Table of Contents

1. Project Objectives

This project involves developing a Python tool specifically designed to analyze log files, with a focus on /var/log/auth.log. The tool's primary function is to extract, examine, and interpret log data to uncover valuable insights about system operations, security events, and potential anomalies. It identifies key information such as command usage, user authentication changes, and security alerts, providing a detailed analysis that enhances the understanding of system behavior and security posture. The project also requires proper documentation, including comments in the code, and submission of both the source code and proof of functionality in the prescribed format.

2. Project Deliverables

This Python script which is written to meet the project objective to process log entries from an authentication log file (typically found in Unix-like systems at /var/log/auth.log) to capture and report on various actions related to user commands, specifically focusing on command executions, user account modifications, and password changes, to meet the following requirements:

Deliverables (PART) 1:

Log Parse auth.log: Extract command usage.

1.1. Include the Timestamp.

1.2. Include the executing user.

1.3. Include the command.

Deliverables (PART) 2:

Log Parse auth.log: Monitor user authentication changes.

2.1. Print details of newly added users, including the Timestamp.

2.2. Print details of deleted users, including the Timestamp.

2.3. Print details of changing passwords, including the Timestamp.

2.4. Print details of when users used the su command.

2.5. Print details of users who used the sudo; include the command.

2.6. Print ALERT! If users failed to use the sudo command; include the command.

Here’s a detailed breakdown of the script's components and their purpose:

3. Imports and Log File Reading

```
import datetime

log_path = '/var/log/auth.log'
with open(log_path, 'rt') as file:
    data = file.readlines()
```

- The script imports the datetime module to handle timestamps.
- It opens the authentication log file in read-text mode, loading all lines into a list called data.

4. Processing Each Log Entry

```
for i in data:
    split_data = i.split()
```

The script iterates through each line of the log file (data), splitting each line into parts for further analysis.

5. Timestamp Formatting

```
dt = datetime.datetime.fromisoformat(split_data[0])
formatted_time = dt.strftime("%Y-%m-%d %H:%M:%S")
```

The script converts the first element of split_data (which is the timestamp) into a datetime object, then formats it as a readable string.

6. Capturing Command Executions

```
if "COMMAND" in split_data[-2]:
  print("Command execution was captured for user:" + ' ' + split_data[1] + ' using ' + split_data[-2]
  + ' ' + split_data[-1] + ' at timestamp: ' + formatted_time)
if "COMMAND" in split_data[-3]:
  print("Command execution was captured for user:" + ' ' + split_data[1] + ' using ' + split_data[-3]
  + ' ' + split_data[-2] + ' ' + split_data[-1] + ' at timestamp: ' + formatted_time)
```

The script checks for the presence of the word "COMMAND" in the second-to-last or third-to-last parts of the split line, indicating that a command was executed. It prints out details about the user who executed the command and the command itself along with the timestamp.

7. Tracking ‘su’ Command Usage

```
if "su:" in split_data:
  su_info = str(split_data[-3][6:])
  su_print = print('The user ' + su_info + ' ' + 'used "su" command at datetime:' + ' ' +
  formatted_time)
```

The script identifies when the ‘su’ (substitute user) command is used, extracting the username and timestamp to print this information.

8. Monitoring ‘sudo’ Command Usage

```
if "sudo:" in split_data:
  sudo_splitinfo1 = split_data[-2]
  sudo_splitinfo2 = split_data[-1]
  if "COMMAND" in sudo_splitinfo1:
    sudo_info = str(sudo_splitinfo1 + ' ' + sudo_splitinfo2)
    print('sudo was used by ' + split_data[3] + ' ' + 'for the following: ' + sudo_info)
```

The script checks for sudo command usage, extracting relevant details and printing them if the command "COMMAND" is found.

9. Alerting on ‘sudo’ Command Failures

```
if "sudoers" in split_data:
  sudo_alert1 = split_data[3]
  sudo_alert2 = split_data[-2]
  sudo_alert3 = split_data[-1]
  print('ALERT!! The user ' + sudo_alert1 + ' ' + 'failed to use the command sudo for the following:
  + sudo_alert2 + ' ' + sudo_alert3)
```

If a user fails to use sudo, indicated by "sudoers" in the line, it prints an alert detailing the failure.

10. User Account Management

```
#newly added user

#Checks if the line contains "sudo:" and then checks if the command executed using sudo is "useradd" or "adduser". If true, the script extracts the user who added
the new user and the command used.

    if "sudo:" in split_data:

      sudo_splitinfo1 = split_data[-2]

      sudo_splitinfo2 = split_data[-1]

      sudo_splitinfo3 = split_data[-3]

      sudo_splitinfo4 = split_data[1]

      if "useradd" in sudo_splitinfo1:

        useradd_info = str(sudo_splitinfo1 + ' ' + sudo_splitinfo2)

        print('New user was added by ' + split_data[3] + ' ' + 'for the following: ' + useradd_info + ' ' + 'at datetime:' + ' ' + formatted_time)

      if "adduser" in sudo_splitinfo3:

        adduser_info = str(sudo_splitinfo3 + ' ' + sudo_splitinfo1 + ' ' + sudo_splitinfo2)

        print('New user was added by ' + sudo_splitinfo4 + ' ' + 'for the following: ' + adduser_info + ' ' + 'at datetime:' + ' ' + formatted_time)

#deleted users

#Checks if the line contains "sudo:" and then checks if the command executed using sudo is "userdel". If true, the script extracts the user who deleted the
existing user and the command used.

    if "sudo:" in split_data:

      sudo_splitinfo1 = split_data[-2]

      sudo_splitinfo2 = split_data[-1]

      if "userdel" in sudo_splitinfo1:

        userdel_info = str(sudo_splitinfo1 + ' ' + sudo_splitinfo2)

        print('Existing user: ' + sudo_splitinfo2 + ' ' + 'was deleted by ' + split_data[3] + ' ' + 'under the following: ' + userdel_info + ' ' + 'at datetime:'
        + ' ' + formatted_time)
```

