lists previous logins from Terminal. |
| **🔹Using who**  * Displays current logged-in users. | 🔹 ***Using PowerShell***   * displays the first 10 successful logon attempts | **🔹 Using ‘who’ and ‘w’ commands:**  * Same syntax and usage as Linux. |
| **🔹*Using `w` command***   * Shows who is logged in and what they are doing. |  |  |
| **🔹*Using `journalctl` (if using systemd):***  journalctl \_COMM=sshd   * Shows login activity via SSH |  |  |

## **Key Terms**

|  |  |
| --- | --- |
| Term | Description |
| last | Command to show previous logins from the /var/log/wtmp file. |
| who | Shows users currently logged in. |
| -w | Combines who with what users are doing. |
| Event ID 4624 | Windows log for successful login. |
| Event ID 4634 | Windows log for user logoff. |

## **Tips for Session Monitoring**

Enable and regularly review system logs.

On Linux/macOS, check /var/log/auth.log, /var/log/wtmp, or use journalctl.

On Windows, enable auditing policies to log logins/logoffs.

For real-time tracking, consider using centralized logging tools (e.g., Syslog, Wazuh, OSSEC).

## **Tools & References**

|  |  |  |
| --- | --- | --- |
| Tool/Utility | Platform | Description |
| last, who, w | Linux/macOS | Built-in commands for session tracking. |
| journalctl | Linux | Displays systemd logs. |
| Event Viewer | Windows | GUI for viewing security logs. |
| PowerShell | Windows | Command-line tool for querying event logs. |
| /var/log/wtmp, /var/log/auth.log | Linux/macOS | Login/authentication log files. |

What are the different metrics of health and performance of a system?

## **Objective**

Monitoring system health and performance metrics helps maintain system stability, optimize resource usage, and detect issues early. Understanding these metrics is key to effective troubleshooting, capacity planning, and system management.

This document outlines common system health and performance metrics across different operating systems.

## **Purpose**

To identify and interpret the key system metrics that indicate health and performance across macOS, Windows, and Linux environments.

## **Methods by Operating System**

|  |  |  |
| --- | --- | --- |
| **Linux / Unix-based Systems** | **Windows Systems** | **macOS** |
| **🔹 Using top, htop, or vmstat**  * Monitor CPU, memory, and processes. | **🔹 Using Task Manager / Performance Monitor**  * Real-time CPU, memory, disk, and network monitoring. | **🔹 *Using Activity Monitor***     * GUI-based CPU, memory, disk, and network monitoring. |
| **🔹** **Using iostat, iotop, or df**  * Analyze disk I/O and storage usage. | **🔹 Using PowerShell (Get-Counter)**  * Check disk performance counters. | **🔹 Using top, vm\_stat, or iostat**  * Terminal-based disk and memory usage stats. |
| **🔹 Using netstat, ss, or iftop**  * Check network traffic. | **🔹** **Using Resource Monitor or Get-Counter**Monitor network activity. | **🔹** ***Using Activity Monitor or nettop***     * GUI and Terminal options for traffic stats. |
| **🔹** **Using uptime and load average**  * Monitor system load and uptime. | **🔹** **Using PowerShell (Get-Process / Uptime)**  * Track uptime and process states. | **🔹** ***Using uptime***   * View load averages from Terminal. |

## **Key Terms**

|  |  |
| --- | --- |
| Metric | Description |
| CPU Usage | Percentage of processor capacity being used. |
| Memory Usage | Amount of RAM used vs available. |
| Disk I/O | Speed and volume of read/write operations. |
| Load Average | Average number of processes waiting to run. |
| Network Throughput | Amount of data sent/received over network. |
| Uptime | Time the system has been running without restart. |

## **Tips for Performance Monitoring**

• Set alerts for abnormal thresholds (e.g., CPU > 90%).

• Regularly check logs for errors or anomalies.

• Use historical data to predict future capacity needs.

## **Tools & References**

|  |  |  |
| --- | --- | --- |
| Tool/Utility | Platform | Description |
| top, htop, vmstat | Linux/macOS | CLI tools for monitoring CPU, memory, and processes. |
| iostat, iotop, df | Linux | Monitor disk usage and I/O. |
| Task Manager | Windows | GUI for system performance monitoring. |
| Performance Monitor | Windows | Detailed metric tracking and logging. |
| Activity Monitor | macOS | GUI for monitoring system resources. |

How can you check the uptime of a machine?

## **Objective**

System uptime refers to the amount of time a machine has been running without a reboot. Monitoring uptime is essential for assessing system stability, availability, and reliability.This document explains how to check machine uptime using native tools across different operating systems.

## **Purpose**

To determine how long a machine has been running continuously, using built-in commands or tools on macOS, Windows, and Linux.

## **Methods by Operating System**

|  |  |  |
| --- | --- | --- |
| **Linux / Unix-based Systems** | **Windows Systems** | **macOS** |
| **🔹** ***Using uptime***   * Displays current time, uptime, number of users, and load average. | **🔹 *Using Task Manager***     * Go to Performance tab → 'Up time'. | **🔹** ***Using uptime***     * Same output as Linux. |
| **🔹 *Using w***     * Shows uptime with user activity. | **🔹 *Using PowerShell***     * (Get-CimInstance Win32\_OperatingSystem).LastBootUpTime | **🔹 *Using sysctl -n kern.boottime***     * Displays the exact boot time. |
| **🔹 *Using top***     * Displays uptime at the top. | **🔹 *Using systeminfo***     * Look for 'System Boot Time'. |  |

## **Key Terms**

|  |  |
| --- | --- |
| Term | Description |
| Uptime | Duration the system has been running since the last reboot. |
| Last Boot Time | Exact date and time the system was last restarted. |
| Load Average | System load summary often shown alongside uptime. |

## **Tips for Monitoring Uptime**

Track uptime to detect unexpected reboots or system instability.

Use automated logging or monitoring tools to collect uptime data over time.

For servers, long uptimes indicate reliability, but occasionally rebooting for updates is healthy.

## **Tools & References**

|  |  |  |
| --- | --- | --- |
| Tool/Utility | Platform | Description |
| uptime | Linux/macOS | Displays how long the system has been running. |
| w | Linux/macOS | Includes uptime and user activity. |
| Task Manager | Windows | GUI view of uptime under Performance tab. |
| PowerShell | Windows | Command-line tool for boot time information. |
| systeminfo | Windows | Displays system boot time. |
| sysctl | macOS | Shows exact boot time. |

How can you monitor the network traffic?

