

Project 3: Threat Intel Processor

This guide explains how to build a Python script that fetches malicious IPs from the **AbuseIPDB** threat feed and checks them against a sample log file.

Problem Statement: Develop a system that automates the consumption of open-source threat intelligence (TI) feeds to detect potential security threats within system and network logs, enabling faster identification of malicious indicators of compromise (IOCs).

Use Case: Build a Python-based tool that runs on a schedule, fetches the latest threat data (malicious IPs, domains, file hashes) from public feeds like **AbuseIPDB** and **AlienVault OTX**, and stores it locally. The tool will then compare these IOCs against simulated network or system logs to find matches, generating alerts for any suspicious activity, such as communication with a known command-and-control server.

Key Modules:

- Threat Intelligence Feed Aggregator (API Clients)
- IOC Storage (Local Database)
- Log Processing & Correlation Engine
- Anomaly Detection & Alerting Mechanism
- Simulated Log Generator

Step 1: Set up an API key

- a. Install Python and Libraries:

```
cybermonk@myLap:~ $ sudo apt update && sudo apt upgrade -y
```

```
cybermonk@myLap:~ $ sudo apt install -y python3 python3-pip
```

```
cybermonk@myLap:~ $ source venv/bin/activate
```

```
cybermonk@myLap:~ $ pip install requests
```

```

cybermonk@myLap:~$ source venv/bin/activate
cybermonk@myLap:~$ pip install requests
Collecting requests
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting charset normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.4-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (37 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2025.10.5-py3-none-any.whl.metadata (2.5 kB)
Downloading requests-2.32.5-py3-none-any.whl (64 kB)
 64.7/64.7 kB 1.2 MB/s eta 0:00:00
Downloading certifi-2025.10.5-py3-none-any.whl (163 kB)
 163.3/163.3 kB 3.4 MB/s eta 0:00:00
Downloading charset_normalizer-3.4.4-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (153 kB)
 153.5/153.5 kB 10.3 MB/s eta 0:00:00
Downloading idna-3.11-py3-none-any.whl (71 kB)
 71.0/71.0 kB 6.7 MB/s eta 0:00:00
Downloading urllib3-2.5.0-py3-none-any.whl (129 kB)
 129.8/129.8 kB 9.8 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset_normalizer, certifi, requests
Successfully installed certifi-2025.10.5 charset_normalizer-3.4.4 idna-3.11 requests-2.32.5 urllib3-2.5.0
cybermonk@myLap:~$

```

b. Get an AbuseIPDB API key: (Steps in **Annexure-1**)

Go to <https://www.abuseipdb.com/> and create a free account.

Navigate to your account section and go to "API".

Create an API key and copy it

Step 2: Create the Python script

Create a single Python file named [threat_checker.py](#).

```
cybermonk@myLap:~$ vim threat\_checker.py
```

```

import requests
import json
import sqlite3

#--- CONFIGURATION ---
API_KEY = 'f79817039c21acb71de0e5f3df54a89361ec797c5c400542b81cd1312d3ced3558b05d61b2802463'
# <-- IMPORTANT: PASTE YOUR KEY HERE
DB_FILE = 'threat_intel.db'

def setup_database():
    """Create the SQLite database and table if they don't exist."""
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()
    cursor.execute("""
CREATE TABLE IF NOT EXISTS iocs (
    ip_address TEXT PRIMARY KEY,
    abuse_confidence INTEGER,
    country_code TEXT
)
""")
    conn.commit()
    conn.close()

```

Key: f79817039c21acb71de0e5f3df54a89361ec797c5c400542b81cd1312d3ced3558b05d61b2802463

The Strategy: Use a Separate Input File

We'll create a new file, for example, [ips_to_test.txt](#). Your Python script will be changed to read every line from this file and use it to build the access.log for testing.

```

# 3. Generate the test log from an external file
input_file = 'ips_to_test.txt'
print(f"\nGenerating 'access.log' from '{input_file}'...")
try:
    with open(input_file, 'r') as infile, open('access.log', 'w') as outfile:
        for line in infile:
            outfile.write(line)
    print("Successfully created 'access.log'.")
except FileNotFoundError:
    print(f"Error: Input file '{input_file}' not found. Please create it.")
    # Create a blank access.log so the next step doesn't fail
    open('access.log', 'w').close()

# 4. Check the logs against our database
check_logs('access.log')

```

How It Works Now

Your workflow is now much simpler:

- To add, remove, or change the IPs you want to test, you only need to edit the [ips_to_test.txt](#) file.
- You never need to touch the `threat_checker.py` script again for this purpose.
- Just run the script as usual (`python3 threat_checker.py`), and it will automatically use the latest list of IPs from your input file for the scan.

```
cybermonk@myLap:~ $ ls
'!'  Infotact-Solution-Internship  access.log  ips_to_test.txt  threat_checker.py  threat_intel.db  venv
cybermonk@myLap:~ $ cat ips_to_test.txt
193.32.162.157
80.94.95.115
43.173.120.131
1.1.1.1
8.8.8.8
185.191.171.12
cybermonk@myLap:~ $
```

