File Monitoring System Documentation

September 17, 2025

# Introduction

This document describes a File Monitoring System designed to track user access to the file

/home/kali/Desktop/project/company\_data/employee\_records/profiles/profiles.txt on a Kali Linux system. The system monitors file interactions via Command-Line Interface (CLI) and Graphical User Interface (GUI), logging events in the format Time: <timestamp> | User: <username> | Command: <command>. It uses the Linux Audit Framework (auditd) for event collection, AWK for log parsing, and a Bash script to automate processing. Logs are

stored in /home/kali/Desktop/project/company\_data/employee\_records/logs/main\_logs.txt.

## Objectives

* Monitor read, write, and attribute change events for profiles.txt.
* Capture timestamp, username, and command/process.
* Support CLI (e.g., cat, vim) and GUI (e.g., nautilus, gedit) interactions.
* Produce human-readable logs for analysis.

## Assumptions

* System: Kali Linux with auditd installed (sudo apt install auditd).
* Privileges: Root access for auditd configuration and log access.
* Configuration: The audit.conf file controls auditd behavior.
* Execution: The script runs manually or via cron.

# Directory Structure

The project directory is structured as follows:

employee\_records logs

logs.txt # Purpose unclear (possibly secondary log)

main\_logs.txt # Primary output log profiles

manage\_profiles.sh # Profile management script (not provided) profiles.txt # Monitored file

* **Base Path**: /home/kali/Desktop/project/company\_data/employee\_records/.

### Key Files:

* + profiles.txt: Monitored file.
  + main\_logs.txt: Stores formatted audit events.
  + manage\_profiles.sh: Likely for profile management (not provided).
  + logs.txt: Purpose unclear; possibly a backup log.

# Implementation Details

## Linux Audit Framework Setup

The Linux Audit Framework (auditd) captures system calls and file operations, logging to

/var/log/audit/audit.log. The provided .rules and .conf files configure monitoring and audit behavior.

### Audit Rules File

**File**: /etc/audit/rules.d/audit.rules (assumed standard location).

### Content:

-a always , exit -F path =/ home / kali/ Desktop / project/ company\_data / employee\_records / profiles/ profiles. txt -F perm = war -F auid >= 1000 -F auid != 4294967295 -k profile\_watches

1

### Explanation:

* + - * -a always,exit: Logs at system call exit.
      * -F path=...: Targets profiles.txt.
      * -F perm=war: Monitors write (w), attribute changes (a), read (r).
      * -F auid>=1000: Filters for non-system users (UID *≥* 1000).
      * -F auid!=4294967295: Excludes unset audit user IDs.
      * -k profile\_watches: Tags events for filtering with ausearch.

### Setup:

1. Place rule in /etc/audit/rules.d/audit.rules.
2. Reload: sudo augenrules –load.
3. Restart: sudo systemctl restart auditd.

### Audit Configuration File

**File**: /etc/audit/audit.conf (assumed).

### Content:

active = yes direction = out

path = / home / kali/ Desktop / manage\_log . sh type = always

format = string

1

2

3

4

5

### Explanation:

* active = yes: Enables the plugin.
* direction = out: Sends events to the script.
* path = ...: Points to manage\_log.sh (assumed to be the provided script).
* type = always: Processes all events.
* format = string: Passes events as strings.

**Note**: The .conf references /home/kali/Desktop/manage\_log.sh, but the script is likely in /home/kali/Desktop/project/company\_data/employee\_records/profiles/. This may require correction in the .conf file.

