PYTHON FUNDAMENTALS

PROJECT: LOG ANALYZER

WILSON LAU WEI QIANG

9339 6960

CFC 190324

S16

JAMES LIM

INTRODUCTIONS

Log files(auth.log) plays a very crucial role in system management by offering importants informations about activities in systems. Auth.log is responsible for recording events related to authentications such as user logins, executed commands with privileges and user task. By effectively examining and interpreting all these logs, we can identify unauthorized attempts for access, track user behaviors and enchant system security. The main goal is to create a specialized python tool to carefully analyze log files/extracting and examine important information's. Moreover by automating the extractions, users can proactively identify threats and promptly respond to incidents/attacks and can continuously improve the system.

Wilson Lau S16
CFC 190324

METHODOLOGIES

‘def auth_log(file_path, option)’

- “file_path” – Represent the path to the log file.
- “option” – Represent the user choice to process.

‘with open(file_path,’r’)’

- “open” – function use to open a file following with file_path.
- “r” – It indicates the file is opened in READ MODE.

‘for line in file:’

- Reads the log file line by line using ‘for’ loop.

‘if option ==1 || if “useradd[“ in line || auth_add_user(line)’

- Function checks if log file contains “useradd[”.
- If condition met, it call function ‘auth_add_user’.

‘userdel[‘

- Check if logfile contain “userdel[“.

‘passwd[‘

- Check if logifile contains “passwd[“.

‘if “su: pam_unix: session open” in line:’

- Check if SUDO commands is used.

‘parts=line.split()’

- Split into list of strings where correspond to a segment of the log line.

‘timestamp = “”.join(parts[0:3])

- Joined the parts to form a complete timestamp strings.

‘user = line.split(‘user: name=’)[1].split()[0]’

- Split the line at user:name and isolate the username which is typically the first after the split.

‘user = line.split(‘user: name=’)[1].split()[0]’

DISCUSSIONS

```
# ========================================
# Reading of Log
# ========================================

def auth_log(file_path, option):
    with open(file_path, 'r') as file:
        for line in file:
            if option == 1:
                if "useradd[" in line:
                    auth_add_user(line)
            elif option == 2:
                if "userdel[" in line:
                    auth_delete_user(line)
            elif option == 3:
                if "passwd[" in line:
                    auth_change_password(line)
            elif option == 4:
                if "su[" in line or "pam_unix(su-l:session): session opened" in line:
                    auth_su_command(line)
            elif option == 5:
                if "sudo:" in line:
                    if "authentication failure" in line or "incorrect password attempts" in line:
                        auth_failed_sudo(line)
            elif option == 6:
                if "sudo:" in line:
                    if "authentication failure" in line or "incorrect password attempts" in line:
                        auth_failed_sudo(line)
                    else:
                        auth_sudo_command(line)
            elif option == 7:
                print(f"{GREEN}Exiting the script now .......")
                time.sleep(3)
                return
```

READING OF THE LOG FILE

Using ‘def’ (define) we can set reuseable command to performs a specific task, it also helps to organize code which easier to understand and maintain the python script. While setting a function name “Auth_log” as a variable follow by parameters “file_path” a variable which direct to the path of the log files set in the script and “option” which dictates the user input option.

‘with open(file_path, 'r') as file:’ is use to handle,open and read the log files from a specified variable ‘file_path’. By opening in read mode ‘r’, the log file will be opened for READ-ONLY and will not attempt to write into the file. Its safety that prevent accidental modification to the log file.

‘for line in lines:’ it read and process each other sequentially. Using for loop on each lines enable sequential of log entries and specific analysis of each event in the file.

‘useradd[‘ is present to check if the substring is within the line of the current log file and to check the presence of ‘useradd’ command has been excuted. The “[“ appears in the log file to signify the PID(process ID) of the command. If substring is found, function (auth_add_user) is called.

‘userdel[‘ is a command used to delete user accounts from system. It find if the line contains ‘userdel’ that indicates a deletion of user event.

‘passwd[’ a command used to change users passwords. It find if the lines contains any ‘passwd’ that indicates a event of password change.

‘su[‘ (Switch user) is a command that change the login user. To find any lines containing ‘SU’ or ‘session opened’ which appears in the log file.

‘sudo’ a command for superuser to execute/elevate a command. It find any lines containing ‘sudo’ or ‘incorrect password’ that indicates a fail attempt by user.

AUTH_ADD_USER

```
# ========================================
# Checking if any user has been added
# ========================================

def auth_add_user(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'user: name=' in line:
        user = line.split('user: name=')[1].split(',')[0]
        print(f"{NORM}User added: {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)
```

'parts=line.split()' -- It split the log lines into individual based on whitespace. It easy for extraction parts of the log entry.

'timestamp="".join(parts(o:3))' -- It combine the first components to third components of the lines split to form the time. Timestamp normally appearing at the beginning of the log file and consist of 3 parts(date & time).

```
"Jun 28 12:00:00 server useradd[1234]: user: name=wilson, uid=1001, gid=1001, home=/home/wilson, shell=/bin/bash"
```

'user=line.split('user:name')[1]/split(',')[0]' – is used to extract and isolate the username from logfile. By adding [1] is to extract everything after user:name(which is 0) . This further split at the coma(' , ') and lastly [0] is where the first element from the output after splitting as stated by above example.