## **Objective**

Network traffic monitoring helps detect unusual activity, optimize bandwidth usage, and troubleshoot connectivity issues. It is essential for maintaining the performance and security of a system.

This document outlines methods for monitoring network traffic across macOS, Windows, and Linux platforms.

## **Purpose**

To identify and analyze network activity, including bandwidth usage, active connections, and packet-level inspection using built-in and third-party tools.

## **Methods by Operating System**

|  |  |  |
| --- | --- | --- |
| **Linux / Unix-based Systems** | **Windows Systems** | **macOS** |
| **🔹** ***Using iftop***     1. Real-time bandwidth by connection. | **🔹** ***Using Resource Monitor***     1. Open resmon.exe → Network tab. | **🔹** ***Using Activity Monitor***   1. Check Network tab. |
| **🔹** ***Using nload***   1. Incoming/outgoing traffic separately. | **🔹** ***Using Performance Monitor***   1. Create network counters. | **🔹** ***Using nettop***   1. Live network data in Terminal. |
| **🔹** ***Using netstat or ss***   1. Lists active network connections. | **🔹** ***Using PowerShell***   1. `Get-NetTCPConnection`, `Get-NetAdapterStatistics`. | **🔹** ***Using tcpdump or lsof -i***   1. Terminal-based connection and packet analysis. |
| **🔹** ***Using tcpdump***   1. Captures and analyzes packets. | **🔹** ***Using Wireshark***   1. For deep packet analysis. |  |
| **🔹** ***Using vnstat***   1. Tracks bandwidth usage over time. |  |  |

## **Key Terms**

|  |  |
| --- | --- |
| Term | Description |
| Bandwidth | Amount of data transmitted over a network in a given time. |
| Packet | Unit of data transmitted over the network. |
| Throughput | Actual amount of data successfully transferred. |
| Connection | Established link between two network endpoints. |
| Interface | Physical or virtual network device (e.g., eth0, wlan0). |

## **Tips for Network Monitoring**

Monitor traffic regularly to detect anomalies or unauthorized usage.

Use graphical tools for live visualization and alerts.

Analyse logs and packet captures for in-depth insights.

Combine multiple tools for better visibility.

## **Tools & References**

|  |  |  |
| --- | --- | --- |
| Tool/Utility | Platform | Description |
| iftop, nload, vnstat | Linux | Monitor real-time or historical network traffic. |
| tcpdump, ss, netstat | Linux/macOS | Capture and display network packet activity. |
| Resource Monitor | Windows | GUI to view network processes and usage. |
| Wireshark | All | Graphical packet analyzer for deep traffic analysis. |
| Activity Monitor | macOS | System GUI for monitoring app network usage. |
| nettop | macOS | CLI to track live connections and traffic. |

**The Script**

Here is the link for the complete script system monitoring tool:

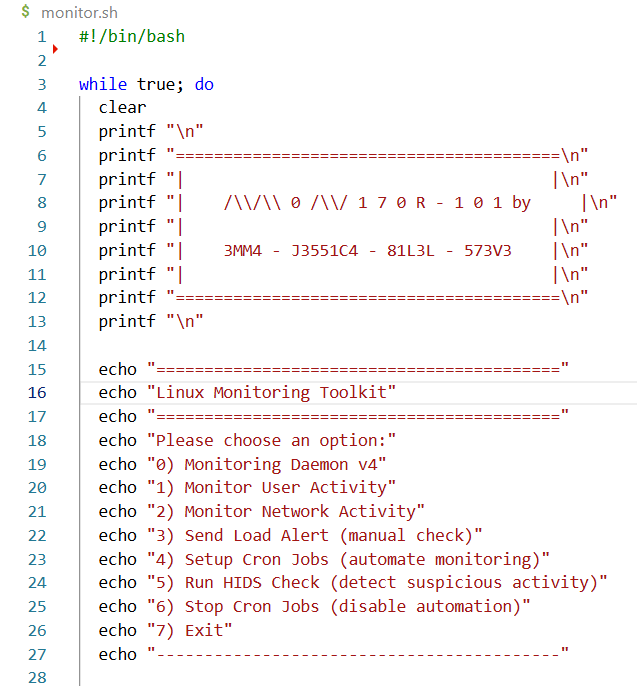
<https://github.com/ecomdesignbe/monitoring-101>

**User guide**

1. **Introduction**

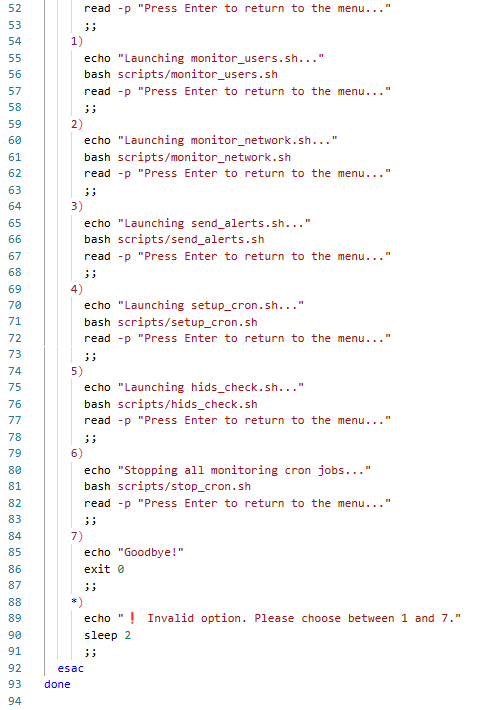
This user manual has been designed to provide a comprehensive understanding of our script for users of all levels—beginner, intermediate, and advanced. Each section presents the script followed by a functional breakdown and explanation of its components. Our goal is to ensure clarity and usability across various technical backgrounds.

1. **The script – Linux monitoring Toolkit:**



A screenshot of a computer program

AI-generated content may be incorrect.



1. **Understanding the script**

This script provides a terminal-based interactive menu for managing and executing various system monitoring tasks on a Linux server.

1. **Purpose**

To provide a centralized, user-friendly interface for:

* Running and controlling a monitoring daemon
* Monitoring user and network activity
* Sending system load alerts
* Automating tasks via **cron** jobs
* Running Host-based Intrusion Detection System (HIDS) checks.