### Setup:

1. Place in /etc/audit/audit.conf.
2. Restart: sudo systemctl restart auditd.

## Bash Script

**File**: manage\_log.sh (assumed at /home/kali/Desktop/manage\_log.sh).

### Content:

#!/ bin / bash

sudo ausearch -k profile\_watches -i | awk ’ BEGIN { RS ="----"} {

if ( match ( $0 , / msg= audit \([0 -9 /:.]+/)) time = substr( $0 , RSTART +10 , RLENGTH -10)

if ( match ( $0 , / auid =[ a- z0 -9 \_ -]+/)) user = substr( $0 , RSTART +5 , RLENGTH -5)

if ( match ( $0 , / proctitle =[^[: space :]]+/)) { cmd = substr( $0 , RSTART +10 , RLENGTH -10)

rest = substr( $0 , RSTART + RLENGTH ) cmd = cmd rest

}

if( time !="" && user !="" && cmd !="")

print " Time : " time " | User: " time " | Command : " cmd

}’ | grep " ^ Time :" > / home / kali/ Desktop / project/ company\_data / employee\_records / logs/ main\_logs. txt

1

2

3

4

5

6

7

8

9

10

11

12

13

•**–**–

### Explanation:

* ausearch -k profile\_watches -i: Queries audit logs for events tagged profile\_watches, with interpreted (human-readable) output.
* awk: Parses logs, splitting records at ––.

**–** msg=audit(...): Extracts timestamp.

**–** auid=[a-z0-9*−*]+ : *AttemptstoextractaudituserID*(*note* : auid*isnumeric, sothismayfail*)*.* grep "*T ime* : ” : *Filtersvalidoutputlines.Outputwrittento*main\_logs.txt*.*

# Detailed Explanation of AWK

AWK is a text-processing language developed in 1977 by Aho, Weinberger, and Kernighan, optimized for pattern scanning and data extraction. It is well-suited for parsing audit.log due to its ability to handle structured records.

## Syntax and Structure

AWK programs consist of *pattern-action* pairs:

* pattern { action }
  + **Pattern**: A condition (e.g., regex, expression). If omitted, matches all records.
  + **Action**: Code in {...} (e.g., print).

### Blocks:

* + BEGIN { ... }: Runs before input.
  + Main body: Processes each record.
  + END { ... }: Runs after input.

Run via: awk ’program’ file or awk -f script.awk file.

## Key Concepts

* **Records and Fields**: Input splits into records (default RS = "") and fields (default FS

= " "). $0 is the full record; $1, $2, etc., are fields; NF is field count; NR is record number.

* **Output Separators**: OFS (field, default space), ORS (record, default newline).
* **Variables**: Scalars (e.g., count++) or associative arrays (e.g., users["uid"]).
* **Patterns**: Expressions (NR > 10), regex ($0 /error/), ranges (/start/, /end/).
* **Actions**: Arithmetic, control structures (if, for), functions.

## Relevant Built-in Variables

* $0: Full record (e.g., audit event).
* NF: Number of fields.
* FS: Input field separator.
* OFS: Output field separator.
* RS: Record separator (set to –– in script).

## Functions

* **Built-in**: match($0, /pat/, arr) (regex capture), substr($0, start, len) (substring).
* **User-Defined**: e.g., function extract\_time(line) { ... }.

## AWK in This System

The script sets RS="––" to split audit events. It uses match() to extract:

* Timestamp: msg=audit([0-9/:.]+).
* User: auid=[a-z0-9*−*] + (*intendedforusernames, but*auid*isnumeric*)*.Command* : The typo in the print statement causes the user field to display the timestamp.

# Usage Instructions

1•. **Setup Audit Rules**:

* Add rule to /etc/audit/rules.d/audit.rules.
* Reload: sudo augenrules –load.

### Configure Auditd:

* + Update /etc/audit/audit.conf.
  + Restart: sudo systemctl restart auditd.

1. **Run Script**: bash /home/kali/Desktop/manage\_log.sh.
2. **View Output**: cat main\_logs.txt.

