

Leonard M. Estos

Cybersecurity Researcher & Technical Author

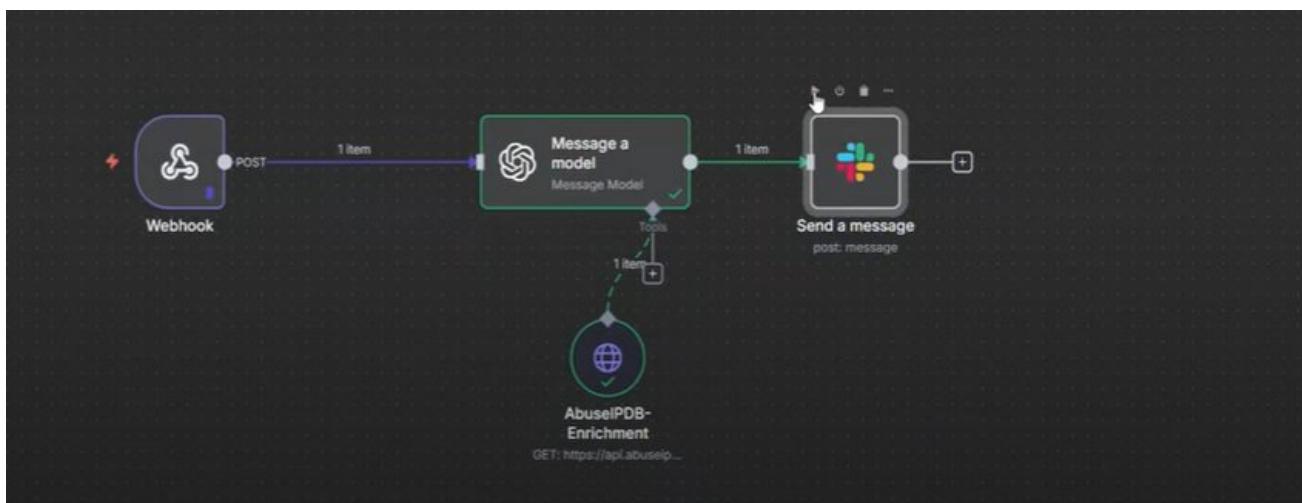
Edmonton, Alberta, Canada

Project Name: How To Use AI in Your SOC Workflow!

SOAR stands for **Security Orchestration, Automation, and Response**.

It is a platform and a set of technologies that helps **Security Operations Center (SOC)** teams manage the massive volume of security alerts they receive by combining tools, automating repetitive tasks, and standardizing incident response. The Primary goal is to Respond, contain, and remediate threats by automating workflows. If **SIEM** is the eyes and the brain of the SOC (Tells you what happened.), now **SOAR** is the hand and feet of the SOC (Tells you how to fix it.).

N8N Workflow is an **open-source**, low-code platform often described as a workflow automation tool or workflow engine. It allows you to connect different apps and services together to automate complex tasks, sync data, and build custom workflows without having to write extensive code.



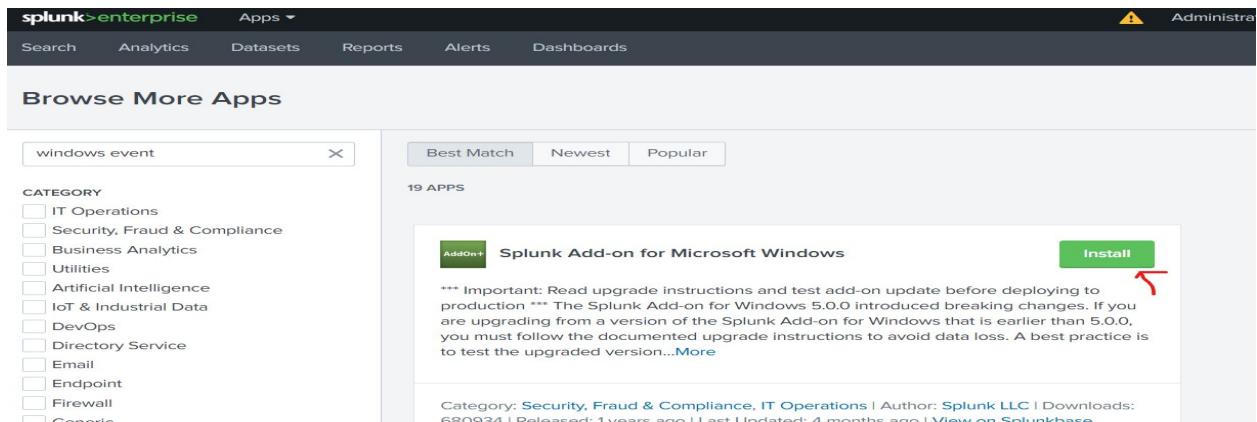
Requirement's:

1. Windows Server or Windows 10 VM
2. Splunk VM = SIEM
3. Kali Linux VM
4. N8N instance VM (Ubuntu)= SOAR Workflow
5. DFIR-IRIS VM = Open-Source Incident Response Platform

Splunk installation and configuration:

1. **Register** to Splunk <https://www.splunk.com> or https://www.splunk.com/en_us/download/splunk-enterprise.html
2. **Install** Splunk Server and Agent (Ubuntu or Windows). For me I installed this in my local device Windows 11.
<https://www.braindumps.com/blog/how-to-install-splunk-on-windows-and-linux-a-complete-step-by-step-guide/>
3. **Configure** your **Splunk Server Applications**
 - In Settings > Forwarding and Receiving > click Configure receiving > New Receiving Port (**9997**) > Click Save
 - Click Settings > Indexes > New Index > Name: **lizjames-project**
 - Click > Apps > Find more apps > search for Windows event and > **Install Splunk Add-on for Microsoft Windows** > sign-in using your Splunk account > Done

The purpose of this **Add-on** is to get the complete information from Windows Indexes.



