Monitor Security Logs for Failed Login Attempts

How It Works: The 5-Node Anomaly Detection Flow

This workflow efficiently processes logs to detect anomalies.

1. **Scheduled Check (Cron Node):** This is the primary trigger. It schedules the workflow to run at a defined interval (e.g., every 15 minutes), ensuring logs are routinely scanned for suspicious activity.
2. **Fetch Logs (HTTP Request Node):** This node is responsible for retrieving logs from an external source. It sends a request to your log API endpoint to get a batch of the most recent logs.
3. **Count Failed Logins (Code Node):** This is the core of the detection logic. The JavaScript code filters the logs for a specific event (`"login_failure"`), counts the total, and identifies unique IPs involved. This information is then passed to the next node.
4. **Failed Logins > Threshold? (If Node):** This node serves as the final filter. It checks if the number of failed logins exceeds a threshold you set (e.g., more than 5 attempts). If it does, the workflow is routed to the notification node; if not, the workflow ends safely.
5. **Send Anomaly Alert (Slack Node):** This node sends an alert to your team if an anomaly is detected. The Slack message includes a summary of the anomaly, such as the number of failed attempts and the IPs involved, enabling a swift response.

How to Set Up

Implementing this essential log anomaly detector in your n8n instance is quick and straightforward.

1. **Prepare Your Credentials & API:**
    - **Log API:** Make sure you have an API endpoint or a way to get logs from your system (e.g., a server, CMS, or application). The logs should be in JSON format, and you'll need any necessary API keys or tokens.
    - **Slack Credential:** Set up a **Slack credential** in n8n and get the **Channel ID** of your security alert channel (e.g., `#security-alerts`).
2. **Import the Workflow JSON:**
    - Create a new workflow in n8n and choose "Import from JSON."
    - Paste the JSON code (which was provided in a previous response).
3. **Configure the Nodes:**
    - **Scheduled Check (Cron):** Set the schedule according to your preference (e.g., every 15 minutes).
    - **Fetch Logs (HTTP Request):** Update the **URL** and **header/authentication** to match your specific log API endpoint.

- **Count Failed Logins (Code):** Verify that the JavaScript code matches your log's JSON format. You may need to adjust `log.event === 'login_failure'` if your log events use a different name.
- **Failed Logins > Threshold? (If):** Adjust the threshold value (e.g., `5`) based on your risk tolerance.
- **Send Anomaly Alert (Slack):** Select your **Slack credential** and enter the correct **Channel ID**.

4. **Test and Activate:**
   - **Manual Test:** Run the workflow manually to confirm it fetches logs and processes them correctly. You can temporarily lower the threshold to `0` to ensure the alert is triggered.
   - **Verify Output:** Check your Slack channel to confirm that alerts are formatted and sent correctly.
   - **Activate:** Once you're confident in its function, activate the workflow. n8n will now automatically monitor your logs on the schedule you set.