Step 3: Run and Verify

- Run the script:

```
cybermonk@myLap:~ $ python3 threat_checker.py
```

- Analyze the Output:** The script will first print that it's fetching data and updating the database. Then, it will scan the `access.log` file it created. Since the threat feed is live, the exact malicious IPs will change, but you should see at least one **"ALERT"** message if any of the IPs in your `access.log` happen to be on the current AbuseIPDB blacklist.

```
cybermonk@myLap:~ $ python3 threat_checker.py
Fetching latest threat intelligence...
Database updated. Added 0 new IPs.

Generating 'access.log' from 'ips_to_test.txt'...
Successfully created 'access.log'.

Scanning log file: access.log...
(!) ALERT: Malicious IP found in logs: 193.32.162.157 (Confidence: 100%)
(!) ALERT: Malicious IP found in logs: 185.191.171.12 (Confidence: 100%)
cybermonk@myLap:~ $
```

Step 4: Create the Cron Job:

Now we'll add the scheduled task to your user's "crontab" file.

- a. **Open the crontab editor:**

```
cybermonk@myLap:~ $ crontab -e
```

If it's your first time, it might ask you to choose a text editor. **nano** is usually the easiest option.

- b. **Add the cron job line:** Go to the bottom of the file and add the following line. This example will run the script every day at 3:00 AM

```
0 3 * * * /home/cybermonk/venv/bin/python3 /home/cybermonk/threat_checker.py >> /home/cybermonk/threat_checker.log 2>&1
```