- Configure your **input.conf** in **C:\Program Files\SplunkUniversalForwarder\etc\system\local**
- How to check > In your Splunk click > Click Search and reporting and type below

Failed login

```
index=lizjames-project source="WinEventLog:Security" EventCode="4625"
host=LizJamesDC1
```

Successful login

```
index=lizjames-project source="WinEventLog:Security" EventCode="4624"  
host=LizJamesDC1
```

4. Install Docker-Compose in Ubuntu machine.

- Type below
 - \$ **Docker-compose** = to check the command availability
 - \$ **Sudo apt install docker-compose**
- Now, if there is an error below like Failed to fetch

```
1.2.45.0~1ubuntu7.3 [1,100 kB]  
Get:17 http://ca.archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:  
2.43.0-1ubuntu7.3 [3,680 kB]  
Get:18 http://ca.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-f  
an all 0.12.16+24.04.1 [34.2 kB]  
Fetched 38.3 MB in 4s (9,425 kB/s)  
E: Failed to fetch http://ca.archive.ubuntu.com/ubuntu/pool/main/r/runc-app/runc  
_1.2.5-0ubuntu1%7e24.04.1_amd64.deb 404 Not Found [IP: 91.189.91.82 80]  
E: Failed to fetch http://ca.archive.ubuntu.com/ubuntu/pool/main/c/containerd-ap  
p/containerd_1.7.27-0ubuntu1%7e24.04.1_amd64.deb 404 Not Found [IP: 91.189.91.  
82 80]  
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis  
sing?  
root@lestos:/# ping google.com
```

- Run = # **sudo nano /etc/apt/sources.list.d/ubuntu.sources**
- Edit the URLs: <http://ca.archive.ubuntu.com/ubuntu> and remove the **ca**

```
GNU nano 7.2          /etc/apt/sources.list.d/ubuntu.sources  
Types: deb  
URIs: http://ca.archive.ubuntu.com/ubuntu/  
Suites: noble noble-updates noble-backports  
Components: main restricted universe multiverse  
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg  
  
Types: deb  
URIs: http://security.ubuntu.com/ubuntu/  
Suites: noble-security  
Components: main restricted universe multiverse  
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

```
GNU nano 7.2          /etc/apt/sources.list.d/ubuntu.sources  
Types: deb  
URIs: http://archive.ubuntu.com/ubuntu/  
Suites: noble noble-updates noble-backports  
Components: main restricted universe multiverse  
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg  
  
Types: deb  
URIs: http://security.ubuntu.com/ubuntu/  
Suites: noble-security  
Components: main restricted universe multiverse  
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

Run `$ sudo apt-get update` and `$ sudo apt-get upgrade -y` and type `$ docker` again. Now run the docker installation again `$ Sudo apt install docker.io` and `Sudo apt install docker-compose`

5. Install the N8N in Ubuntu machine. The `root` is `/lestos@lestos:` and file is `docker-compose.yaml` if you want to edit the IP.

```
$ mkdir n8n-compose
```

```
$ cd n8n-compose/
```

```
$ sudo nano docker-compose.yaml
```

 and provide the configuration below.

Services:

N8n:

```
  image: n8nio/n8n:latest
```

```
  restart: always
```

Ports:

```
  - 5678:5678
```

Environment:

```
  - N8N_HOST= 172.18.135.54 (your Ubuntu IP)
```

```
  - N8N_PORT=5678
```

```
  - N8N_PROTOCOL=http
```

```
  - N8N_SECURE_COOKIE=false
```

```
  - GENERIC_TIMEZONE=America/Toronto
```

Volumes:

```
  - ./n8n_data:/home/node/.n8n
```

```
GNU nano 7.2
services:
  n8n:
    image: n8nio/n8n:latest
    restart: always
    ports:
      - "5678:5678"
    environment:
      - N8N_HOST=172.18.135.54
      - N8N_PORT=5678
      - N8N_PROTOCOL=http
      - N8N_SECURE_COOKIE=false
      - GENERIC_TIMEZONE=America/Toronto
    volumes:
      - .n8n_data:/home/node/.n8n
```

Ctrl - X + Y to save the configuration

- Start to verify your configuration: `$ sudo nano docker-compose.yaml`

Run `$ apt install docker-compose`

Run `$ sudo docker-compose pull`

Note: If encountered an **error** run the **setup tools package**

Run `$ sudo apt-get install python3-distutils`

Run `$ apt install python3-pip`

Continue:

Run `$ sudo docker-compose pull`

Run `$ sudo docker-compose up -d`

- Now, try to access the URL type IP (Device):5678 e.g. **172.18.135.54:5678**

Note: If encountered an **error** run below

`$ sudo ufw status` = if result is inactive

`$ ll` = the issue is that `.n8n_data/` have no permission and permission is in root not at lestos.

```
root@lestos:/n8n-compose# sudo ufw status
Status: inactive
root@lestos:/n8n-compose# ll
total 16
drwxr-xr-x  3 root root 4096 Oct 10 14:11 .
drwxr-xr-x 24 root root 4096 Oct 10 13:09 ..
-rw-r--r--  1 root root  312 Oct 10 13:47 docker-compose.yaml
drwxr-xr-x  2 root root 4096 Oct 10 14:11 .n8n_data/
root@lestos:/n8n-compose#
```

