

Forum:[Hack Forums](#)

Thread: [Program for searching and checking working proxy servers](#)

DarkG8 - April 12, 2025, 6:09 am

Features: Checks and installs required modules (requests, fake_useragent and pystyle)

Direct fix for errors related to missing libraries.

Parses lists from GitHub and specialized APIs

Tests proxies via Google query

Determines proxy country via API

Uses ThreadPoolExecutor to speed up testing

Stops when limit (user defined) is reached

Writes working proxies to the working_proxies.txt file

Generates random User-Agent for each request

Code:

```
Code
import sys
import subprocess
import os
import socket
import urllib.request
import time
from concurrent.futures import ThreadPoolExecutor
import requests
from fake_useragent import UserAgent
import random
from pystyle import Colors, Colorate, Center

# Constants for color output
RESET = "\033[0m"
GREEN_TEXT = "\033[32m"
RED_TEXT = "\033[31m"

# Global UserAgent object for caching
ua = UserAgent()

required_modules = [
    "pystyle",
    "urllib3",
```

```

    "fake_useragent",
    "requests"
]

def print_with_delay(text, delay=0.1):
    """Prints text with a delay between characters"""
    for char in text:
        print(char, end="", flush=True)
        time.sleep(delay)
    print()

def install(package):
    """Installs the specified package via pip"""
    subprocess.check_call([sys.executable, "-m", "pip", "install", package])

def check_and_install_modules():
    """Checks and installs required modules"""
    print_with_delay(
        GREEN_TEXT + "Welcome to the Free Proxy installation wizard.\n"
        "Now we will install all dependencies for the program to work properly\n"
        "Installation will not take more than 3 minutes.\n3\n2\n1\n" + RESET)

    for module in required_modules:
        try:
            __import__(module)
            print(GREEN_TEXT + f"{module} already installed." + RESET)
        except ImportError:
            print(f"Installation {module}...")
            install(module)
            print_with_delay(GREEN_TEXT + f"Module installed {module}" + RESET)

def user_agents():
    """Generates a random User-Agent"""
    browser_types = ['chrome', 'firefox', 'safari', 'edge']
    random_browser = random.choice(browser_types)
    return getattr(ua, random_browser)

def fetch_proxies():
    """Obtains a list of proxies from various sources"""
    proxies = set() # Use set to avoid duplicates
    timeout = 300
    urls = [

        "https://api.proxyscrape.com/v3/free-proxy-list/get?request=displayproxies&protocol=http&time"
        "out={timeout}",
        "https://raw.githubusercontent.com/monosans/proxy-list/main/proxies/http.txt"
    ]

    headers = {
        "accept": "text/plain, */*; q=0.01",
        "accept-language": "en-US,en;q=0.8",
        "user-agent": user_agents(),
    }

```

```

for url in urls:
    try:
        response = requests.get(url, headers=headers, timeout=10)
        response.raise_for_status()
        proxies.update(response.text.strip().split('\n'))
    except requests.RequestException as e:
        print(f"{RED_TEXT}Error when receiving a proxy: {e}{RESET}")

return list(proxies)

def is_proxy_working(proxy, timeout=3):
    """Checks if the proxy is functional"""
    try:
        test_url = "http://www.google.com"
        proxies = {"http": f"http://{proxy}"}
        response = requests.get(test_url, proxies=proxies, timeout=timeout)
        return response.status_code == 200
    except:
        return False

def get_country(ip):
    """Determines the country by IP"""
    ip_api_url = "http://ip-api.com/json/"
    try:
        response = requests.get(ip_api_url + ip, timeout=5)
        response.raise_for_status()
        return response.json().get('country', 'Unknown')
    except:
        return "Unknown"

def fetch_and_check_proxies():
    """Basic function of proxy verification"""
    try:
        proxies = fetch_proxies()
        print(f"{GREEN_TEXT}[*] Proxies found: {len(proxies)}{RESET}")

        working_proxies = []

        # Processing user input with validation
        while True:
            try:
                limit = int(input("[+] Enter the number of proxies you need: "))
                if limit > 0:
                    break
                print(f"{RED_TEXT}The number must be positive{RESET}")
            except ValueError:
                print(f"{RED_TEXT}Enter an integer{RESET}")

        with ThreadPoolExecutor() as executor:
            futures = []
            for proxy in proxies:
                if len(working_proxies) >= limit:

