LOG MONITORING WORKFLOW

Project 3

Fahad Shahzad
06/05/24

THE SCENARIO ---------- 2
THE WORKFLOW ---------- 2
THE PROGRAMMING---------- 2
---------- 3
THE EXPECTED OUTPUT---------- 4
---------- 4
---------- 5
THE DOCUMENTATION ---------- 5
THE UNUSUAL BEHAVIOUR---------- 5
THE POTENTIAL ITERATIONS ---------- 5
REFERENCES---------- 6

The Scenario

Turn a New Leaf is an NGO that supports youth. They have asked us, the Access Log Analyst to monitor their users. The users will only/must log into the system every Thursday to confirm or update employment.

The system the company uses are Windows, Linux along with two webservers. Based of these we are required to monitor activity to detect unusual traffic. Then to send it over an email documenting the issues discovered.

The Workflow

Since we are monitoring and reporting traffic related to failed log in of users, for the Windows machine we can do this by using the built-in application Event Viewer to capture the logs, set alerts to be triggered when certain criteria are met like failed log in, and then save logs. Since these logs are saved in the C: drive by default, we will change this to the desktop, we then then use the built-in task feature to go through the saved file and then email the application.

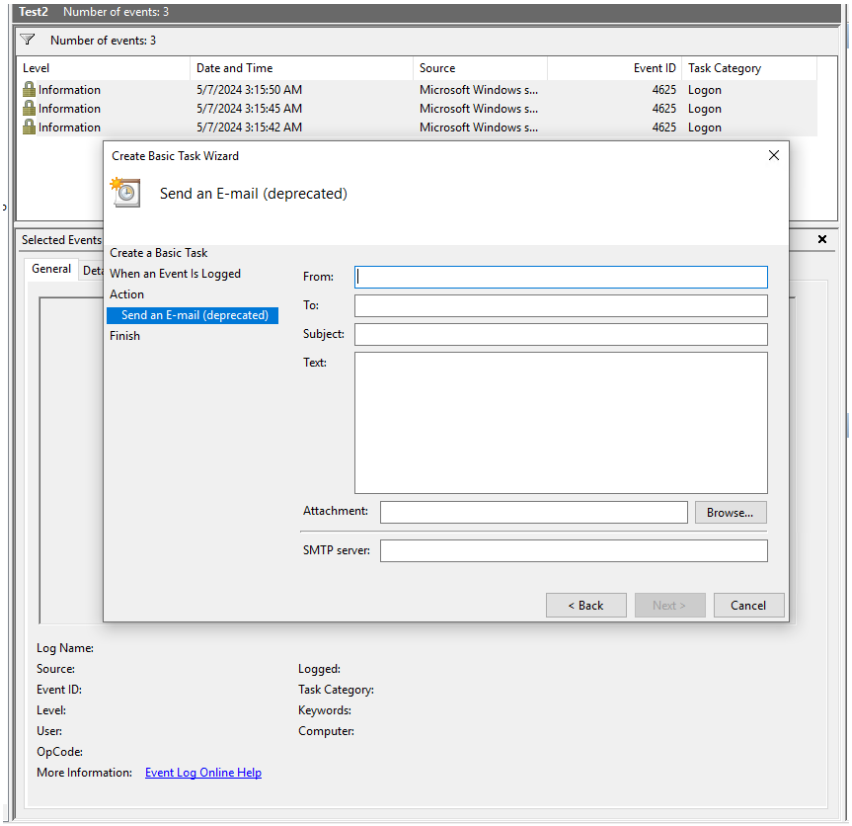
With the Linux machine, it will be slightly different since the OS is different. We will first have to ensure that the syslog is installed on the Linux machines, one confirmed and installed. All logs will automatically store to **auth.log** folder in Linux, we will then run bash script an hour before midnight, that will Regex through the **auth.log** folder, and grep the failed log in or unusual traffic and store it into a file that we will have a crontab set to midnight to compile and email the contents to the organization for reporting.

Similarly to the Linux machine, for the two webservers that we have, the logs are stored in the Apache or nginx folder respectfully. We will run a script that will use Regex to filter through the logs, and grep to extract all entries of failed log in attempts made and store them into a file. The document will have a crontab set to midnight and will compile and email the document to the organization.

The Programming

For the monitoring of the unusual activity related to login of users on every Thursday. The following tools will be used:

- Windows Event Viewer (to log events and create alerts)
    - Use Event Viewer To create a new view**,** then save the events in the view**, then a**ssign a task to email the view as shown below.