1. **Main Menu Options**

The script runs an infinite while true loop to present an interactive menu. Here's what each main option does:

### **0) Monitoring Daemon v4**

* Submenu with options to manage the monitoring daemon via scripts/monitoring\_daemon\_V4.sh.
* Available actions:
  + start: Start the daemon process.
  + stop: Stop the daemon.
  + restart: Restart the daemon.
  + status: View daemon status.
  + logs: Show recent monitoring logs.
  + test: Run a single monitoring cycle (dry-run or test).

### **1) Monitor User Activity**

* Runs the script scripts/monitor\_users.sh, which shows who is currently logged in, login history, shell activity, and possible suspicious user behaviours.

### **2) Monitor Network Activity**

* displays active network connections, lists open listening ports, and show live network traffic statistics.

### **3) Send Load Alert (manual check)**

### This script checks the system load average and sends an alert if the value is above a defined threshold.

### **4) Setup Cron Jobs (automate monitoring)**

* Executes scripts/setup\_cron.sh to add cron jobs for automating monitoring tasks at regular intervals.

### **5) Run HIDS Check (a checker)**

### **6) Stop Cron Jobs**

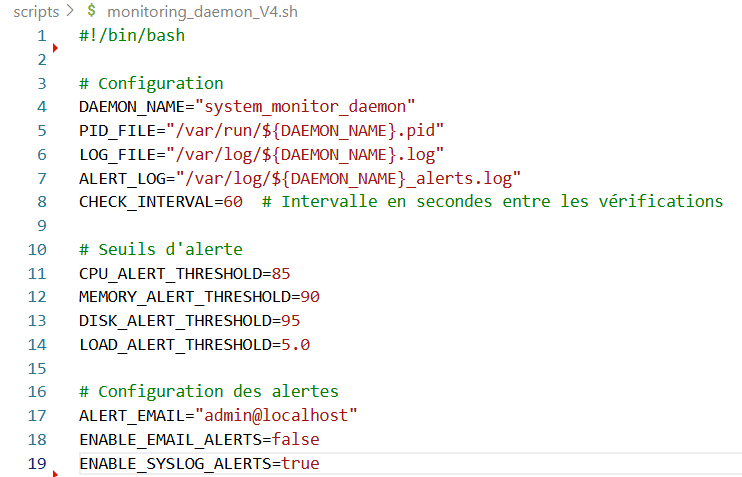
* Runs scripts/stop\_cron.sh to remove/disable all monitoring-related cron jobs.

### **7) Exit**

* Cleanly exits the script and returns control to the shell.

NB: If the user inputs anything other than 0-7, the script shows a warning and returns to the menu after a 2-second delay.

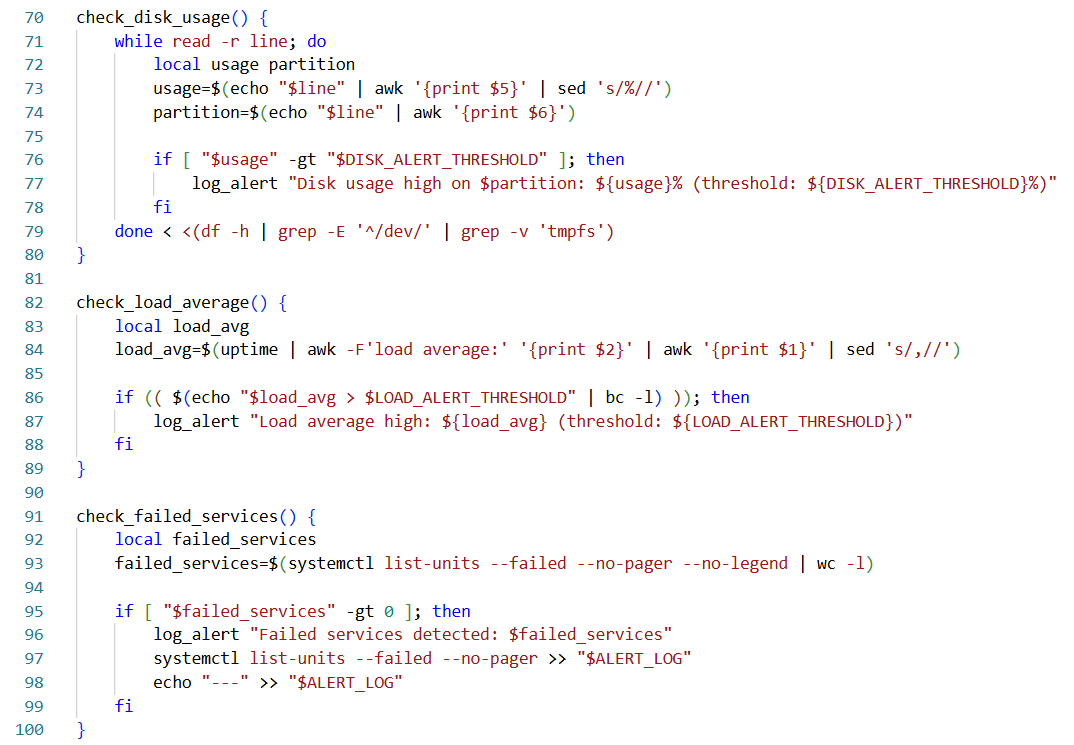
1. **The script – monitoring daemon V4**