We will now provide the permission

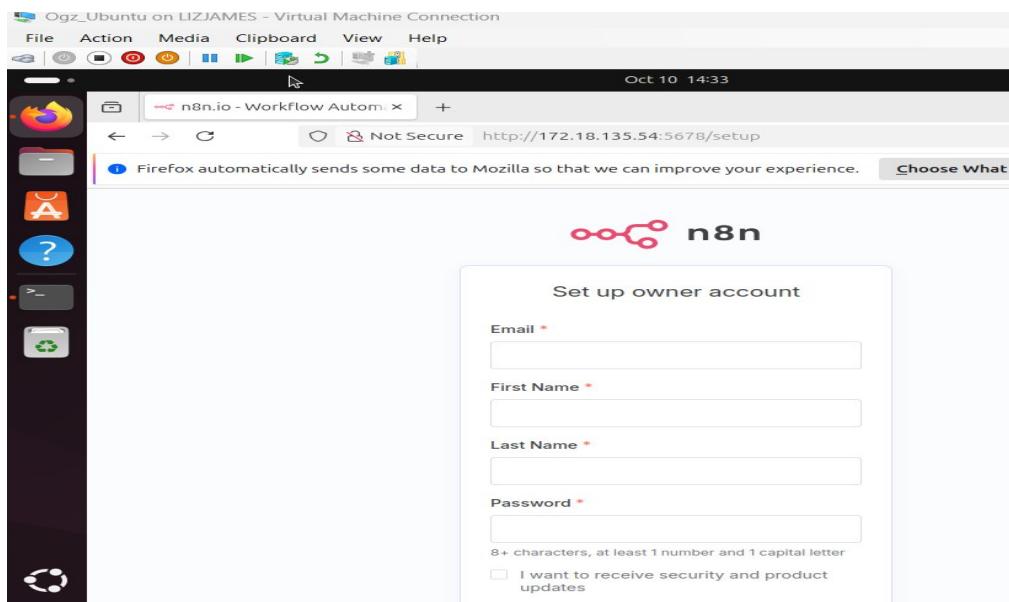
`$ sudo chown -R 1000:1000 .n8n_data/`

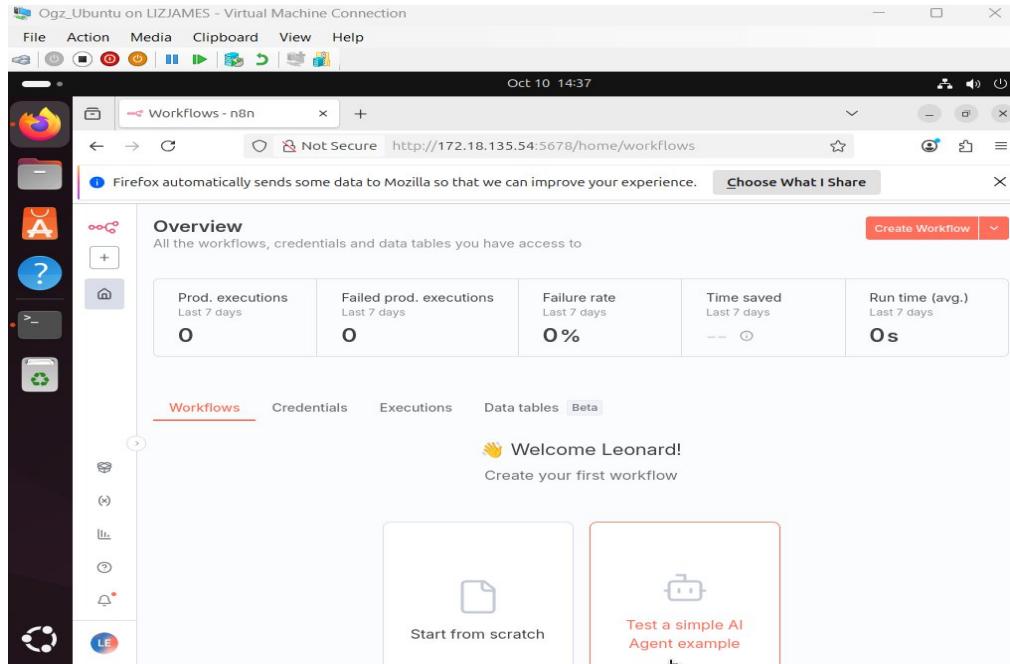
`$ ll` = to check that permission is now lestos

```
total 16
drwxr-xr-x 3 root  root  4096 Oct 10 14:11 .
drwxr-xr-x 24 root  root  4096 Oct 10 13:09 ..
-rw-r--r-- 1 root  root   312 Oct 10 13:47 docker-compose.yaml
drwxr-xr-x 2 lestos lestos 4096 Oct 10 14:11 .n8n_data/
root@lestos:/n8n-compose#
```

```
$ clear
$ sudo docker-compose down
$ sudo docker-compose up -d
```

6. The results, perfect! Now, continue to sign up





7. Now, go back to your SPLUNK and try to do a Search and Reporting

index=lizjames-project EventCode=4625

Time	Event
10/10/25 3:26:26 PM	LogName:Security EventCode=4625 EventType=0 ComputerName=LizJamesDC1.lizjames.com Show all 61 lines EventCode = 4625 eventtype = endpoint_services_processes eventtype = windows_event_signature track_event_signatures eventtype = windows_logon_failure authentication eventtype = windows_security_authentication authentication eventtype = windows_ta_data eventtype = wineventlog_security os windows eventtype = winedevlog security host = LizJamesDC1 source = WinEventLogSecurity sourcetype = WinEventLog

8. Now, we will create an alert for every **FAILED login**. Provide the query below.

`index=lizjames-project EventCode=4625 | stats count by`

`_time,ComputerName,user,src_ip`

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** `index=lizjames-project EventCode=4625 | stats count by _time,ComputerName,user,src_ip`
- Results:** 4 events found between 10/10/25 12:00:00.000 AM and 10/10/25 4:25:07.000 PM.
- Table View:** The results are presented in a table with columns: `_time`, `ComputerName`, `user`, `src_ip`, and `count`.
- Data:**

_time	ComputerName	user	src_ip	count
2025-10-10 14:45:10	LizJamesDC1.lizjames.com	administrator	0.0.0.0	1
2025-10-10 15:08:36	LizJamesDC1.lizjames.com	administrator	127.0.0.1	1
2025-10-10 15:08:40	LizJamesDC1.lizjames.com	administrator	127.0.0.1	1
2025-10-10 15:26:26	LizJamesDC1.lizjames.com	administrator	127.0.0.1	1