- Bash command in terminal using "#!/bin/bash" to allow the system to how to execute the file type ".sh"
- Terminal execution will use the following command to find the login failure.
    - sudo grep "authentication failure" /var/log/auth.log | head -3 >> /home/linux/Desktop/LogTestFile.txt
    - The "sudo" command allows us to use admin rights to access system files, we use the "grep" command to filter through the auth.log file to find the failed attempts.
    - The ">>" indicates that we are trying to move the filtered data into the given text file using the provided path.

```
#!/bin/bash

grep "authentication failure" /var/log/auth.log >> /home/kali/Desktop/logFiles1.txt

wc -l < /home/kali/Desktop/logFiles1.txt > /home/kali/Desktop/test1.txt
```

- Additionally, we will use the "wc -l" command to then count the number of entries made in the extracted log file and save it to a separate count file.
- Similarly, we would use sudo grep "error" /var/log/apache2/error_log | head -3 >> /home/linux/Desktop/LogTestFile.txt

- Crontab (to automate tasks)
  - [59 23 * * 4] crontab will be used to automate the execution of scripts and ready to then send both the log files for the webserver error logs and Linux authentication logs along with the count of each respective via email come the next morning.

The Expected Output

The scripts and command executed are expected to produce the following:

After getting the grep and store the data into a sperate file, the below is the expected result. This will show the manager everything they need to know from the time the error occurred to which user was causing the issue:

```
1 2024-05-07T21:08:56.746741-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
2 2024-05-07T21:09:01.302092-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
3 2024-05-07T21:09:05.143353-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
4 2024-05-07T21:48:23.471322-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
5 2024-05-07T21:48:27.392335-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
6 2024-05-07T21:48:30.119494-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
7 2024-05-07T21:48:33.099399-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
8 2024-05-07T21:48:36.141859-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
9 2024-05-07T21:49:00.820708-04:00 kali polkit-agent-helper-1[5454]: pam_unix(polkit-1:auth): authentication failur
  logname= uid=1000 euid=0 tty= ruser=kali rhost=  user=kali
10 2024-05-07T21:51:01.788770-04:00 kali polkit-agent-helper-1[5611]: pam_unix(polkit-1:auth): authentication failur
  logname= uid=1000 euid=0 tty= ruser=kali rhost=  user=kali
11 2024-05-07T21:51:08.582059-04:00 kali polkit-agent-helper-1[5615]: pam_unix(polkit-1:auth): authentication failur
  logname= uid=1000 euid=0 tty= ruser=kali rhost=  user=kali
12 2024-05-07T22:13:19.303516-04:00 kali lightdm: pam_unix(lightdm:auth): authentication failure; logname= uid=0 eui
  ruser= rhost=  user=kali
13 2024-05-07T22:20:02.809549-04:00 kali polkit-agent-helper-1[7572]: pam_unix(polkit-1:auth): authentication failur
  logname= uid=1000 euid=0 tty= ruser=kali rhost=  user=kali
```

After counting the above collected data, the below should be the result. This will allow the manager to get an understanding of the magnitude of the errors received and if they met certain threshold for them. It also allows them to get an idea of what happened without having to sit through all logs:

```
1 13
2
```

The Documentation

For the monitoring process, the logs for Windows, Linux and Webservers respectively will be running continually for as long as there is power and network to the machines and servers 24/7. This means that on days other than the required Thursday, if someone was to log in, that would raise a flag since no one should be doing so.

Furthermore, since the majority of login will occur on Thursday, the log filtering, processing, and complying will take place also on every Thursday. Hence the scripts will execute on Thursdays at midnight since there should not be any further login attempts past that point.

Once the relevant data is extracted from all logs and stored into one easy to follow document using scripts, an email will be generated inserting the compiled logs and sent to manager every Friday morning.

The Unusual Behaviour

What we will be monitoring is going to be unusual activity. Unusual behaviour will be either:

Bad login attempts

Authentication Failure

The Potential Iterations

An element of the workflow that can be improved is the automation of scripts and email creation. Since we know that a continuous task will take place with out a foreseeable end, we should be able to streamline the process as much as possible and be able to create scripts to do menial task such as email compiling and sending for monitor status reports. This can be done by getting a better understanding of Python and creating a template of what the email structure will be like in python as code. Research and experiments need to be conducted to achieve this.

References

Bash scripting cheatsheet. (n.d.). *Devhints.io Cheatsheets*. https://devhints.io/bash

Crontab.guru - The cron schedule expression generator. (n.d.). https://crontab.guru/

Dib, F. (n.d.). *regex101: build, test, and debug regex*. Regex101. https://regex101.com/