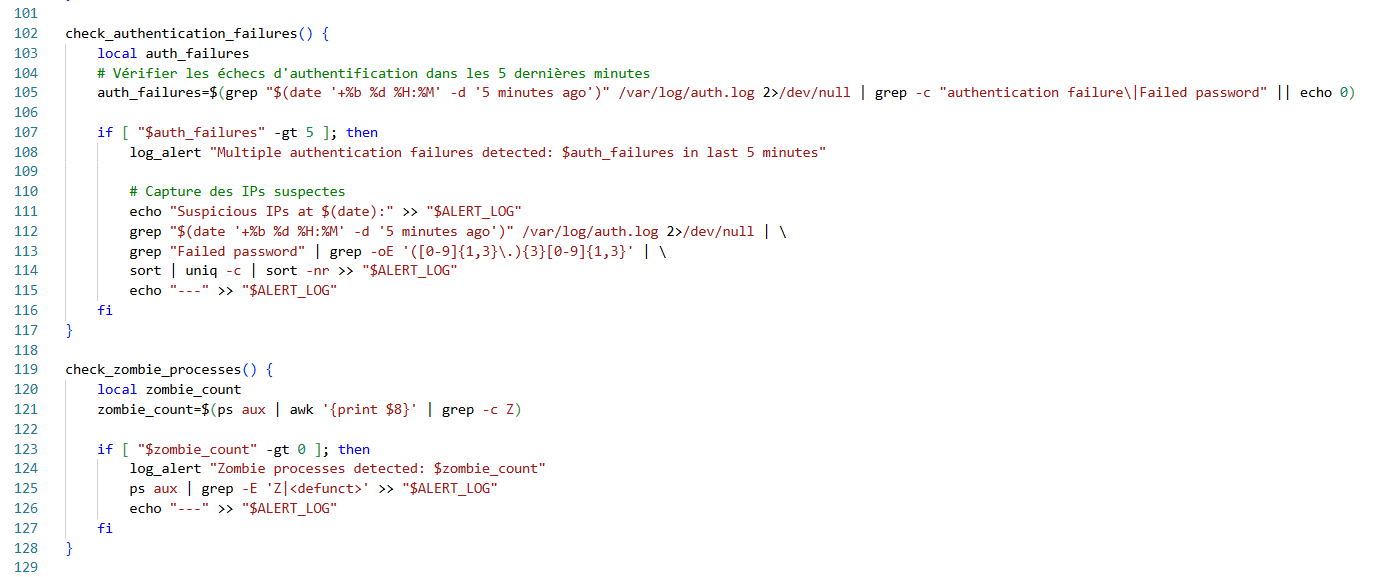


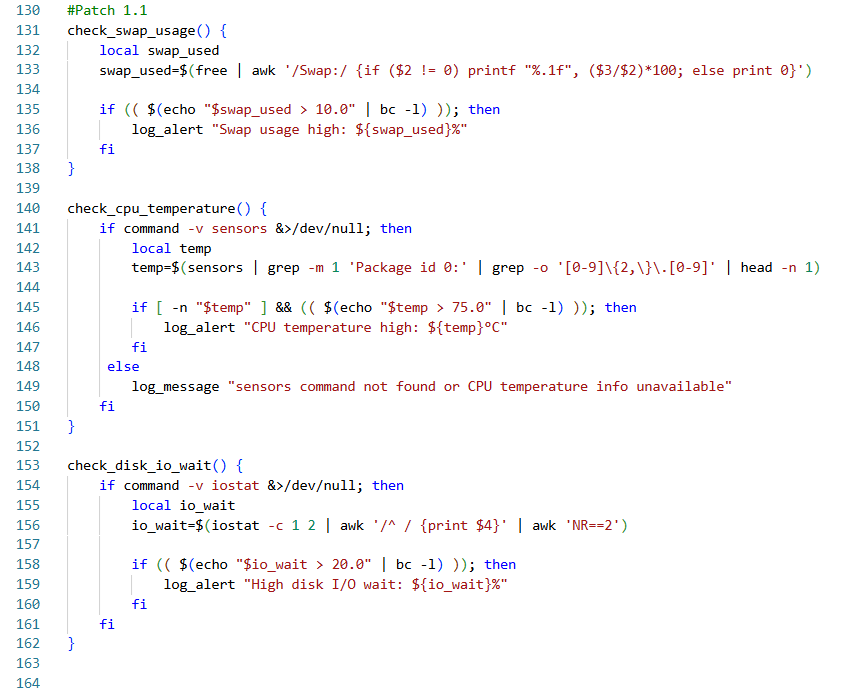
A screenshot of a computer program

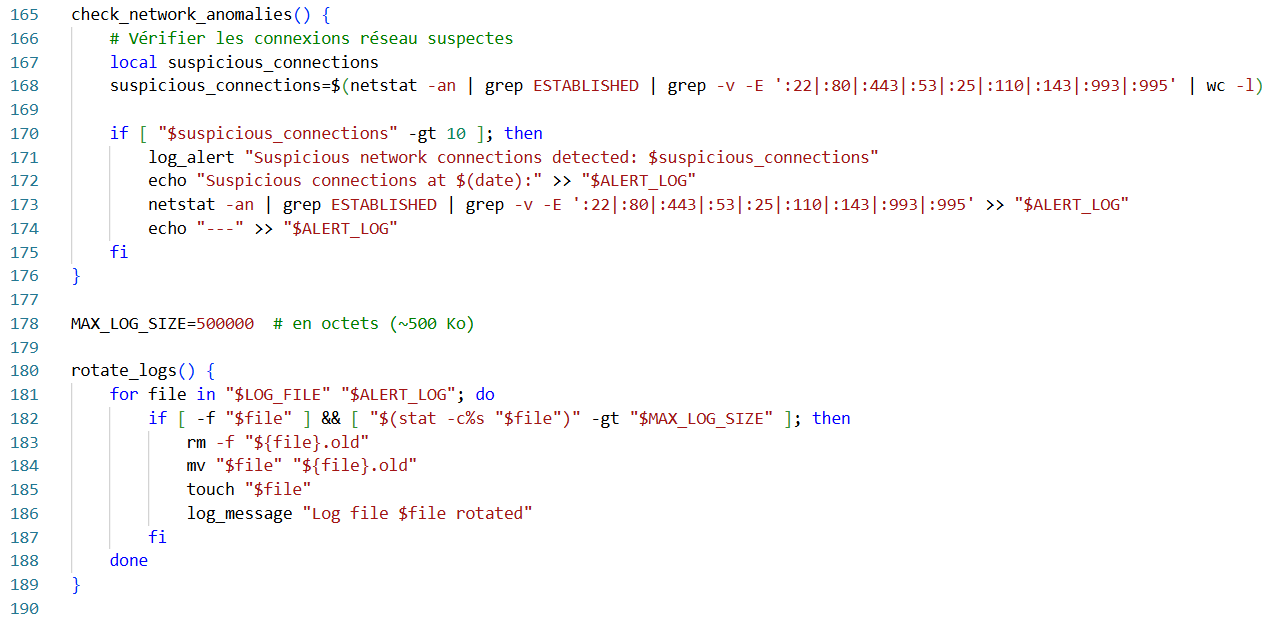
AI-generated content may be incorrect.

