

Report: Onion Search Scraper (Python)

1. Objective

The purpose of this Python script is to **automate the scraping of search results from a dark web (.onion) search engine**.

The script takes multiple search keywords (e.g., *market, forum, bitcoin*), fetches corresponding results, extracts structured data, and saves it into a JSON file for further analysis.

2. Tools & Libraries Used

- **Python 3.13**
- **requests_html** → For sending HTTP requests and handling pages.
- **BeautifulSoup (bs4)** → For parsing and extracting specific HTML elements.
- **json** → For exporting the scraped results into a structured format (JSON).
- **Tor Proxy (SOCKS5)** → Requests are routed through Tor running on **127.0.0.1:9050**.

3. Workflow of the Script

Step 1: Setup Session & Proxy

```
session = HTMLSession()  
proxies = {  
    'http': 'socks5h://127.0.0.1:9050',  
    'https': 'socks5h://127.0.0.1:9050'  
}
```

A Tor proxy is defined to ensure anonymity and access to **.onion** domains.

Step 2: Define Search Terms

```
search_terms = ['market', 'forum', 'bitcoin']
```

A list of keywords is defined. The script will loop through each keyword.

Step 3: Fetch HTML for Each Search

```
url = f"http://<onion_address>/search/?q={term}"
r = session.get(url, proxies=proxies)
soup = BeautifulSoup(r.text, "html.parser")
```

For each keyword, the script requests the search page and loads the HTML into BeautifulSoup.

Step 4: Parse Search Results

Each search result is inside `<li class="result">`.

From each result, the following fields are extracted:

- **Title** → `<h4><a>` tag
- **URL** → `href` attribute
- **Description** → `<p>` tag
- **Cite (source)** → `<cite>` tag
- **Last Seen** → ``

Example parsing code:

```
title = li.h4.find("a").text.strip() if li.h4 and li.h4.find('a') else None
url_ = li.find('a')['href'] if li.find('a') else None
desc = li.find("p").text.strip() if li.find('p') else None
```

Step 5: Store Data in JSON Format

For each keyword, results are stored in a dictionary:

```

results[term].append({
    "title": title,
    "url": url_,
    "description": desc,
    "cite": cite,
    "last_seen": last_seen
})

```

At the end, the entire dictionary is dumped into `results.json`.

4. Sample Output (JSON)

```
{
  "market": [
    {
      "title": "Dark Market",
      "url": "http://xyz.onion/market",
      "description": "Buy and sell goods anonymously.",
      "cite": "xyz.onion",
      "last_seen": "2 days ago"
    }
  ],
  "forum": [
    {
      "title": "Hacker Forum",
      "url": "http://abc.onion/forum",
      "description": "Discussion about exploits and security.",
      "cite": "abc.onion",
      "last_seen": "1 day ago"
    }
  ]
}
```

5. Advantages of This Script

- **Automation** → Multiple keywords can be searched in one run.
- **Structured Data** → JSON format makes analysis easier.
- **Anonymity** → All requests go through Tor (SOCKS5).

- **Scalability** → Easy to add more keywords or save data to a database later.

6. Limitations

- Requires Tor running on `127.0.0.1:9050`.
- If the target onion site uses heavy JavaScript, `requests_html.render()` may be needed.
- Scraping speed depends on Tor network latency.

CODE:

```
from requests_html import HTMLSession
from bs4 import BeautifulSoup
import json

session = HTMLSession()

proxies = {
    'http': 'socks5h://127.0.0.1:9050',
    'https': 'socks5h://127.0.0.1:9050'
}

search_terms = ['market', 'forum', 'bitcoin']
results = {}

for term in search_terms:
    url = f"http://juhanurmihxlp77nkq76byazcldy2hlmovfu2epvl5ankdibso4csyd.onion/search/?q={term}"
    r = session.get(url, proxies=proxies)
    soup = BeautifulSoup(r.text, "html.parser")
    list_li = soup.find_all('li', class_='result')

    results[term] = []
```

```
search_terms = ['market', 'forum', 'bitcoin']
results = {}

for term in search_terms:
    url = f"http://juhanurmihxlp77nkq76byazcldy2hlmovfu2epvl5ankdibso4csyd.onion/search/?q={term}"
    r = session.get(url, proxies=proxies)
    soup = BeautifulSoup(r.text, "html.parser")
    list_li = soup.find_all('li', class_='result')

    results[term] = []

    for li in list_li:
        title = li.h4.find("a").text.strip() if li.h4 and li.h4.find('a') else None
        url_ = li.find('a')['href'] if li.find('a') else None
        desc = li.find("p").text.strip() if li.find('p') else None
        cite = li.find('cite').text.strip() if li.find('cite') else None
        last_seen = li.find("span", {"class": "lastSeen"}).get_text(strip=True) if li.find("span",

            results[term].append({
                "title": title,
                "url": url_,
                "description": desc,
                "cite": cite,
                "last_seen": last_seen
            })

with open("results.json", "w", encoding="utf-8") as f:
    json.dump(results, f, indent=4, ensure_ascii=False)

print("✅ Data saved to results.json")
```

