



SCOTT
MACLEAN

OCTOBER 2024

TURN A NEW LEAF - LOG ANALYSIS



TABLE OF CONTENTS

Executive Summary	3
Python Script	4
Bash Script	6
Crontab Configuration	8
Potential Iterations	9
Citations and References	10

EXECUTIVE SUMMARY

This report is for company Turn a New Leaf, a medium-sized non-profit, to implement an access log monitoring workflow to detect and respond to potential security incidents. This solution involves a Python script that scans the organization's Access_logs.txt file for signs of failed login attempts. If the number of failed logins exceeds a defined threshold within a 24-hour period, an email alert is automatically sent to the management team.

While the existing workflow provides a basic level of security monitoring, there are opportunities to enhance its capabilities and integrate it more seamlessly into Turn a New Leaf's broader security operations. Potential improvements include expanding the alerting mechanism to integrate with SIEM's, applying more advanced detection techniques to identify complex attack patterns, and developing robust reporting and visualization to provide deeper insights.

Additionally, expanding the monitoring scope to ingest logs from multiple sources, such as firewall logs, would give security teams a more comprehensive view of activity across the environment.



PYTHON SCRIPT

```

1  import re
2
3  import smtplib
4  from datetime import datetime, timedelta
5
6  # Set the log file path
7  log_file = 'Access_logs.txt'
8
9  # Define the email settings
10 smtp_server = 'smtp.gmail.com'
11 smtp_port = 587
12 smtp_username = 'your_email@example.com'
13 smtp_password = 'your_password'
14 alert_email = 'manager@turnanewhleaf.org'
15
16 # Define the monitoring parameters
17 failed_login_threshold = 10 # Alert if more than 10 failed logins in a day
18 monitoring_period = timedelta(days=1) # Monitor the last 24 hours
19
20 def monitor_access_logs():
21     """
22     Monitor the Access_logs.txt file for unusual activity and send email alerts.
23     """
24     # Read the log file
25     with open(log_file, 'r') as f:
26         logs = f.readlines()
27
28     # Filter for failed login attempts using regex
29     failed_logins = 0
30     for line in logs:
31         if re.search(r'\b(failed|failure|fail)\b', line, re.IGNORECASE):
32             failed_logins += 1
33
34     # Check if the failed login threshold is exceeded
35     if failed_logins > failed_login_threshold:
36         # Construct the email message
37         subject = 'Unusual Activity Detected - Turn a New Leaf'
38         body = f'Dear Manager,\n\nThe access logs show {failed_logins} potential failed login attempts in the last 24 hours, which exceeds the threshold of {failed_login_threshold}.\n\nPlease investigate this activity.\n\nBest regards,\n\nAccess Log Analyst'
39
40         # Send the email alert
41         send_email(subject, body)
42         print(f'Alert sent: {failed_logins} potential failed login attempts detected.')
43     else:
44         print('No unusual activity detected.')
45
46 def send_email(subject, body):
47     """
48     Send an email alert using the provided SMTP settings.
49     """
50     msg = f'Subject: {subject}\n\n{body}'
51     with smtplib.SMTP(smtp_server, smtp_port) as smtp:
52         smtp.starttls()
53         smtp.login(smtp_username, smtp_password)
54         smtp.sendmail(smtp_username, alert_email, msg)
55
56 if __name__ == '__main__':

```

PYTHON SCRIPT

CONTINUED

This script was developed to monitor the access logs for unusual activity, specifically failed login attempts. The main objects of this script are to:

1. Import libraries such as re for regex, smtplib for email, datetime for date and time.
2. Define the email address settings, to notify who should receive any alerts.
3. Define the monitoring parameters, such as a threshold of 10 failed login attempts in a day will trigger an alert.
4. Name the function of monitoring the access logs, named `monitor_access_logs`.
5. Read the file line by line and filter for failed login attempts looking for failed, failure, or fail keywords.
6. If the threshold has been exceeded, draft an email to the email address in the settings alerting.
7. If no thresholds have been exceeded, print "No unusual activity detected".
8. Providing the function for sending the email to the right address.