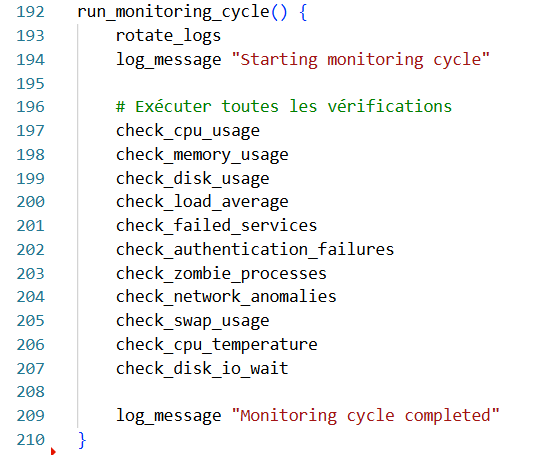


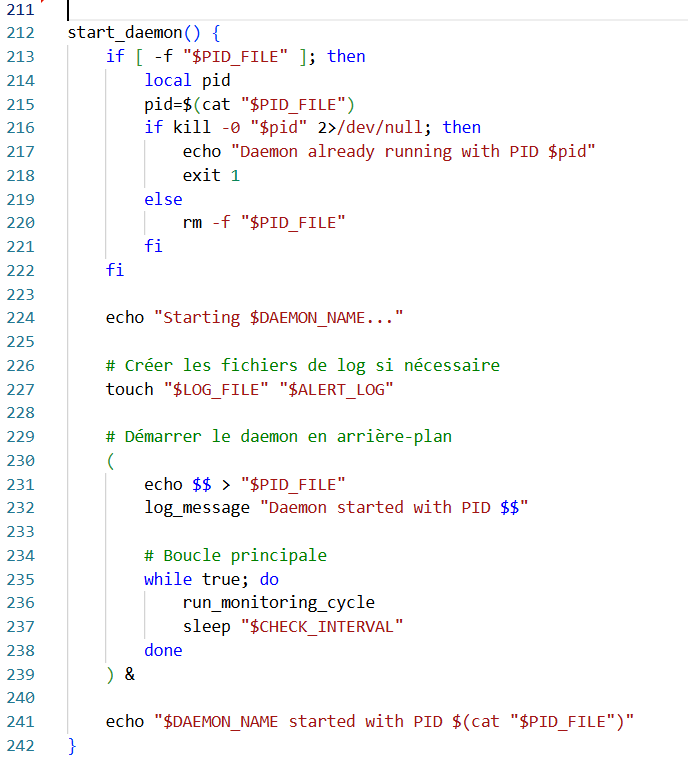


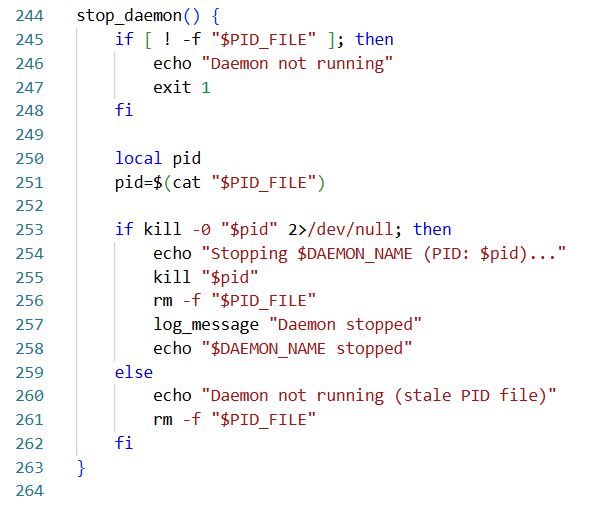


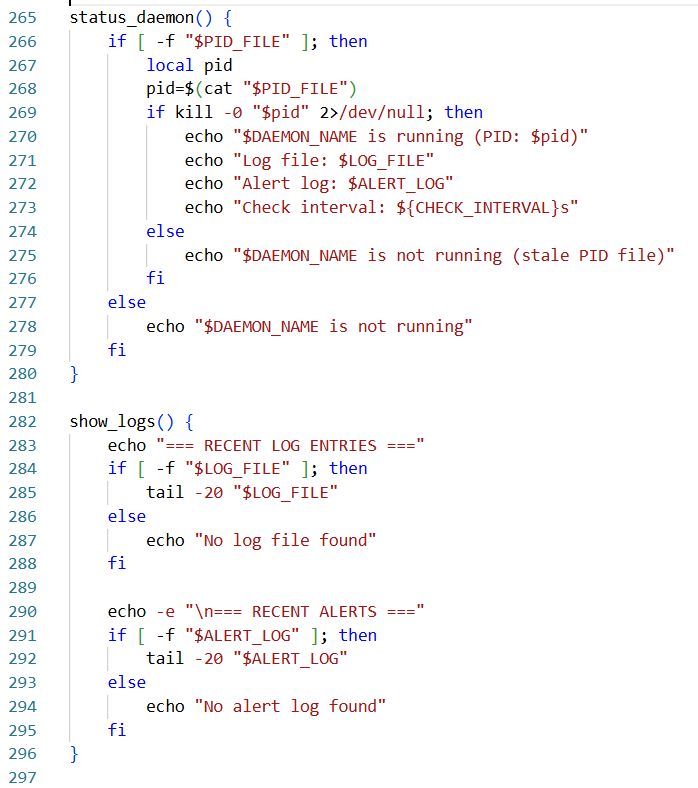


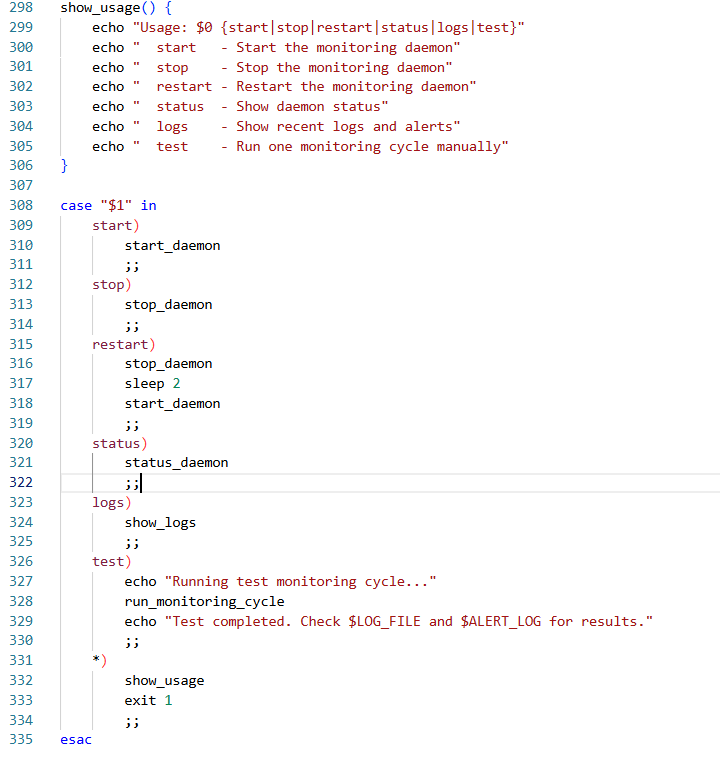












1. **Understanding the script**
2. **Configuration**

**DAEMON\_NAME**: Unique name of the monitoring daemon.

**PID\_FILE**: Stores the PID of the running daemon to avoid duplicates.

**LOG\_FILE**: Stores standard monitoring logs.

**ALERT\_LOG**: Stores alerts triggered by system issues.

**CHECK\_INTERVAL**: Time (in seconds) between each monitoring cycle.

1. **Thresholds**

These define the limits that will trigger alert messages.

**CPU\_ALERT\_THRESHOLD**=85 **MEMORY\_ALERT\_THRESHOLD**=90 **DISK\_ALERT\_THRESHOLD**=95 **LOAD\_ALERT\_THRESHOLD**=5.0

1. **Alerts**

**ALERT\_EMAIL**="admin@localhost"  
**ENABLE\_EMAIL\_ALERTS**=false  
**ENABLE\_SYSLOG\_ALERTS**=true

1. **Functions:**
2. ***Core functions***

### **log\_message()**

Logs a general message with timestamp to the main log.

### **log\_alert()**

* To alert log file.
* To syslog (if enabled).
* By email (if enabled and supported).
* Also forwards to general log.

1. ***Monitoring functions***

Each function performs a specific health check and logs alerts when thresholds are exceeded.

### **check\_cpu\_usage()**

* Checks CPU load using top.
* Logs alert if CPU usage > $CPU\_ALERT\_THRESHOLD.

### **check\_memory\_usage()**

* Checks memory consumption using free.
* Logs alert if memory > threshold.

### **check\_disk\_usage()**

* Uses df to inspect mounted filesystems.
* Alerts if any usage exceeds disk threshold.

### **check\_load\_average()**

* Extracts 1-minute load average from uptime.
* Compares it to load threshold.

### **check\_failed\_services()**

* Uses systemctl to detect failed services.
* Lists them in alert log.

### **check\_authentication\_failures()**

* Parses /var/log/auth.log for failed login attempts in last 5 minutes.
* Triggers alert if more than 5 failures.
* Logs suspicious IP addresses.