Click Save As > Alert > Title: > Run on Cron Schedule > Last 24 hours > Change all Cron Expression to ***** > Click Trigger Action > Add Actions > Click Add to Triggered Alerts > Click Add Actions again > Webhook > Now, for the URL we will use our N8N URL

9. Now, we will start with our N8N configuration.

- Click start from Scratch > Add first step > search for Webhook
- Change the **HTTP Method** for GET to **POST**
- Note look the URL below it will change from **GET to POST**
- Now, **get the URL and paste this to SPLUNK Webhook Triggered Alert**

The screenshot shows the N8N Webhook configuration screen with the following settings:

- Parameters:** Webhook URLs
- Test URL:** http://172.18.135.54:5678/webhook-test/201b972e-7b2d-4bf1-8824-21601db7bf16
- HTTP Method:** POST (highlighted with a red arrow)
- Path:** 201b972e-7b2d-4bf1-8824-21601db7bf16
- Authentication:** None
- Respond:** Immediately

- Result in SPLUNK

Leo-Brute-Force-Test

Enabled: Yes. [Disable](#)

App: search

Permissions: Private. Owned by lestos. [Edit](#)

Modified: Oct 10, 2025 4:48:35 PM

Alert Type: Scheduled. Cron Schedule. [Edit](#)

Trigger Condition: ... Number of Results is > 0. [Edit](#)

Actions: 2 Actions [Edit](#)

Add to Triggered Alerts

Webhook

Trigger History

20 per page ▾

	TriggerTime	Actions
1	2025-10-10 16:49:04 Mountain Daylight Time	View Results

Make sure that you will disable the alert the keep as triggered every minute after your testing. Go back to SPLUNK and click Disabled. > click Settings > All configurations > Find your Alert or type in search Leo and in Status click disabled or enable.

- Now, go back to your N8N > click Listen for test event

Results:

Pull in events from Webhook

[Listen for test event](#)

Once you've finished building your workflow, run it without having to click this button by using the production webhook URL. [More info](#)

[Back to canvas](#)

Webhook

[A Listen for test event](#)

Parameters Settings Docs

Webhook URLs

Test URL Production URL

POST http://172.18.135.54:5678/webhook-test/20fb972e-7b2d-4bf1-8824-21601db7bf16

HTTP Method POST

Path 20fb972e-7b2d-4bf1-8824-21601db7bf16

Authentication None

Respond Immediately

If you are sending back a response, add a "Content-Type" response header with the appropriate value to avoid unexpected behavior

Options No properties

OUTPUT

1 item

headers	params	query	body
accept-encoding : identity content-length : 411 host : 172.18.135.54:5678 content-type : application/json user-agent : Splunk/FC3AA9F-CEB1-46EE-B11A-4A1BB6D56F1A connection : close	(empty object)	(empty object)	<p>sid : scheduler_lestos_search_RMD5e18d47c7bb0a9dcf_at_1760136720_6</p> <p>search_name : Leo-Brute-Force-Test</p> <p>app : search</p> <p>owner : lestos</p> <p>results_link : http://172.18.135.54:5678/webhook-test/20fb972e-7b2d-4bf1-8824-21601db7bf16</p> <p>result : 1760129100</p> <p>ComputerName : LizJamesDC1.lizjames.com</p> <p>user : administrator</p> <p>src_ip : 0.0.0.0</p> <p>count : 1</p>

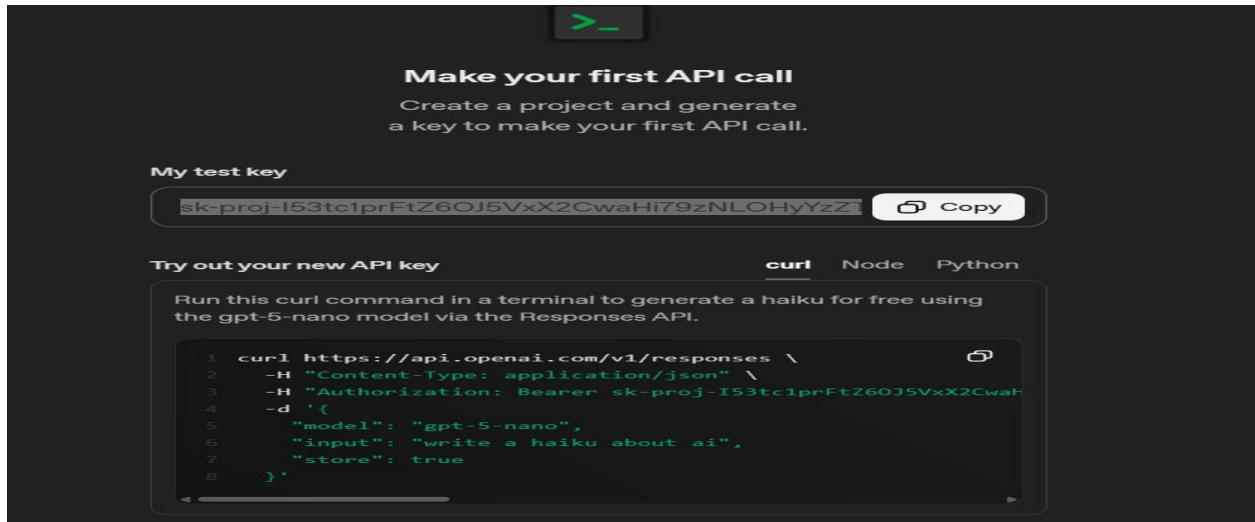
You can pin this output instead of waiting for a test event.