BASH SCRIPT

```

1  #!/bin/bash
2
3  # Set the log file path
4  LOG_FILE="Access_logs.txt"
5
6  # Define the email settings
7  SMTP_SERVER="smtp.gmail.com"
8  SMTP_PORT=587
9  SMTP_USERNAME="your_email@example.com"
10 SMTP_PASSWORD="your_password"
11 ALERT_EMAIL="manager@turnanewhleaf.org"
12
13 # Define the monitoring parameters
14 FAILED_LOGIN_THRESHOLD=10 # Alert if more than 10 failed logins in a day
15 MONITORING_PERIOD=86400 # Monitor the last 24 hours (in seconds)
16
17 monitor_access_logs() {
18     # Read the log file
19     readarray -t logs < "$LOG_FILE"
20
21     # Filter for failed login attempts
22     failed_logins=0
23     for line in "${logs[@]}; do
24         if [[ "$line" == *"failed login attempt"* ]]; then
25             ((failed_logins++))
26         fi
27     done
28
29     # Check if the failed login threshold is exceeded
30     if [ "$failed_logins" -gt "$FAILED_LOGIN_THRESHOLD" ]; then
31         # Construct the email message
32         subject="Unusual Activity Detected - Turn a New Leaf"
33         body="Dear Manager,\n\nThe access logs show $failed_logins failed login attempts in
the last 24 hours, which exceeds the threshold of $FAILED_LOGIN_THRESHOLD.\n\nPlease
investigate this activity.\n\nBest regards,\nAccess Log Analyst"
34
35         # Send the email alert
36         send_email "$subject" "$body"
37         echo "Alert sent: $failed_logins failed login attempts detected."
38     else
39         echo "No unusual activity detected."
40     fi
41 }
42
43 send_email() {
44     local subject="$1"
45     local body="$2"
46     local message="Subject: $subject\n\n$body"
47
48     # Use the 'mail' command to send the email
49     echo "$message" | /usr/bin/mail -s "$subject" -r "$SMTP_USERNAME" "$ALERT_EMAIL"
50 }
51
52 # Monitor the access logs
53 monitor_access_logs