### **check\_zombie\_processes()**

* Searches for zombie (Z) or defunct processes.
* Logs alert with process details.

### **check\_swap\_usage()**

* Alerts if swap usage > 10%.
* Helps detect memory pressure.

### **check\_cpu\_temperature()**

* Requires sensors from lm-sensors.
* Warns if CPU temperature exceeds 75°C.

### **check\_disk\_io\_wait()**

* Uses iostat to detect I/O wait > 20%.
* Indicates disk bottlenecks.

### **check\_network\_anomalies()**

* Scans for unexpected open connections not matching common service ports.
* Alerts if > 10 suspicious connections.

1. **Log Management**

### **rotate\_logs()**

* Checks log sizes.
* Rotates (.old) and recreates logs if over 500 KB.
* Avoids uncontrolled log growth.

1. **Monitoring Lifecycle**

### **run\_monitoring\_cycle()**

* Executes all monitoring functions.
* Handles log rotation.
* Central routine for one complete system check.

1. **Daemon Control Functions**

### **start\_daemon()**

* Starts the daemon in background.
* Writes PID, creates log files, and loops monitoring every CHECK\_INTERVAL.

### **stop\_daemon()**

* Stops daemon if PID file exists and process is active.
* Cleans up PID file.

### **status\_daemon()**

* Reports whether daemon is running, with PID and log info.

### **show\_logs()**

* Displays last 20 lines of standard and alert logs.

### **show\_usage()**

* Displays correct usage syntax and available commands.

1. **The script – Monitoring users**



1. **Understanding the script**

### **1. Currently Logged-In Users**

* Uses the who command to list all users currently logged in.
* Shows info such as login name, terminal, login time, and source IP (if remote).

### **2. Last 20 User Sessions**

* Uses the last command to list the 20 most recent user logins/logouts.
* Helps track who accessed the system and when.

### **3. Last 20 Shell Commands from Current User**

* Displays the last 20 commands run by the current user from .bash\_history.
* May not show recent activity if shell session hasn’t exited (bash writes history on logout).