10. Building Automation Workflow:

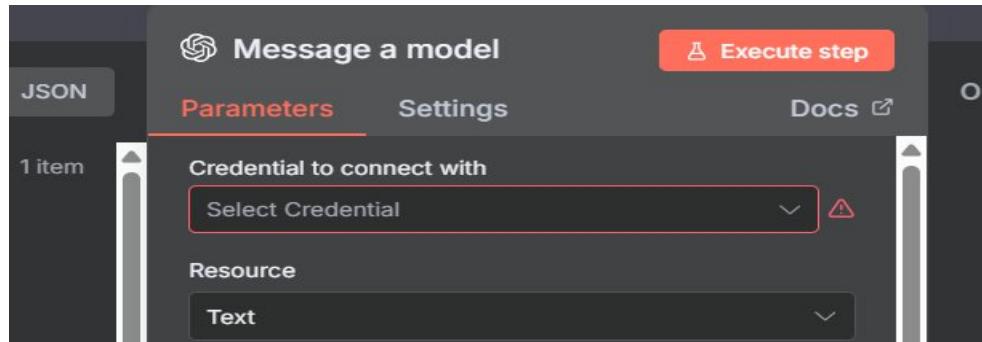
We will create an Automation inside the N8N, note that we will need an API credit for an OpenAI to make this work.

- Now, go back to your N8N and click > Back to Canvas > right click the Webhook and click PIN
- In the top icon + click and search chatgpt > click OpenAI > Click Message a model > Now, it will need a new credential a need an API key
- Create an account to OpenAI (Chat GPT) > Go to <https://openai.com/> > click Login > API Platform > Create your account
- Click START Building > Organization Name: S2S SOC Project > Provide the email you will invite

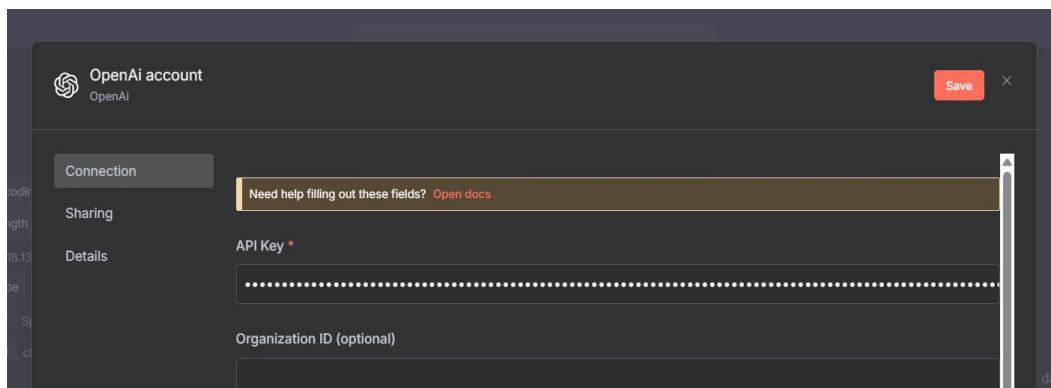
- API Name: S2S SOC Project > Project name: Default project > Click generate API



- Now, copy your OpenAI API key and go back to your N8N > **Click Credential to connect with** > click create a new credential



- Paste your OpenAI API key and click Save



- However, we need to purchase some credits. In your OpenAI > **click settings and click Billing** > Add payment details \$5 is enough for this project.

- Now, once purchase completed **go back to your N8N** automation and configure the Message a model below > click **Model: GPT-4-1-Mini** > **Role: Assistant** > **Prompt:**

Act as a Tier 1 SOC Analyst assistant. When provided with a security alert or incident details (Including indicators of compromise, logs, or metadata), perform the following steps:

Summarize the alert - Provide a clear summary of what triggered the alert, which systems/users are affected, and the nature of the activity (e.g., suspicious login, malware detection, lateral movement).

Enrich with threat intelligence - Correlate any IOC's (IP addresses, domains, hashes) with known threat intel sources. Highlight if the indicators are associated with known malware or threat actors.

Assess severity - Based on MITRE ATT&CK mapping, identify tactics/technique, and provide an initial severity rating (Low, Medium, High, Critical).

Recommend next actions - Suggest investigation steps and potential containment actions.

Role: Assistant

- Add another MESSAGE:**

PROMPT:

Format output clearly - Return findings in a structured format (Summary, IOC Enrichment, Serverity Assesment, Reccomendation Actions).

ROLE: System

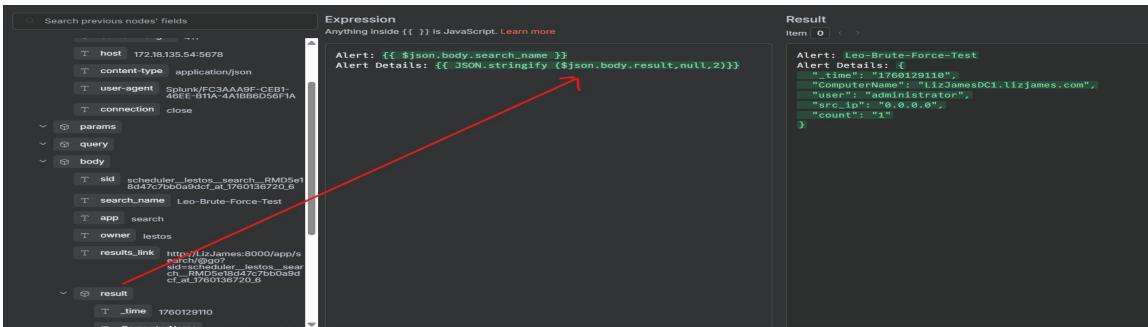
- Add another MESSAGE:**

PROMPT:

Alert: Drag the **search_name** to the prompt, see sample output below.