OUTPUT FILE:

The screenshot shows a VS Code interface running on a Kali Linux VM. The title bar indicates it's a VMware Workstation 17 Player session. The Explorer sidebar shows a folder structure for 'PYTHON_EH' containing files like '.vscode', 'env', 'bin', 'include', 'lib', 'lib64', '.gitignore', 'pyenv.cfg', 'search.py', 'python_basic', 'build_in.py', 'python_packages.py', and 'results.json'. The 'results.json' file is selected in the Explorer and is open in the main editor area. The editor shows JSON data with several entries, each representing a search result for 'bitcoin'. The first entry is for 'Bitcoin Generator Online' with a URL of <http://62wetcj3ivykvnmc7ferkhikyczfcyrrhuwy66huwlesumrk2r2yd.onion>. Other entries include 'BITCOIN STEALER SOFTWARE - Bitcoin Stealer Tools Download' and a page about paying with Bitcoin. The terminal below shows the command `python /home/kali/Documents/Python_EH/search.py` was run, and the output shows 'Data saved to results.json' twice. The status bar at the bottom right shows the file has 19965 lines, 2 spaces, and is in UTF-8 encoding.

```
14839     "bitcoin": [
19908     {
19910         "url": "/search/redirect?search_term=bitcoin&redirect_url=http://62wetcj3ivykvnmc7ferkhikyczfcyrrhuwy66huwlesumrk2r2yd.onion",
19911         "description": "Bitcoin Generator Online. Powerful free tool for mining and exploitat",
19912         "cite": "62wetcj3ivykvnmc7ferkhikyczfcyrrhuwy66huwlesumrk2r2yd.onion",
19913         "last_seen": "0 minutes"
19914     },
19915     {
19916         "title": "Bitcoin Generator Online - Generate BTC To Your Online Wallet!",
19917         "url": "/search/redirect?search_term=bitcoin&redirect_url=http://62wetcj3ivykvnmc7ferkhikyczfcyrrhuwy66huwlesumrk2r2yd.onion",
19918         "description": "Bitcoin Generator Online. Powerful free tool for mining and exploitat",
19919         "cite": "62wetcj3ivykvnmc7ferkhikyczfcyrrhuwy66huwlesumrk2r2yd.onion",
19920         "last_seen": "0 minutes"
19921     },
19922     {
19923         "title": "BITCOIN STEALER SOFTWARE - Bitcoin Stealer Tools Download",
19924         "url": "/search/redirect?search_term=bitcoin&redirect_url=http://lumos22ajbpwwzn24ccb",
19925         "description": "This bitcoin stealer when installed on any computer and configured wi",
19926         "cite": "lumos22ajbpwwzn24ccbhsqrrxapnbhszoppdzd7hsjseuqxiiipbad.onion",
19927         "last_seen": "0 minutes"
19928     },
19929     {
19930         "title": "¿Cómo pagar con Bitcoin? - Duxor",
19931         "url": "/search/redirect?search_term=bitcoin&redirect_url=http://duxorwood5csh76hpoo",
19932         "description": "Duxor es una herramienta de pago en línea que te permite pagar con Bitcoin",
19933         "cite": "duxorwood5csh76hpoo.onion",
19934         "last_seen": "0 minutes"
19935     }
19936 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ Data saved to results.json

• (env) - (kali㉿kali) - [~/Documents/Python_EH]

• \$ /home/kali/Documents/Python_EH/bin/python /home/kali/Documents/Python_EH/env/search.py

✓ Data saved to results.json

Ln 19965, Col 2 Spaces: 4 UTF-8 LF JSON