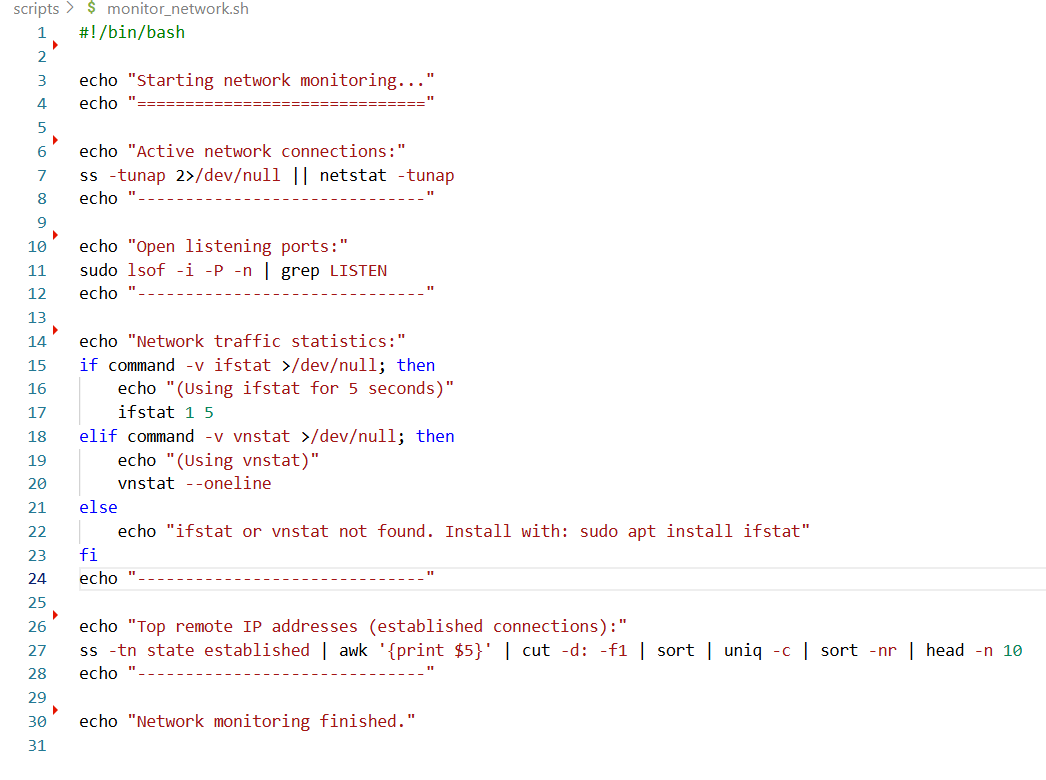
### **4. Recent Sudo Commands**

* Searches auth.log for lines indicating sudo usage.
* Displays the 10 most recent sudo commands executed.
* May require root privileges.

### **5. Failed Login Attempts (Last 20)**

* Scans auth.log for failed password login attempts.
* Shows most recent 20 failures to detect brute-force or unauthorized access attempts.

1. **The script – Monitoring network**



1. **Understanding the script**

### **1. Active Network Connections**

* Uses ss (preferred) or netstat to list active TCP and UDP connections.
* Options:
  + -t: TCP
  + -u: UDP
  + -n: Show numeric IPs/ports (no DNS resolution)
  + -a: All sockets
  + -p: Show associated process

**Fallback**: If ss is unavailable, falls back to netstat.

### **2. Open Listening Ports**

* Lists open ports where the system is accepting connections.
* lsof options:
  + -i: Network files
  + -P: Show port numbers (not service names)
  + -n: Show IPs (not hostnames)
* Filters for processes in a LISTEN state.

**Note**: Requires sudo for complete visibility.

### **3. Network Traffic Statistics**

* Shows bandwidth usage.
* Priority:
  + ifstat: Real-time stats every second for 5 seconds.
  + vnstat: Summarized statistics based on logged data.
* Prompts to install tools if neither is found.

**Helps** detect high outbound/inbound traffic.

### **4. Top Remote IP Addresses (Established Connections)**

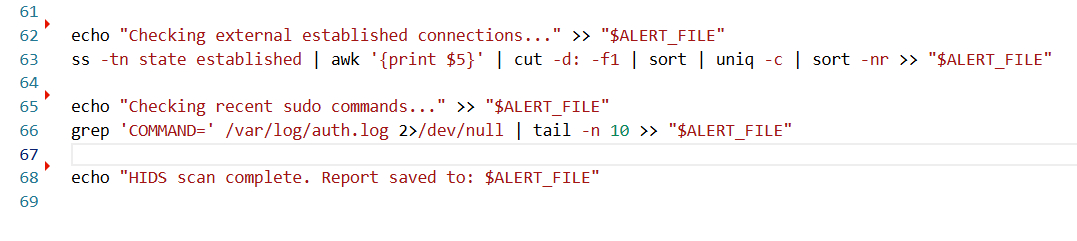
* Lists top remote IP addresses by number of established TCP connections.
* Workflow:
  + ss -tn state established: Get TCP connections in established state
  + awk '{print $5}': Extract remote IP and port
  + cut -d: -f1: Isolate IP
  + sort | uniq -c | sort -nr | head -n 10: Count and rank IPs

**Use Case**: Identify potential scanning or DDoS activity.

1. **The script – hids check**







1. **Understanding the script – hids check**

### **1. Initialize Log File**

Creates a unique timestamped log file in the alerts/ folder to store results.

### **2. System File Integrity Check**

Checks for unauthorized changes to critical system files using **SHA256 hash comparisons**.

#### **Behavior:**

* Stores hashes in hidden files (e.g., .passwd.hash) inside the scripts/ folder.
* On subsequent runs:
  + Compares current hash with stored one.
  + Logs a **warning** if the file hash has changed.

#### **Detects:**

* Malicious edits to /etc/passwd or /etc/shadow
* Tampering with /bin/bash

### **3. New User Accounts Detection**

Compares the current list of usernames with a previously saved list.

#### **Behavior:**

* Stores usernames in scripts/.user\_list.txt
* Uses comm to find **new users** that were not present in earlier runs.
* Logs any newly added users.

#### **Detects:**

* Unauthorized user account creation

### **4. New Open Ports Check**

Detects changes in listening network ports.

#### **Behavior:**

* Stores the list of open ports in scripts/.open\_ports.txt
* Compares with current snapshot using comm
* Logs any **new ports** found

#### **Detects:**

* New services listening on unexpected ports (backdoors, rogue servers)

### **5. External Established Connections**

Lists IP addresses with established connections to the host, sorted by frequency.

#### **Detects:**

* Unusual or persistent remote connections
* Potential exfiltration or command-and-control behavior

### **6. Recent Sudo Commands**

Extracts the last 10 sudo commands from auth.log.

