

Forum:[Hack Forums](#)

Thread: [TikTok Mass Username Checker](#)

Clumsy - August 28, 2024, 10:09 pm

This Python script asynchronously checks username availability using proxies. It reads usernames from a file and retrieves a list of proxies. The script rotates through these proxies to make concurrent requests to an API for username validation. It handles errors gracefully and logs the results to a JSON file, indicating whether each username is available or taken. The use of asynchronous requests with aiohttp and asyncio ensures efficient processing of multiple usernames, while robust error handling and logging provide clear insights into the script's execution and outcomes.

Features

Proxy Rotation with Failover Handles proxy failures and rotates proxies seamlessly.

Logging Maintains a log of checked usernames and their status.

Concurrency Utilizes asynchronous requests to speed up the process.

Error Handling Robust error handling for both network issues and API responses.

Configurable Settings Allows easy customization for different needs.

Python Requirements

Code pip install aiohttp aiofiles

The Script

```
Code import aiohttp
import asyncio
import logging
from itertools import cycle
import json
from aiofiles import open as aio_open

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# Function to read usernames from file
async def read_usernames(filename):
    async with aio_open(filename, 'r') as file:
        usernames = await file.readlines()
    return [username.strip() for username in usernames]

# Function to get proxy list
async def get_proxies():
    url = 'https://api.proxyscraper.com/v2/?request=displayproxies&protocol=HTTP'
    try:
        async with aiohttp.ClientSession() as session:
            async with session.get(url) as response:
                response.raise_for_status()
                proxies = await response.text()
                return proxies.splitlines()
```

```
except aiohttp.ClientError as e:
    logger.error(f"Failed to fetch proxies. Error: {e}")
    return []
```

Function to check username availability

```
async def check_username(session, username, proxy):
    proxy_dict = {
        "http": proxy,
        "https": proxy
    }
    url = f'https://api.tiktok.com/username/check/?username={username}'
    try:
        async with session.post(url, proxy=proxy_dict.get("http")) as response:
            response.raise_for_status()
            result = await response.json()
            availability = "available" if result.get('available') else "taken"
            logger.info(f"Username '{username}' is {availability}!")
            return username, availability
    except aiohttp.ClientError as e:
        logger.error(f"Request failed for username '{username}'. Error: {e}")
        return username, "error"
```

Main function

```
async def main():
    usernames = await read_usernames('words.txt')
    proxies = cycle(await get_proxies())

    if not proxies:
        logger.error("No proxies available. Exiting.")
        return

    async with aiohttp.ClientSession() as session:
        tasks = []
        for username in usernames:
            proxy = next(proxies, None)
            if proxy:
                task = check_username(session, username, proxy)
                tasks.append(task)
            else:
                logger.error("No more proxies available.")
                break

        results = await asyncio.gather(*tasks)

    # Log results to file
    async with aio_open('results.json', 'w') as file:
        json.dump(dict(results), file, indent=4)

    logger.info("Processing completed. Results saved to 'results.json'.")

# Entry point
if __name__ == "__main__":
    asyncio.run(main())
```

Here is a wordlist

Code <https://www.mit.edu/~ecprice/wordlist.10000>

Here is list of working proxies

Code <https://pastebin.com/raw/XA7fwBi4>

Enjoy if anyone would like to donate [https://cash.app/\\$SleepTheGod](https://cash.app/$SleepTheGod)