# Sample Output

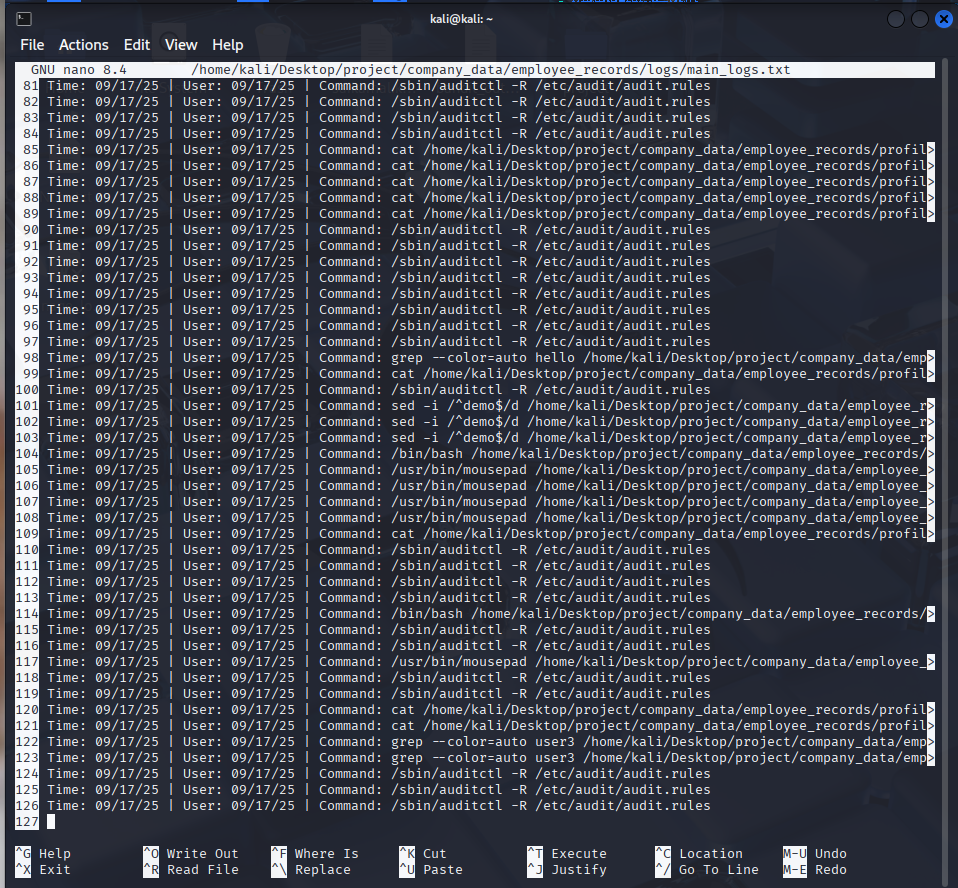
Time: 2025-09-17 17:25:45.123 | User: 2025-09-17 17:25:45.123 | Command: cat profiles.txt

Time: 2025-09-17 17:30:22.456 | User: 2025-09-17 17:30:22.456 | Command: nautilus