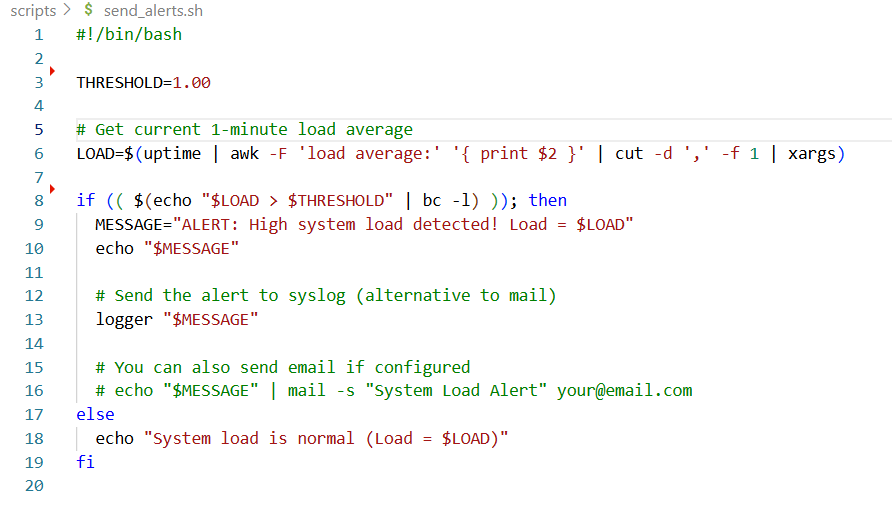
#### **Detects:**

* Suspicious or unauthorized privilege escalation attempts
* Unusual administrative actions

### **7. Completion Message**

Informs the user that the script has completed and where the report is stored.

1. **The script – Alerts sending**



1. **Understanding the script**

## **Load Retrieval**

* Uses the uptime command to extract the **1-minute load average**.
* awk, cut, and xargs are used to clean and extract the numeric value from the output.

## **Load Evaluation**

* Compares the current load to the threshold using bc (a calculator tool that supports floating-point arithmetic).
* Triggers alert **only if the current load is greater** than the defined threshold.

## **Alerting Mechanism**

If the system load is too high:

* Logs a **clear alert message** to:
  + **Console output** (for manual or real-time checking)
  + **System logs** via the logger command (stored in /var/log/syslog or equivalent)

### **Optional Email Alert**

* This line is commented out by default.
* If you have mail configured (e.g., via mailutils or sendmail), it can send email notifications for real-time alerts.

## **Normal Condition**

* If the system load is below the threshold, prints a confirmation message indicating everything is fine.

## **Dependencies**

* uptime: to retrieve load averages
* awk, cut, xargs: for parsing
* bc: for floating-point comparison
* logger: for logging to syslog
* (Optional) mail: for sending email alerts

1. **The script – setup cron**



1. **Understanding the script**

### **1. Script Directory Detection**

* Determines the **absolute path** of the directory where this script is located.
* This allows the script to construct **reliable paths** for sibling scripts and directories.

### **2. Relative Path Definitions**

* Points to:
  + The monitoring scripts located in the same directory.
  + logs/ and backup/ directories one level above (i.e., parent directory).

### **3. Directory Creation**

* Ensures that required directories exist.
* -p avoids errors if the directories already exist.

### **4. Crontab Backup**

* Creates a **timestamped backup** of the current crontab.
* This is a safeguard in case changes need to be reverted.

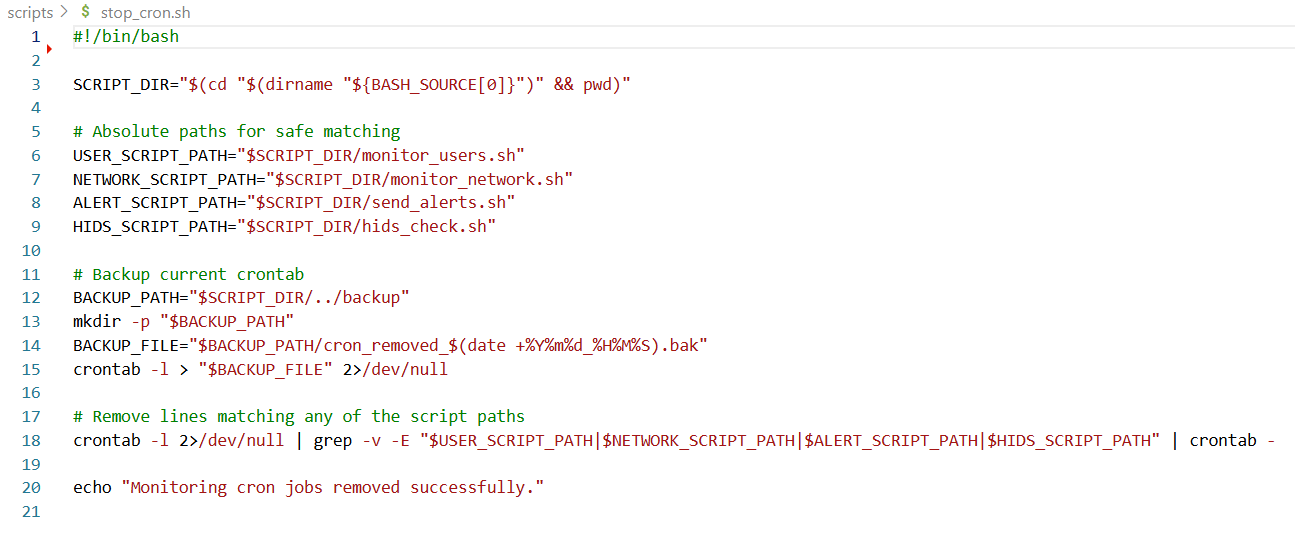
### **5. Scheduling the Cron Jobs**

* Appends new jobs to the existing crontab.
* Each job:
  + Runs **every 5 minutes**.
  + Executes the associated monitoring script.
  + Redirects both **stdout and stderr** to the appropriate .log file in the logs directory.

### **6. User Confirmation**

* Prints a message to confirm successful setup.

1. **The script – stop cron**



1. **Understanding the script**

### **1. Determine Script Location**

* Calculates the absolute path to the directory containing this script.
* Ensures consistent behavior regardless of how the script is called (relative or absolute path).

### **2. Define Paths to Monitoring Scripts**

* Constructs full absolute paths to the scripts that are scheduled via cron.
* Used to match and remove specific lines from the crontab.

### **3. Back Up Existing Crontab**

* Creates a backup/ directory (if it doesn’t already exist) one level above the script’s location.
* Saves a timestamped backup of the current crontab before making changes.
* Useful for restoring previous settings if needed.

### **4. Filter and Rewrite the Crontab**

* Reads the current crontab.
* Uses grep -v -E to **exclude any lines that contain paths to the monitoring scripts**.
* Feeds the filtered output back into crontab, effectively updating it by removing the matched entries.

### **5. Confirmation Message**

* Informs the user that the removal process has completed.