AUTH_DELETE_USER

```
# ========================================
# Checking if any user has been deleted
# ========================================

def auth_delete_user(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'user ' in line:
        user = line.split('user ')[1].split()[0]
        print(f"{NORM}User deleted : {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)
```

```
"Jun 28 12:00:00 server userdel[1234]: user wilson deleted"
```

'timestamp="".join(parts(o:3))' -- It combine the first components to third components of the lines split to form the time. Timestamp normally appearing at the beginning of the log file and consist of 3 parts(date & time).

'user=line.split('user')[1].split()[0] – This split the log line at "user".[1] is the second element after the split user[0] while 'split()[0]' is the second split where the username is.

Wilson Lau S16
CFC 190324

AUTH_CHANGE_PASSWORD

```
# ========================================
# Checking if password has changed
# ========================================

def auth_change_password(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'passwd[' in line and 'password for ' in line:
        user = line.split('password for ')[1].split()[0]
        print(f"{NORM}Password changed for user : {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)
```

'timestamp="".join(parts(o:3))' -- It combine the first components to third components of the lines split to form the time. Timestamp normally appearing at the beginning of the log file and consist of 3 parts(date & time).

'if 'passwd[' in line and 'password for ' in line:' – a function call to check substring 'passwd' or 'password' is in the each line. Passwd usually appear in log entries to indicate the user account for which the password has changed.

'user=line.split('password for')[1].split()[0] – This extract the username and split at the line of 'password for'[0] and take the second part [1] which begins with the username and split()[0] again and take the first element[0].

```
Jun 28 12:34:56 myserver passwd[1234]: password for wilson changed by root
```

SU COMMANDS

```
# ========================================
# SU commands lines
# ========================================
def auth_su_command(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'session opened for user ' in line:
        user = line.split('session opened for user ')[1].split()[0]
        print(f"{NORM}SU command used by : {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)
    elif '(to ' in line:
        user = line.split('(to ')[1].split(')')[0]
        print(f"{NORM}SU command used to switch to user : {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)
```

'timestamp="".join(parts(o:3))' -- It combine the first components to third components of the lines split to form the time. Timestamp normally appearing at the beginning of the log file and consist of 3 parts(date & time).

'user = line.split('session opened for user ')[1].split()[0]' – It extract the username from the log file that indicate a seccion was open(connected) for a specific user'

- 'line.split('session open')' use the substring as the delimiter and split into two parts.
- '[1].split()[0]' – The second element '[1]' is selected after the substring 'session is open'. While .split()[0] future split using whitespace delimiter which extract the first element [0]

Wilson Lau S16
CFC 190324

SUDO COMMANDS & FAIL SUDO COMMANDS

```
# ========================================
# Sudo commands lines
# ========================================

def auth_sudo_command(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'sudo: ' in line and ' COMMAND=' in line:
        user = line.split('sudo: ')[1].split()[0]
        command = line.split('COMMAND=')[1]
        print(f"{NORM}Sudo command used by : {RED}{user} {NORM}at {GREEN}{timestamp} - {CYAN}Command: {command}")
        time.sleep(1)

def auth_failed_sudo(line):
    parts = line.split()
    timestamp = " ".join(parts[0:3])
    if 'sudo: ' in line and ('authentication failure' in line or 'incorrect password attempts' in line):
        user = line.split('sudo: ')[1].split()[0]
        print(f"{RED}ALERT!!!! {NORM}Failed sudo attempt by : {RED}{user} {NORM}at {CYAN}{timestamp}")
        time.sleep(1)

# ========================================
```

'timestamp=””.join(parts(o:3))' -- It combine the first components to third components of the lines split to form the time. Timestamp normally appearing at the beginning of the log file and consist of 3 parts(date & time).

'if 'sudo: ' in line and ' COMMAND=' in line:' – Function call if the line contain both 'sudo / COMMAND=' where its normally indicated in the log file.

'user=line.split('sudo:')[1].split()[0] – it extract the username and command from 'sudo' command log.

- 'line.split('sudo:') [1]– using sudo as delimiter and split into two parts where sudo[0] and the command will be after 'sudo' which will be the second element[1].
- .split()[0] – Using whitespace as delimiter and split into two parts where the first element[0] will be extracted out.

```
wilson : TTY=pts/0 ; PWD=/home/wilson ; USER=root ; COMMAND=/bin/ls
```

'if 'sudo: ' in line and ():' – This function check if the line contain 'sudo', 'authentication failure' or 'incorrect attempts'

'user=line.split('sudo:')[1].split()[0]' – it extract the username and command from 'sudo' command log.

- 'line.split('sudo:') [1]– using sudo as delimiter and split into two parts where sudo[0] and the command will be after 'sudo' which will be the second element[1].
- .split()[0] – Using whitespace as delimiter and split into two parts where the first element[0] will be extracted out.

CONCLUSION

Log analyzer script is important towards enchancing cybersecurity measurements by providing realtime visibility into critical user activities and system changes. Detailed tracking and reporting capabilities enable early detection of unwanted suspicious behaviors thus allow swift respond to potential attacks or threats it also effective parsing and support compliance and incident response efforts. It contributed to a deeper understanding of our security environment. Adaptation of the analyzer will be essential and important in maintaining a robust defends against evolving cyber threats.

REFERENCES

https://linux.die.net/
https://stackoverflow.com/
https://discuss.python.org/
https://medium.com/
https://github.com/h5py/h5py/issues/1220