Auditd Analysis script writing summary  
  
Software for saving in the SQL database data from audited software about file changes

Background:

I was given a task to write software as part of an exercise. The task was defined as the analysis of files created by the Auditd software in Ubuntu OS.

I did not know about Auditd before. Therefore, the first thing I went to learn about the system. I did an internet search to learn about her. This is a software that produces logs in Linux.

Because of the schedule, I decided to focus on existing implementations to use the software. I found the UK government's list of recommendations and an organized list on GitHub (Gov.uk audit rules). Later I saw that the questions in the twere about other topics and I decided to build the list of rules myself according to the results requested in the queries from the database.

In the first step, I thought that the software I wrote would check that Audited is installed, activated and define the required rules. After trying I realized that I don't have enough time for everything. Prioritizing this as a lower priority, I decided to install manually from the terminal and leave this task for the end. (I didn't have time to write this part of the code)

The list of rules that seemed relevant to me is in the audit\_rules.txt file.

The software will check the rules actually created by loggers according to the keys present there. Only some of them can be activated, but it is assumed that the key names have been written correctly.

user interface:

It has not been defined how to implement the user interface. I thought of two options, activation via command line parameters or via a selection list printed to the user. I chose the second option, but there is no significant difference between the options in the application.

If the software has been used by real users, you should add explanations, a request for help, etc. Due to the time limit I did not write these parts.

the plan:

The program assumes that Auditd exists and creates log files according to the defined rules.

The software checks that there is a database with the correct tables, if a table is missing it creates it. I used SQLite as recommended in the assignment.

When running, the software asks the user whether to update the database. This is an operation that takes time and I allowed it to be skipped. It makes sense to add a feature that will activate an automatic update once a day.

Finally, the software asks the user which query to execute and prints a result to the terminal. The list of options is taken directly from section H in the task.

The process of collecting the information and updating the database:

1. The system checks when the database changed, and which files changed after that (from the predefined folder).
2. For each relevant file, the system runs all the lines and checks whether the event exists in the database. From analyzing the log files and checking online, I realized that the unique identifier is a combination of time and serial number of the event. From the research I conducted in the vast majority of cases all the lines relevant to the event are recorded one after the other. The software uses this and collects all the rows into one event and saves it in the database.
3. For each detected event, it is checked if it exists in the database, if not it is added. (It should be noted that there are few cases where there is new information for an existing event. It is necessary to handle this but I did not have time. Currently the software skips lines for which there is an event in the database)
4. The software informs the user of the number of events added to the system.
5. Extracting the event ID and extracting the information stored in the database fields is done using Regex.

The database:

I used SQLite saved locally. You can change the save location in the attached .env file

.

The database contains four tables:

Logs: which contains the fields that are relevant to queries and the main information of the database

Rules: the list of rules and an explanation of each of them

Users: the list of users and additional details about them

Database logs: a log of the operation performed in the database is created in each session

N. B. Because of the time limit, I actually implemented the insertion of the information into the main table only.

Assumptions:

* Auditd is installed and running and creating log files.
* Set at least one rule from the list as required, otherwise the program is meaningless.
* The program and user have appropriate permissions to create a database.
* The Auditd log files are in a default location (see the .env file).
* I synchronized the times between the database and the Auditd log files according to UTC. I only checked on the computer and didn't have time to check this point in depth.
* The folder path that has the most activity in section H.5, referring to type: PATH (not exe)

Challenges:

* My time estimation was wrong in some cases, especially I didn't leave enough time for documentation and I didn't write it while writing the code.
* The business purpose of the software was not clear enough to me, so the initial research and design were more challenging.
* Writing a Regex expression, I realized that it takes longer than expected without ChatGPT, I looked for a replacement and used a generator I found on the Internet to create expressions.

What I would improve:

* I didn't write adequate documentation while writing the code and didn't leave time to write it.
* Exception handling in accessing the database and files can be improved by dividing them into different exception cases. For example, it is possible to divide between a bug in the query code, in accessing the file or in violation of SQL rules.
* There is a lack of completeness in several places in the code (mainly handling event updates, database logs, checking that audited exists and activating the rules)
* The audit\_rules.txt file is written with the intention of reading it through the script and then running the rules. I didn't have time to implement it in the end.