The script monitors user account changes:

- **Adding a User**: It looks for ‘useradd’ or ‘adduser’ commands and prints details about the user added.
- **Deleting a User**: It captures and reports when a user is deleted using the ‘userdel’ command.

11. Password Changes

```
sudo_splitinfo1 = split_data[-4]
sudo_splitinfo2 = split_data[-3]
chpsswd_info = str(sudo_splitinfo1 + ' ' + sudo_splitinfo2)
if "password changed" in chpsswd_info:
  print('User: ' + split_data[-1] + ' ' + 'changed password at datetime:' + ' ' + formatted_time)
```

Lastly, it checks for password change actions and logs them if found in the entries.

12. Script Usage and Performance

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python log_analyzer.py | sort
ALERT!! The user newuser002 failed to use the command sudo for the following: COMMAND=/usr/bin/cat auth.log
Command execution was captured for user: kali using COMMAND=/usr/bin/apt-get update at timestamp: 2024-10-06 02:35:39
Command execution was captured for user: kali using COMMAND=/usr/bin/apt-get update at timestamp: 2024-10-06 04:38:04
Command execution was captured for user: kali using COMMAND=/usr/bin/cat auth.log at timestamp: 2024-10-08 06:59:31
Command execution was captured for user: kali using COMMAND=/usr/bin/passwd newuser001 at timestamp: 2024-10-07 02:16:44
Command execution was captured for user: kali using COMMAND=/usr/bin/systemctl start ssh at timestamp: 2024-10-06 04:23:12
Command execution was captured for user: kali using COMMAND=/usr/sbin/adduser newuser002 at timestamp: 2024-10-08 06:57:16
Command execution was captured for user: kali using COMMAND=/usr/sbin/adduser -p Godzilla at timestamp: 2024-10-12 08:13:24
Command execution was captured for user: kali using COMMAND=/usr/sbin/service --status-all at timestamp: 2024-10-06 04:21:17
Command execution was captured for user: kali using COMMAND=/usr/sbin/useradd newuser001 at timestamp: 2024-10-07 02:16:18
Command execution was captured for user: kali using COMMAND=/usr/sbin/userdel newuser001 at timestamp: 2024-10-07 02:26:45
Existing user: newuser001 was deleted by kali under the following: COMMAND=/usr/sbin/userdel newuser001 at datetime: 2024-10-07 02:26:45
New user was added by kali for the following: COMMAND=/usr/sbin/adduser -p Godzilla at datetime: 2024-10-12 08:13:24
New user was added by kali for the following: COMMAND=/usr/sbin/useradd newuser001 at datetime: 2024-10-07 02:16:18
sudo was used by kali for the following: COMMAND=/usr/bin/apt-get update
sudo was used by kali for the following: COMMAND=/usr/bin/apt-get update
sudo was used by kali for the following: COMMAND=/usr/bin/passwd newuser001
sudo was used by kali for the following: COMMAND=/usr/sbin/adduser newuser002
sudo was used by kali for the following: COMMAND=/usr/sbin/service --status-all
sudo was used by kali for the following: COMMAND=/usr/sbin/useradd newuser001
sudo was used by kali for the following: COMMAND=/usr/sbin/userdel newuser001
sudo was used by newuser002 for the following: COMMAND=/usr/bin/cat auth.log
The user kali used "su" command at datetime: 2024-10-06 04:36:49
The user kali used "su" command at datetime: 2024-10-06 04:36:58
The user kali used "su" command at datetime: 2024-10-06 04:37:46
The user newuser002 used "su" command at datetime: 2024-10-08 06:59:44
User: newuser001 changed password at datetime: 2024-10-07 02:16:57
User: newuser002 changed password at datetime: 2024-10-08 06:57:25
```

Execution of the script as shown above indicated that the script is running error free. For the ease of reference, the bash command ‘grep sort’ was used when running the python script for the ease of sequential viewing. As per the project requirements, the objective of parsing for PART 1 and PART 2 of the project structure has been successfully delivered.

13. Possible Areas for Improvement

Overall, the script effectively extracts relevant information from authentication logs, providing valuable insights into system activity. However, the script can be refined, through enhancement and improvement in areas such as:

- ***Error handling:*** *add try-except blocks to handle potential log file errors or parsing issues.*
- ***Code organization:*** *consider separating logic into functions for better readability and maintainability.*
- ***Log file rotation:*** *consider handling log file rotation to avoid missing log entries.*

14. Conclusion

The script serves as a log parser for auditing user activity related to authentication and authorization actions on a Unix-like system. It specifically tracks command executions, the usage of su and sudo, alerts on failures, and logs user management activities. Which may be critical for system administrators to maintain security and monitor user actions on the system.

15. References

--NIL--