```

```
break
```

```
futures.append(executor.submit(
    lambda p: (p, get_country(p.split(':')[0])) if is_proxy_working(p) else (None, None),
    proxy.strip()
))
```

```
for future in futures:
    proxy, country = future.result()
    if proxy:
        print(f'{GREEN_TEXT}Working proxy: {proxy}, Country: {country}{RESET}')
        working_proxies.append(proxy)
        if len(working_proxies) >= limit:
            break
```

```
# Saving working proxies to a file
if working_proxies:
    with open("working_proxies.txt", "w") as f:
        f.write("\n".join(working_proxies))
    print(f'{GREEN_TEXT}[*] Working proxies are saved in working_proxies.txt{RESET}')
```

```
except Exception as e:
    print(f'{RED_TEXT}Error: {e}{RESET}')
finally:
    input(f'{Colors.white}[$] Press Enter to return to the main menu...')
```

```
def main():
    """Main function"""
    os.system('cls' if os.name == 'nt' else 'clear')
    check_and_install_modules()

    banner = """
    ????????????????? ?????????????????? ????????? ?????????? ????????? ???
    ?????????????????????????????????????????????????????????????????????????????????????????????
    ????????? ?????????????????????? ?????????? ?????????????????????????????? ?????????? ??????????
    ????????? ?????????????????????? ?????????? ?????????? ?????????????????????? ?????????? ?????????
    ??? ????? ?????????????????????????????????????? ??? ????? ?????????????????????????????????? ???
    ??? ????? ?????????????????????????????????????? ??? ????? ?????????????????????????????????? ???
    [%] Author: DarkG8
    """
```

```
os.system('cls' if os.name == 'nt' else 'clear')
print(Colorate.Horizontal(Colors.purple_to_red, Center.XCenter(banner)))
```

```
while True:
    user_response = input(Colors.white + "Want to start looking for proxies? - Yes / No: ")
    user_response = user_response.strip().lower()
    if user_response == 'yes':
        fetch_and_check_proxies()
    elif user_response == 'no':
        print("Close the program")
        time.sleep(1)
        sys.exit(0)
```

```
else:
    print(f'{RED_TEXT}The answer Yes or No is accepted{RESET}')

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print(f'{RED_TEXT}[] Program terminated by the user{RESET}')
        sys.exit(0)
    except Exception as e:
        print(f'{RED_TEXT}[] Critical error: {e}{RESET}')
        sys.exit(1)
```

tkboom - April 12, 2025, 6:23 am

It should bring a lot of convenience.

DarkG8 - April 12, 2025, 7:00 am

(Apr 12, 2025 06:23 AM)tkboom Wrote: [\[CITE\]](#) It should bring a lot of convenience.
Thank you very much.

Jamalio - April 28, 2025, 3:13 pm

Thanks again for your

DarkG8 - April 29, 2025, 3:19 am

(Apr 28, 2025 03:13 PM)Jamalio Wrote: [\[CITE\]](#) Thanks again for your
You're welcome, use it.

ginxforce - May 4, 2025, 7:44 pm

toress - May 5, 2025, 2:57 am

Thank you very much!

mixpodcast16 - May 6, 2025, 5:24 am

Thanks for sharing, im testing

DarkG8 - May 6, 2025, 7:11 am

(May 5, 2025 02:57 AM)toress Wrote: [\[CITE\]](#) Thank you very much!
Thank you.

(May 6, 2025 05:24 AM)mixpodcast16 Wrote: [\[CITE\]](#) Thanks for sharing, im testing
Oh, you must be the first person to test it. I'd be glad if you could share your opinion afterwards.

Rsynced - May 6, 2025, 9:04 am

Thanks for sharing, this is great!