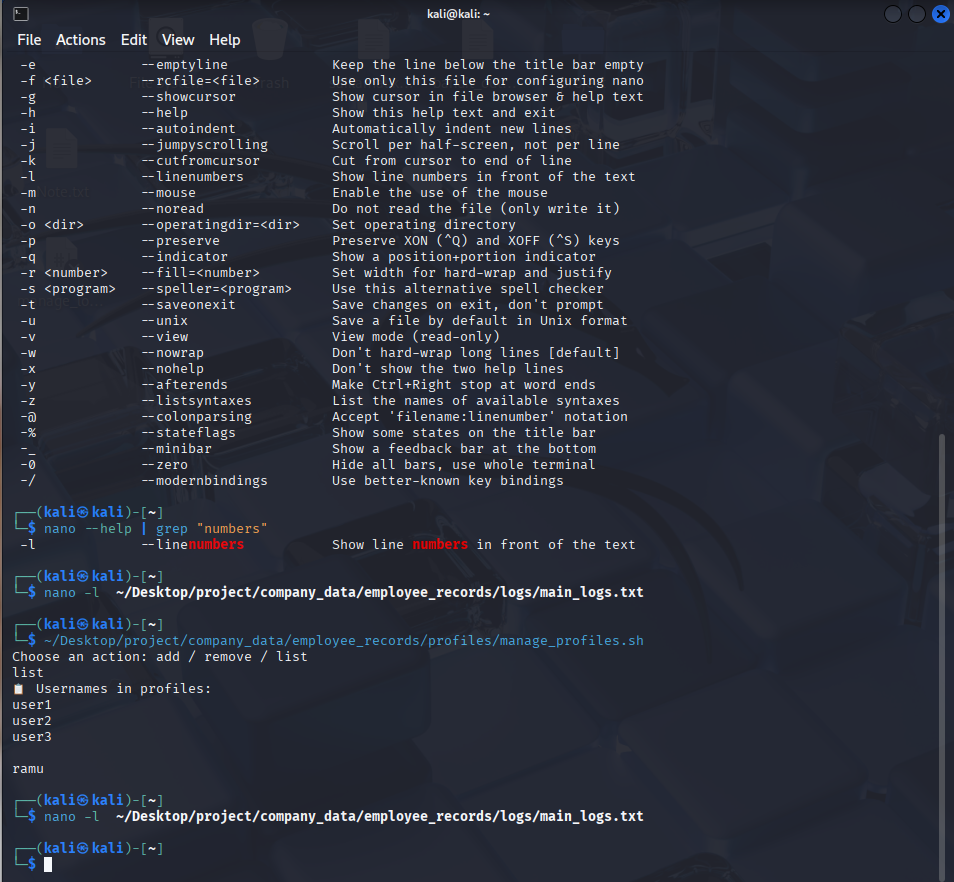
# Testing and Validation

* **Test CLI**: Run cat profiles.txt and check main\_logs.txt.

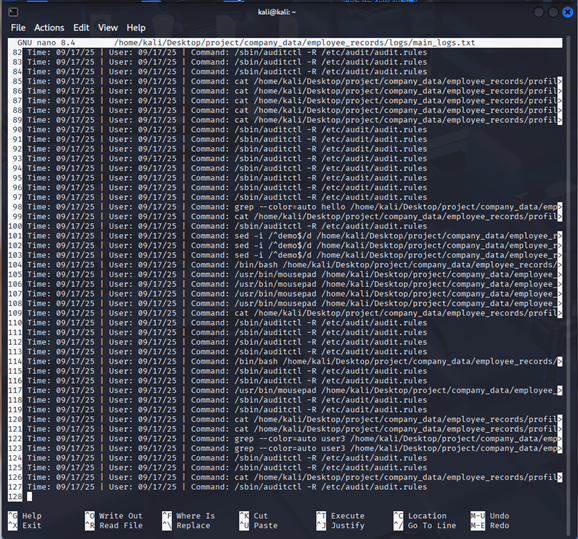
Before access file log file screenshot

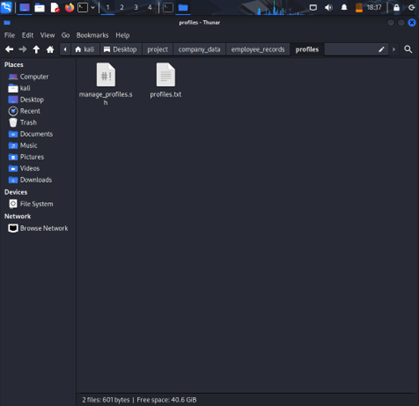


Accessing file using CLI



After accessing the file



* **Test GUI**: Open profiles.txt directly through GUI
* **Verify Logs**: Use sudo ausearch -k profile\_watches.

# Conclusion

The system monitors file access using auditd and AWK, supporting CLI and GUI events. AWK’s pattern-matching is effective for log parsing, despite minor issues in user parsing. Future work could address the user field typo and enable real-time monitoring.