Alert Details: `Alert: {{ $json.body.search_name }}`

Alert Details: `Alert Details: {{ JSON.stringify($json.body.result,null,2) }}`



```

Search previous nodes fields
host 172.18.135.54:5678
content-type application/json
user-agent Mozilla/5.0(Windows NT 10.0; Win64; x64) AppleWebKit/537.36(KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
connection close
params
query
body
  sid scheduler_lestos_search_RMDSel1
  8d47c7bb0a9dcf_at_1760136720_6
  search_name Leo-Brute-Force-Test
  app search
  owner lestos
  results_link http://liz.james:8000/applications/leostest@go7test
  result
    time 1760129110
    ...
Result
Item: 0 < >
Alert: Leo-Brute-Force-Test
Alert Details: {
  "time": "2022-07-06T09:51:10Z",
  "ConsumerName": "LizJamesDC1.lizjames.com",
  "User": "administrator",
  "src_ip": "0.0.0.0",
  "count": 1
}

```

What we did in alert details: `{{JSON.stringify > (drag the result,null,2)}}`

ROLE: User

11. Connect our workflow to SLACK.

- Create an account to slack
- Click Create Workgroup > Company name S2S > Provide your name > Provide your picture > Provide email invite > click Start with the **Limited Free Version**
- Right click Channels > New Channel > Next > Channel name: **alerts** > Click Create

12. Now, go back to your **N8N**.

- in the right-hand side click > + > In the search type **Slack** > choose **Slack** > Go down and find the Message Action and choose Send a message
- In the Send a message > credential to connect with > Create a new credential
- Now, we don't know how to do it > Click Open Docs [Slack credentials | n8n Docs](#) > find the Using API access token

Using API access token

1. Open your **Slack API Apps** = <https://api.slack.com/apps> page.
2. Select **Create New App > From scratch**.
3. Enter an **App Name**.
4. Select the **Workspace** where you'll be developing your app.
5. Select **Create App**. The app details open.

App Credentials

These credentials allow your app to access the Slack API. They are secret. Please don't share your app credentials with anyone, include them in public code repositories, or store them in insecure ways.

App ID A09KZM4K2G2	Date of App Creation October 11, 2025
Client ID 9683195663732.9679718648546	
Client Secret	Show Regenerate
You'll need to send this secret along with your client ID when making your oauth.v2.access request.	
Signing Secret	Show Regenerate
Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.	

Now, follow the other instruction:

6. In the left menu under Features, select OAuth & Permissions.
7. In the Scopes section, select appropriate scopes for your app. Refer to [Scopes](#) for a list of recommended scopes.
8. Go to Scopes and click **Add an OAuth Scopes** > **Click Channels Read so on and so forth** (Add all the scopes in **Add an OAuth Scopes**, after you've added scopes, go up to the **OAuth Tokens** section and select **Install S2S** to Workspace > Click Allow. You must be a Slack workspace admin to complete this action.

Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description	Remove
channels:read	View basic information about public channels in a workspace	
chat:write	Send messages as @S2S-SOC_Project	
files:read	View files shared in channels and conversations that S2S-SOC_Project has been added to	
files:write	Upload, edit, and delete files as S2S-SOC_Project	
groups:read	View basic information about private channels that S2S-SOC_Project has been added to	
im:read	View basic information about direct messages that S2S-SOC_Project has been added to	

9. Select Allow.
10. Copy the Bot User OAuth Token and enter it as the Access Token in your n8n credential.

OAuth Tokens

These tokens were automatically generated when you installed the app to **S2S**. You can use these to authenticate your app. [Learn more](#).

Bot User OAuth Token

xoxb-9683195663732-9693778336705-zFPNOienkTEwD8nTkXkODdVf

Copy

Access Level: Workspace

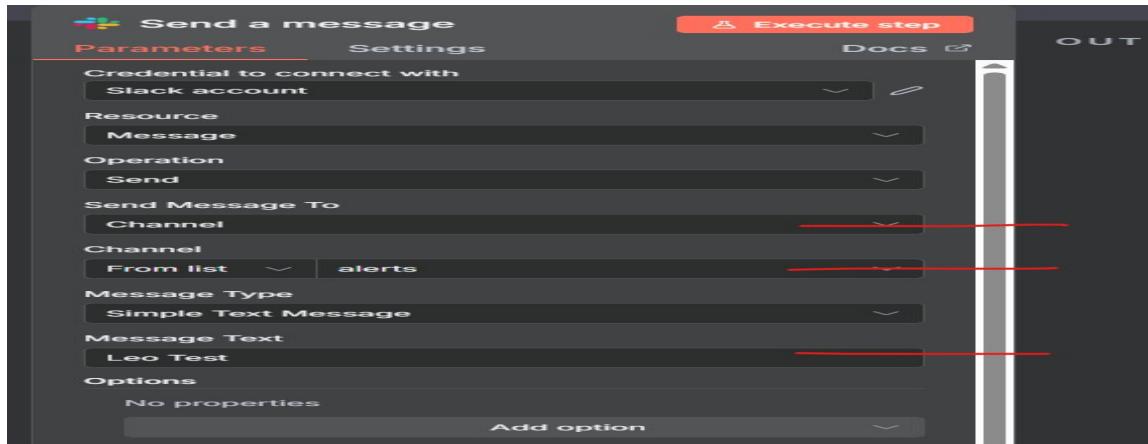
Reinstall to S2S

11. If you're using this credential for the [Slack Trigger](#), follow the steps in [Slack Trigger configuration](#) to finish setting up your app.

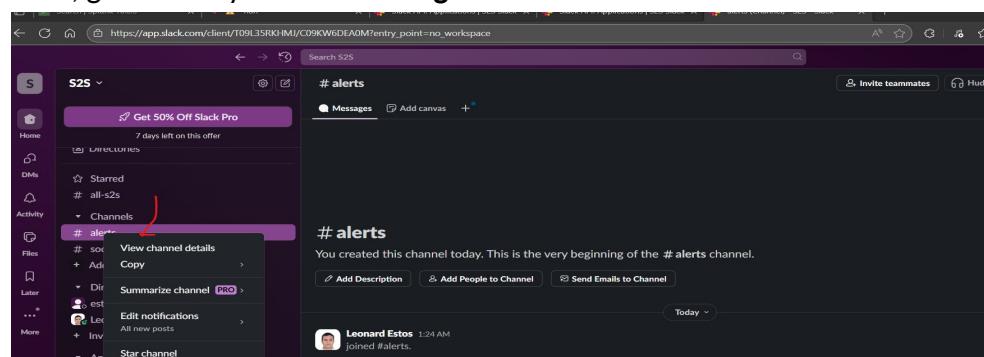
13. Now, go back to your N8N and copy the Bot User OAuth Token > Click Save.