```

BASH SCRIPT

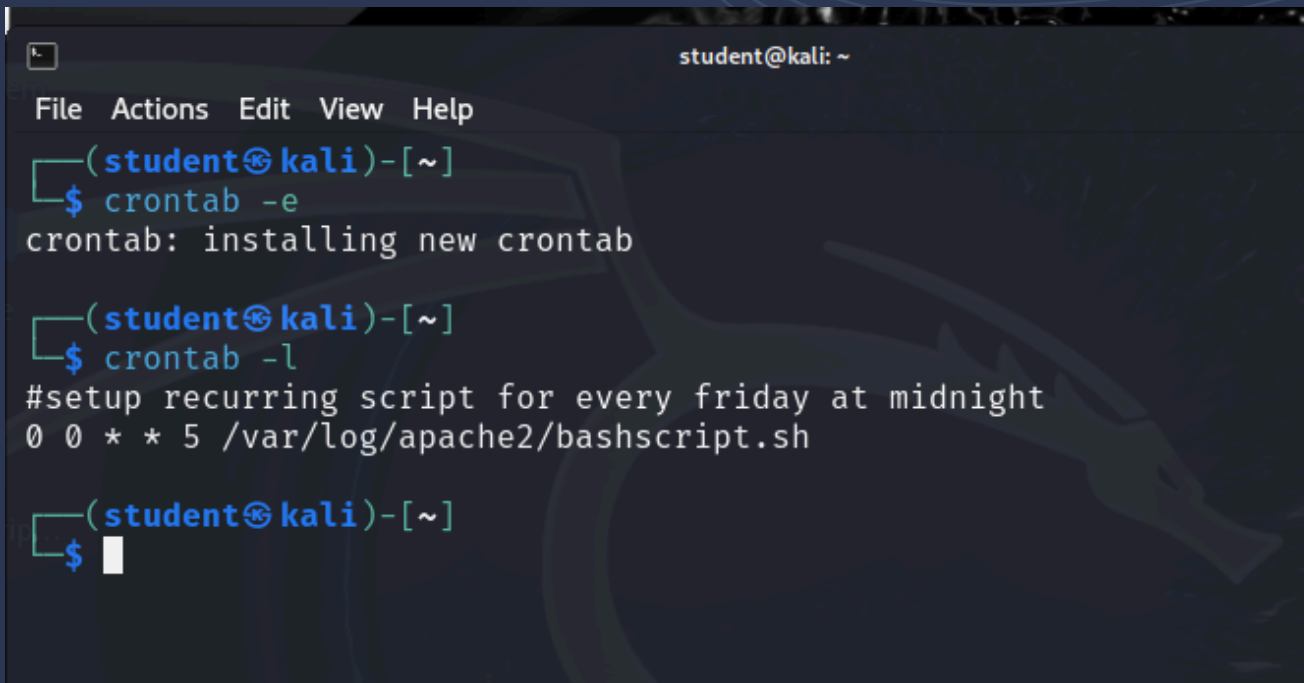
CONTINUED

The Bash script performs a similar function to the Python script, but using a shell-based approach. It monitors the `Access_logs.txt` file for signs of failed login attempts and sends an email alert if the number of detected failures exceeds a specified threshold.

Script Sections:

1. Sets the `log_file` as the `Access_logs.txt`
2. Defines the email address parameters
3. Defines the monitoring parameters with a threshold of also 10 failed login attempts in a day.
4. Reads the log files line by line and filters for "failed login attempt"
5. Checks if the failed login threshold has been exceeded and drafts an email template if it has.
6. Sends the alert email if it has been exceeded, otherwise prints no unusual activity detected.

CRONTAB CONFIGURATION

A screenshot of a terminal window titled 'student@kali: ~'. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows three commands being executed: 1. '\$ crontab -e' which results in 'crontab: installing new crontab'. 2. '\$ crontab -l' which lists the current crontab: '#setup recurring script for every friday at midnight' followed by the cron expression '0 0 * * 5 /var/log/apache2/bashscript.sh'. 3. A third prompt '\$' with a cursor, indicating the next step in the configuration process.

```
student@kali: ~  
File Actions Edit View Help  
(student@kali)-[~]  
$ crontab -e  
crontab: installing new crontab  
  
(student@kali)-[~]  
$ crontab -l  
#setup recurring script for every friday at midnight  
0 0 * * 5 /var/log/apache2/bashscript.sh  
  
(student@kali)-[~]  
$
```

Firstly we have to make sure either the python script or the bash script is in the same directory as the access_logs file. To do this we navigate to `cd /var/log/apache2`, use the command `ls` to list all the files and make sure that the scripts are in the right place. We also need to run the command `sudo chmod +x /var/log/apache2/bashscript.sh` to make sure that the script has executable permission. Fig 1 shows us the cron configuration for how to run the script every Friday at midnight. Friday at midnight was chosen as the employees have to update their employment status every Thursday.

POTENTIAL ITERATIONS

Improve Alerting Mechanism

One potential improvement would be to enhance the alerting mechanism beyond just email. Some ideas:

- Integrate with a ticketing system or incident management tool to automatically create tickets for unusual activity.
- Send alerts to a messaging platform (Slack, Teams, etc.) to improve visibility and on-call response.
- Implement tiered alerting based on severity - e.g. email for moderate issues, SMS/push notifications for critical ones.

Expand to Multi-Source Monitoring

The current solution focuses only on the Access_logs.txt file. Expanding to monitor multiple log sources could provide a more expanded view.

- Ingest logs from other systems like firewalls, web servers, authentication servers, etc.
- Correlate and aggregate data from across the environment to identify complex attack patterns.
- Leverage a log management or SIEM solution to centralize and analyze all security-relevant data.

CITATIONS AND REFERENCES

AND TOOLS USED

Compass. The Use of RegEx with Bash Scripts (Cyber Security Immersive). (n.d.).
<https://web.compass.lighthouselabs.ca/p/cyber/days/w03d4/activities/2929>

Compass. The Use of the Scripts with CRON (Cyber Security Immersive). (n.d.).
<https://web.compass.lighthouselabs.ca/p/cyber/days/w03d5/activities/2939>

Firewall. Firewall, Data Source DS0018 | MITRE ATT&CK®. (n.d.).
<https://attack.mitre.org/datasources/DS0018/>

TOOLS USED

Claude 3.5 Sonnet AI for Python and Bash Script development and refinement

