**Monitoring Indicators of Compromise @ Turn a New Leaf**

**Table of Contents**

1. Executive Summary
2. Introduction
3. Workflow Structure
   * Workflow
   * Programming
   * Expected Output
   * Documentation
   * Unusual Behaviour
   * Potential Iterations
4. Solutions Section
   * Workflow Description
   * Code with Comments
   * Sample Output
5. Conclusion
6. References

**Executive Summary**

This project aims to monitor and document unusual network traffic for Turn a New Leaf, a non-profit organization. The focus is on identifying and alerting for an unusual number of failed login attempts. The report outlines the workflow, scripts, expected outputs, and documentation process, utilizing both Bash and Python.

**Introduction**

**Objective**: To establish a workflow for monitoring unusual network traffic, specifically failed login attempts, and to alert the administrator when anomalies are detected.

**Company Profile**: Turn a New Leaf supports youth in rural communities to seek employment. Members log into the company system every Thursday to update their employment status and job search activities. The organization uses both Windows and Linux machines and has two web servers.

**Scope**: The project involves collecting, analyzing, and monitoring login attempt data from the web servers, identifying unusual patterns, and alerting the administrator.

**Workflow Structure**

**Workflow**

**Description**: This workflow outlines the steps to monitor and document logs for unusual traffic, focusing on failed login attempts. Monitoring will be done daily, with a detailed review every Thursday to align with the company’s member login schedule.

1. **Data Collection**: Capture network traffic and login attempts.
2. **Data Extraction**: Extract relevant information from logs.
3. **Pattern Matching**: Identify unusual patterns in the data.
4. **Alerting**: Send alerts for detected anomalies.
5. **Documentation**: Document the process and findings.
6. **Response**: Investigate and mitigate identified issues.

**Programming**

**Tools**:

* **Bash**: For automating log extraction and alerting.
* **Python**: For data extraction and pattern matching.
* **Regex**: For identifying patterns in log data.

**Key Commands and Scripts**:

**Data Collection**:

* sudo tcpdump -i eth0 -w traffic.pcap
* grep "Failed login" /var/log/auth.log > failed\_logins.log

**Data Extraction**:

* import re
* # Open the log file containing failed login attempts
* with open('failed\_logins.log', 'r') as file:
* logs = file.readlines()
* failed\_logins = []
* # Define a regex pattern to extract the timestamp and IP address from each log entry
* pattern = re.compile(r'(\w{3} \d{2} \d{2}:\d{2}:\d{2}) .\* Failed password for .\* from (\d+\.\d+\.\d+\.\d+)')
* for log in logs:
* match = pattern.search(log)
* if match:
* # Append the extracted timestamp and IP address to the failed\_logins list
* failed\_logins.append((match.group(1), match.group(2)))
* # Print the list of failed login attempts
* print("Failed Logins:", failed\_logins)

**Pattern Matching**:

* from collections import Counter
* # Count the number of failed login attempts from each IP address
* ip\_counter = Counter(ip for \_, ip in failed\_logins)
* # Define a threshold for what constitutes an unusual number of failed login attempts
* threshold = 5 # Adjust this value as needed
* # Identify IP addresses that exceed the threshold
* unusual\_ips = [ip for ip, count in ip\_counter.items() if count > threshold]
* # Print the list of unusual IP addresses
* print("Unusual IPs:", unusual\_ips)

**Alerting**:

* # Run the Python script to detect unusual IPs and store the result in a variable
* unusual\_ips=$(python3 detect\_unusual\_ips.py)
* # Check if there are any unusual IPs detected
* if [ ! -z "$unusual\_ips" ]; then
* # Send an email alert to the administrator with the list of unusual IPs
* echo "Unusual IPs detected: $unusual\_ips" | mail -s "Alert: Unusual Traffic" admin@example.com
* fi

**Expected Output**

* **Data Collection**: .pcap file and failed\_logins.log containing captured network traffic and failed login attempts.
* **Data Extraction**: List of source and destination IP addresses, and timestamps of failed logins.
* **Pattern Matching**: List of IP addresses with an unusually high number of failed login attempts.
* **Alerting**: Email alerts sent to the administrator for unusual traffic.

**Importance**: These outputs are crucial for identifying and responding to potential security threats, ensuring the network’s integrity and security.

**Documentation**

* **Process**: Document each step, including scripts and commands used.
* **Timing**: Daily monitoring with a detailed review every Thursday.
* **Sharing**: Use a shared document or internal wiki to update and share findings with your manager.

**Unusual Behaviour**

* **Flags**:
  + High number of failed login attempts from a single IP.
  + Unusual patterns in login times or sources.
  + Repeated access attempts to restricted areas.