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 3 * * * /home/cybermonk/venv/bin/python3 /home/cybermonk/threat_checker.py >> /home/cybermonk/threat_checker.log 2>&1
```

- c. **Save and Exit:**

In **nano**, press **Ctrl + X**, then **Y** to confirm, and **Enter** to save.

You should see a message like crontab: *installing new crontab*.

Annexure-1:

Step-by-step process of creating a free account on the website **AbuseIPDB** and generating an **API key**.

- a. Go to <https://www.abuseipdb.com/> and create a free account.



- b. **Registering an Account:** The third image shows the registration form being filled out with user details (Name: *Sandip Biswa*, Username: *sun272000*) for the free "Individual" plan.

Register - AbuseIPDB

Registration is free, and gives you access to our powerful [abuse lookup](#) and [reporting API](#), plus the ability to view and delete your own reports. If you need higher API limits, or a custom AbuseIPDB blacklist, sign up for a paid plan. All plans have a 30-day free trial!

Register an AbuseIPDB Account

Required Fields marked with: *

First Name *
Sandip

Last Name *
Biswa

E-Mail Address *
sandipbiswa10@gmail.com
Email is available

Company Name (if Applicable)
Enter Company Name

Country of Origin
Bhutan

Username *
sun272000
username is available

VAT No. / Tax ID (if Applicable) *
Enter VAT No. / Tax ID

Register As *
Individual

Individual **FREE** 1,000 Checks

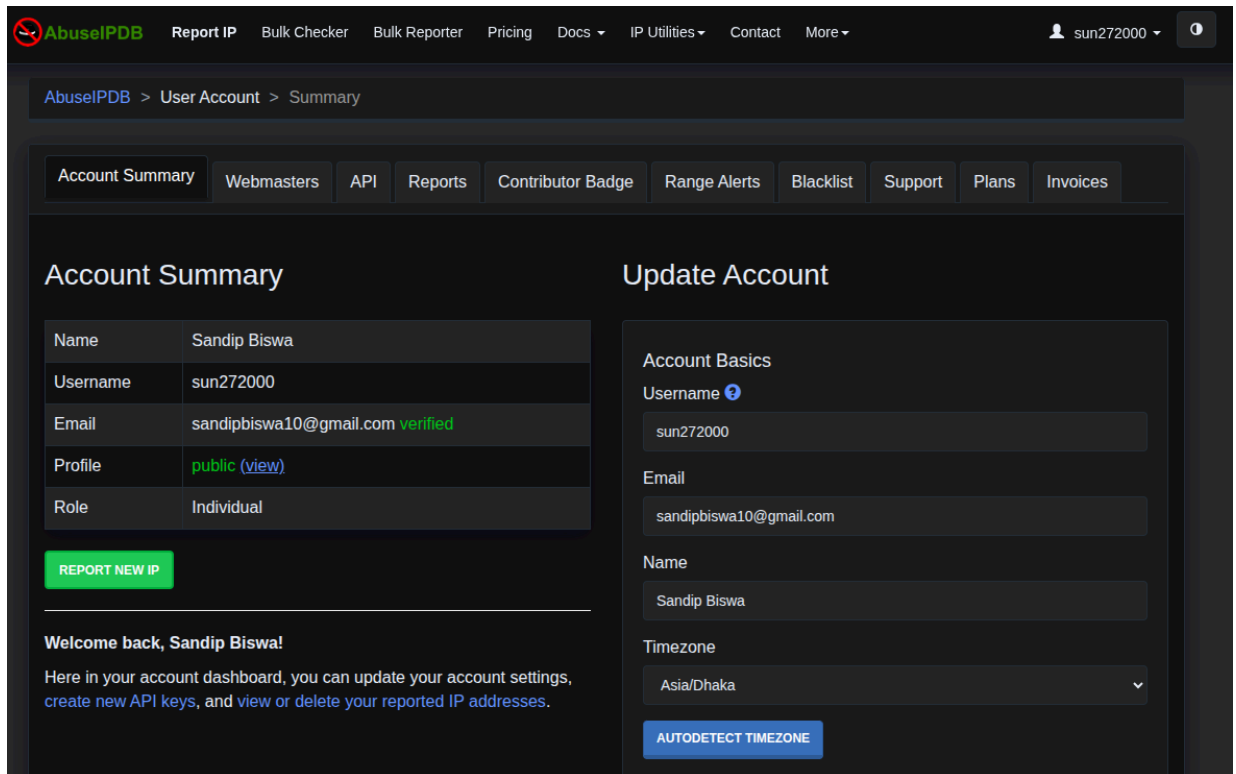
Join the AbuseIPDB community, contribute abuse reports, and use the API to protect your lower-traffic servers and websites

FREE FOREVER

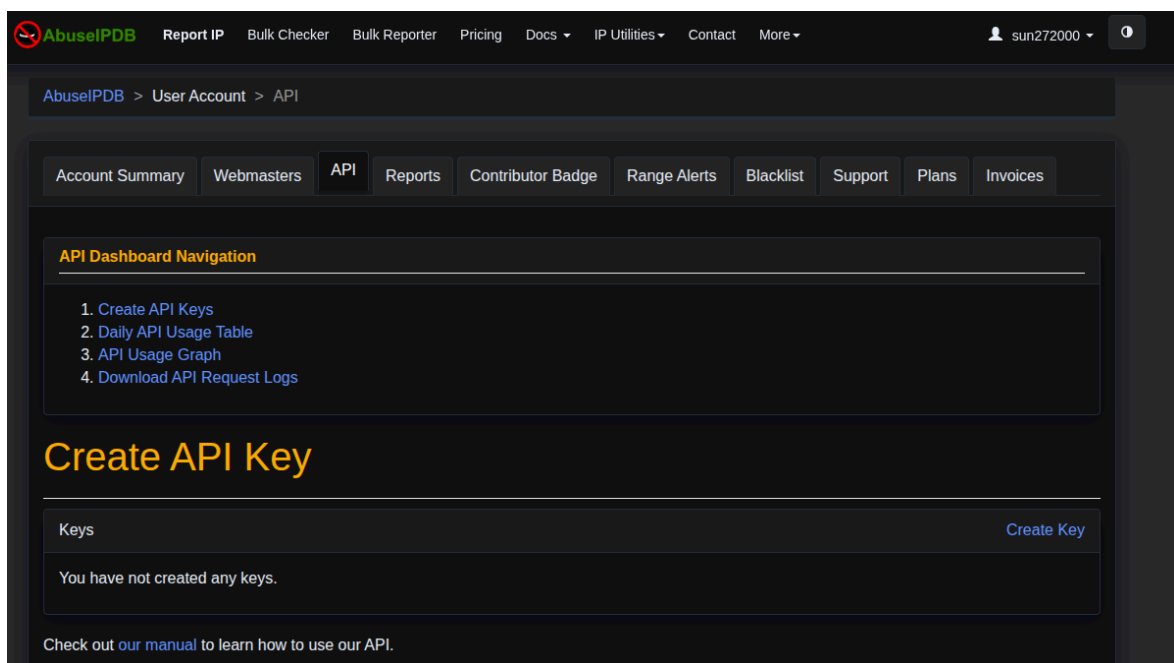
Basic Plan 10,000 Checks

Premium Plan 50,000 Checks

- c. **Navigating to the API Section:** The fifth image shows that the user has clicked on the "API" tab within their account dashboard, which initially shows that no API keys have been created.



- d. **Creating an API Key:** The sixth image shows that an API key named "Test-threat_intel_process" has been successfully generated.





- e. **Viewing API Usage:** The final image displays the "API Usage Graph" for the newly created key, showing a spike in "backlist" activity on October 17, 2025.