- Under the **Send Message To: Channel**
- Under the **from List: Alert**
- **Message Text: Leo Test > Click Execute Steps**



- Now, you will encounter **error "Problem in node 'Send a message'** Slack error response: "not_in_channel"
- Now, go back to your **SLACK > Right click the # alerts > view channel details**



- Click **Integrations > Select Add Apps > Select S2S-SOC_Project**
- Now, go back to your **N8N** and **Click Execute Steps**. Result it's now working.

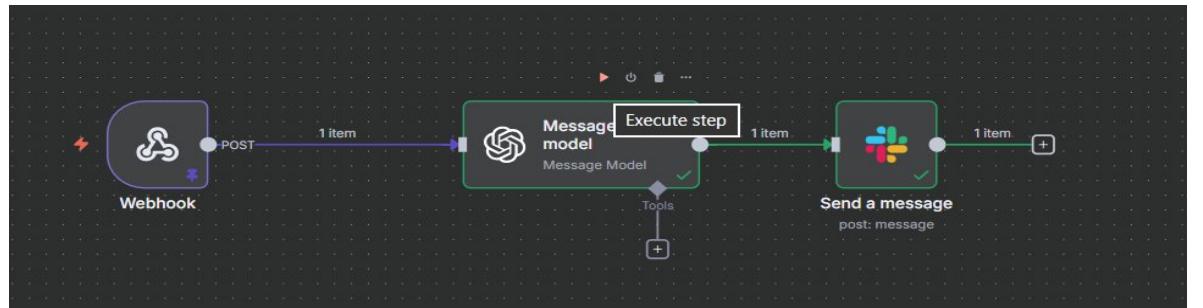
The screenshot shows the N8N canvas with a workflow. The input step is "Message a model" with content about a network scan. The main step is "Send a message" to Slack channel "#alerts" with message "Leo Test". The output shows the JSON response from Slack.

- Now, go back to SLACK again and click > # Alert > check if Leo Test is there.

The screenshot shows the Slack interface with the '#alerts' channel selected. It displays a message from Slackbot and two messages from the 'S2S-SOC_Project' app, one of which is 'Leo Test'.

14. Now, the Moment of Truth, we will now Test the workflow automation > Go back to your N8N > click Back to Canvas

- Click the Message a model > Click Play button seen below.
- Now, click the Slack > Click Play button.



- Check the Slack and #Alert will be available there.
- Now we need to put the Content Message in the N8N > In N8N > double click the SLACK and configuration will open again > Drag the content to Message Text > click Execute Step

The screenshot shows the n8n workflow editor. On the left, under 'INPUT', there is a node labeled 'Message a model' with several fields: 'index' (0), 'message' (with 'role' set to 'assistant' and 'content' containing a detailed alert summary), 'refusal' (null), 'annotations', 'logprobs' (null), and 'finish_reason' (stop). Below this is a 'Webhook' node and a 'Variables and context' node. On the right, the 'Send a message' step is configured with the following parameters:

- Credential to connect with:** Slack account
- Resource:** Message
- Operation:** Send
- Send Message To:** Channel
- Channel:** From list (alerts)
- Message Type:** Simple Text Message
- Message Text:** `{{ \$json.message.content }}`

The 'Result' section shows the output message: "The alert 'Leo-Brute-Force-Test' was triggered on the system 'LizJamesDC1.lizjames.com' involving login activity by the user 'administrator'. The source IP address is '0.0.0.0', which is unusual and typically indicates an invalid or placeholder IP. The alert indicates a single event, possibly a test or detected brute-force attempt using the administrator account." Below the message is a tip: "Tip: Anything inside {{ }} is JavaScript. Learn more."

- Now, go back to SLACK again > In # Alert check the results

Now for the SOC, this is a nice POC (Proof of Concept)

- POC**, a small-scale project or demonstration that proves an idea's feasibility and viability before committing significant resources. It shows that a concept can be successfully implemented in the real world, validate assumptions, and provide stakeholders with evidence that the project is worth pursuing. A POC is not a final product.

The screenshot shows the Slack interface with the '# alerts' channel selected. A message from the 'S2S-SOC_Project' app at 12:02 PM is shown:

```

**Summary:**  

The alert "Leo-Brute-Force-Test" was triggered on the system "LizJamesDC1.lizjames.com," involving login activity by the user "administrator." The source IP address is "0.0.0.0," which is unusual and typically indicates an invalid or placeholder IP. The alert indicates a single event, possibly a test or detected brute-force attempt using the administrator account.

**IOC Enrichment:**  

- "Source IP:" 0.0.0.0 is a non-routable meta-address used to denote an invalid, unknown, or non-applicable target. It is not associated with any external threat actor or malware.  

- **User:** "administrator" is a highly privileged account, often targeted in brute-force attacks by threat actors.  

- No additional IOCs such as domains or hashes were provided.

**Severity Assessment:**  

- **MITRE ATT&CK Tactic:** Initial Access / Credential Access  

- **Technique:** Brute Force (T1110) - Attempted access using repeated credential guessing.  

- **Severity:** Medium - The use of "administrator" and a brute-force test suggests potential reconnaissance or attack, but only one login attempt was recorded and source IP is anomalous.

```

- Now, we will test this with outside IP address basically those IP's that are used for abuse. Go to www.abuseipdb.com > get some test IP.