**Potential Iterations**

* **Improvements**:
  + **Automation**: Enhance scripts to automate more steps, reducing manual intervention.
  + **Accuracy**: Improve regex patterns for better accuracy in identifying failed logins.
  + **Scalability**: Adapt scripts to handle larger datasets and more complex network environments.
* **Skill Development**:
  + **Advanced Scripting**: Learn more about advanced Bash and Python scripting techniques.
  + **Data Analysis**: Improve skills in data analysis and pattern recognition.
  + **Security Protocols**: Stay updated on the latest security protocols and best practices.

**Solutions Section**

**Workflow Description**

1. **Data Collection**:

**Bash Script**:

# Capture network traffic and save it to a file named traffic.pcap

sudo tcpdump -i eth0 -w traffic.pcap

# Extract failed login attempts from the authentication log and save them to failed\_logins.log

grep "Failed login" /var/log/auth.log > failed\_logins.log

* + **Expected Output**: A .pcap file and failed\_logins.log containing captured network traffic and failed login attempts.

1. **Data Extraction**: Extract relevant information from logs.

**Python Script**:

* + import re
  + # Open the log file containing failed login attempts
  + with open('failed\_logins.log', 'r') as file:
  + logs = file.readlines()
  + failed\_logins = []
  + # Define a regex pattern to extract the timestamp and IP address from each log entry
  + pattern = re.compile(r'(\w{3} \d{2} \d{2}:\d{2}:\d{2}) .\* Failed password for .\* from (\d+\.\d+\.\d+\.\d+)')
  + for log in logs:
  + match = pattern.search(log)
  + if match:
  + # Append the extracted timestamp and IP address to the failed\_logins list
  + failed\_logins.append((match.group(1), match.group(2)))
  + # Print the list of failed login attempts
  + print("Failed Logins:", failed\_logins)
  + **Expected Output**: List of tuples containing timestamps and IP addresses of failed login attempts.

1. **Pattern Matching**: Identify unusual patterns, such as a high number of failed logins from a single IP.

**Python Script**:

* + from collections import Counter
  + # Count the number of failed login attempts from each IP address
  + ip\_counter = Counter(ip for \_, ip in failed\_logins)
  + # Define a threshold for what constitutes an unusual number of failed login attempts
  + threshold = 5 # Adjust this value as needed
  + # Identify IP addresses that exceed the threshold
  + unusual\_ips = [ip for ip, count in ip\_counter.items() if count > threshold]
  + # Print the list of unusual IP addresses
  + print("Unusual IPs:", unusual\_ips)
  + **Expected Output**: List of IP addresses with an unusually high number of failed login attempts.

1. **Alerting**: Send alerts for unusual traffic.
   * **Bash Script**:
   * # Run the Python script to detect unusual IPs and store the result in a variable
   * unusual\_ips=$(python3 detect\_unusual\_ips.py)
   * # Check if there are any unusual IPs detected
   * if [ ! -z "$unusual\_ips" ]; then
   * # Send an email alert to the administrator with the list of unusual IPs
   * echo "Unusual IPs detected: $unusual\_ips" | mail -s "Alert: Unusual Traffic" admin@example.com
   * fi
   * **Expected Output**: Email alert sent to the administrator.

**Sample Output**

**Data Collection**:

* traffic.pcap
* failed\_logins.log

**Data Extraction**:

* Failed Logins: [('Sep 13 12:00:00', '192.168.1.10'), ('Sep 13 12:05:00', '192.168.1.11')]

**Pattern Matching**:

* Unusual IPs: ['192.168.1.10']

**Conclusion**

This project successfully establishes a workflow for monitoring unusual network traffic at Turn a New Leaf. By automating the process of data collection, extraction, pattern matching, and alerting, the organization can efficiently identify and respond to potential security threats. Continuous improvement and skill development are essential to enhance the effectiveness of this monitoring system.

**References & Citation**

Newham, C. (2005). *Learning the bash Shell* (3rd ed.). O’Reilly Media.

Garrels, M. (2008). *Bash Guide for Beginners*. Fultus Corporation.

Sweigart, A. (2015). *Automate the Boring Stuff with Python*. No Starch Press.

McKinney, W. (2017). *Python for Data Analysis* (2nd ed.). O’Reilly Media.

Chappell, L. (2017). *Wireshark Network Analysis* (3rd ed.). Chappell University.

Sanders, C. (2017). *Practical Packet Analysis* (3rd ed.). No Starch Press.

Friedl, J. E. F. (2006). *Mastering Regular Expressions* (3rd ed.). O’Reilly Media.

Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker’s Handbook* (2nd ed.). Wiley.

Stallings, W. (2013). *Network Security Essentials* (5th ed.). Pearson.