- Now, in your N8N > Click the OpenAI (ChatGPT) Message a Model > Go down and go to PROMPT > Add the Source IP: 94.248.237.19 came from abuseipdb.
- Now, change the Null with ['_time','user','ComputerName']

```

Prompt
Alert: {{ $json.body.search_name }}
Alert Details: {{ JSON.stringify ($json.on.body.result,null,2)}}
Source IP: 94.248.237.19

Result
Item [ 0 ]

```

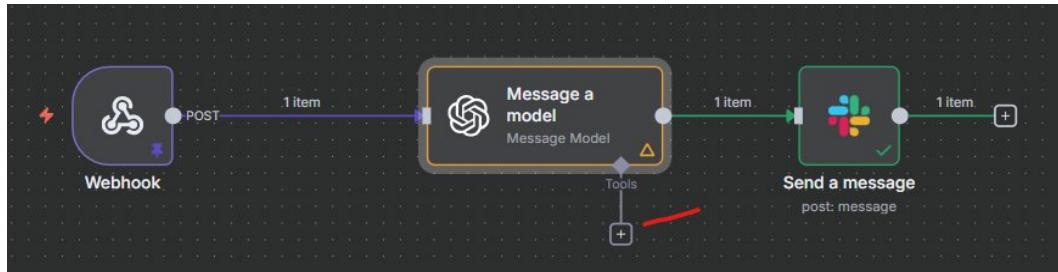
```

Prompt
Alert: {{ $json.body.search_name }}
Alert Details: {{ JSON.stringify ($json.on.body.result,['_time','user','ComputerName'],2)}}
Source IP: 94.248.237.19

Result
Item [ 0 ]

```

- Go back to your N8N > Click Back to Canvas > Now you can see a Tool below the Message a Model > Double click the + > Click HTTP request Tool



- Now, we need to sign-up in the www.abuseipdb.com to get the API Key. > Login to abuseipdb > Click API > Click Create key > Click CREATE



- Now click > Check out [our manual](#) to learn how to use our API. > Click CHECK Endpoint > Copy the Curl Code, see below

The check endpoint accepts a single IP address (v4 or v6). Optionally you may set the `maxAgeInDays` parameter to only return reports within the last x amount of days.

```
# The -G option will convert form parameters (-d options) into query parameters.
# The CHECK endpoint is a GET request
curl -G https://api.abuseipdb.com/api/v2/check \
--data-urlencode "ipAddress=118.25.6.39" \
-d maxAgeInDays=90 \
-d verbose \
-H "Key: YOUR_OWN_API_KEY" \
-H "Accept: application/json"
```

This will yield the following JSON response:

```
{
  "data": {
    "ipAddress": "118.25.6.39",
    "isPublic": true,
    "ipVersion": 4,
    "isWhitelisted": false
  }
}
```

- Now go back to your N8N > click IMPORT and paste the curl code.

Makes an HTTP request and returns the response data

Import cURL command

cURL Command

```
curl -G https://api.abuseipdb.com/api/v2/check \
--data-urlencode "ipAddress=118.25.6.39" \
-d maxAgeInDays=90 \
-d verbose \
-H "Key: YOUR_OWN_API_KEY" \
-H "Accept: application/json"
```

This will overwrite any changes you have already made to the current node

Import

- Once imported > Go down to the Query Parameters > Value removes the hardcoded IP > Change the IP with your test www.abuseipdb.com = 94.248.237.19

Query Parameters

Name: ipAddress

Value: 118.25.6.39

- Going down again, see the Specify Headers/Header Parameters > in Value: Paste your abuseipdp API Key

Send Headers

Specify Headers

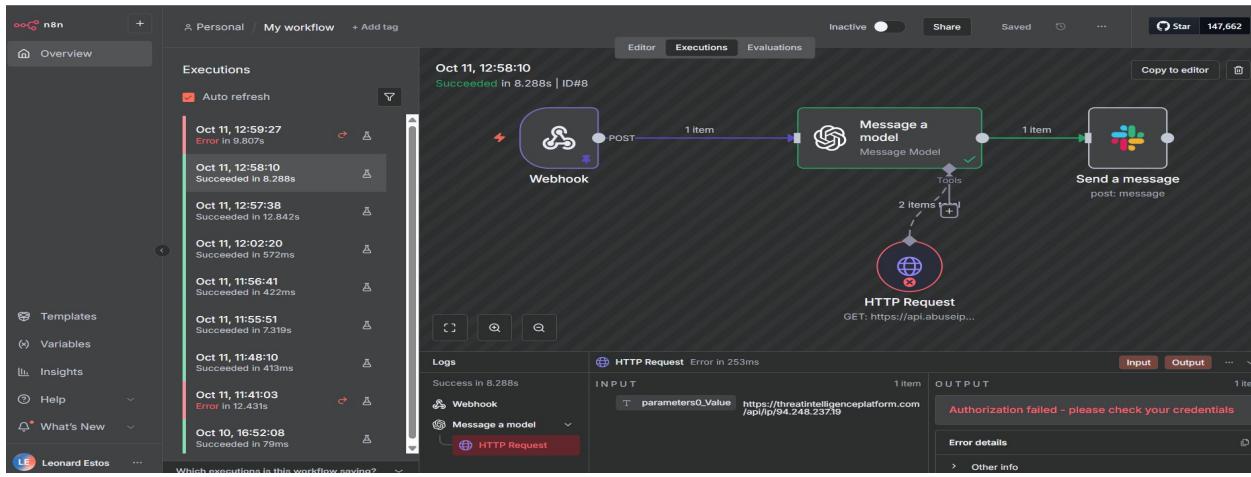
Using Fields Below

Header Parameters

Name: Key

Value: d7b9280572267079614bee027b7c92983051b

Name: Accept



" When you train Smarter, you defend Stronger"

Leonard Estos