

TORRECON

TORRECON – Tool Introduction

TORRECON is a Tor Onion Service Reconnaissance Tool designed for ethical, non-intrusive security research.

It safely analyzes .onion websites through the Tor network to detect service status, onion version, backend technologies, frameworks, and exposed APIs.

The tool collects publicly accessible metadata, such as robots.txt and sitemap.xml, while respecting Tor anonymity and avoiding exploitation.

TORRECON is ideal for academic projects, OSINT research, and defensive security assessments, providing reliable insights without compromising ethical standards.

2. OBJECTIVE OF PoC

- Validate Tor-based routing
- Demonstrate onion service analysis
- Identify frameworks, backend technologies, APIs, and protections

3. ENVIRONMENT SETUP

Operating System: Windows / Linux

Python Version: 3.8+

Tor Service: Running locally

Libraries: requests, beautifulsoup4

4. ARCHITECTURE OVERVIEW

User → TORRECON → Tor SOCKS Proxy (127.0.0.1:9050) → Onion Service → Recon Output

5. STEP-BY-STEP PROCESS

Step 1: Start Tor service

Step 2: Install dependencies

Step 3: Execute TORRECON with onion URL

Step 4: Analyze HTTP responses

Step 5: Display reconnaissance report

6. REQUIRED IMAGES FOR PoC

Image 1: Environment Setup Screenshot

Python version,

Open terminal (CMD / PowerShell) and run:

```
bash
python --version
```

[Copy code](#)

or

```
bash
python -V
```

[Copy code](#)

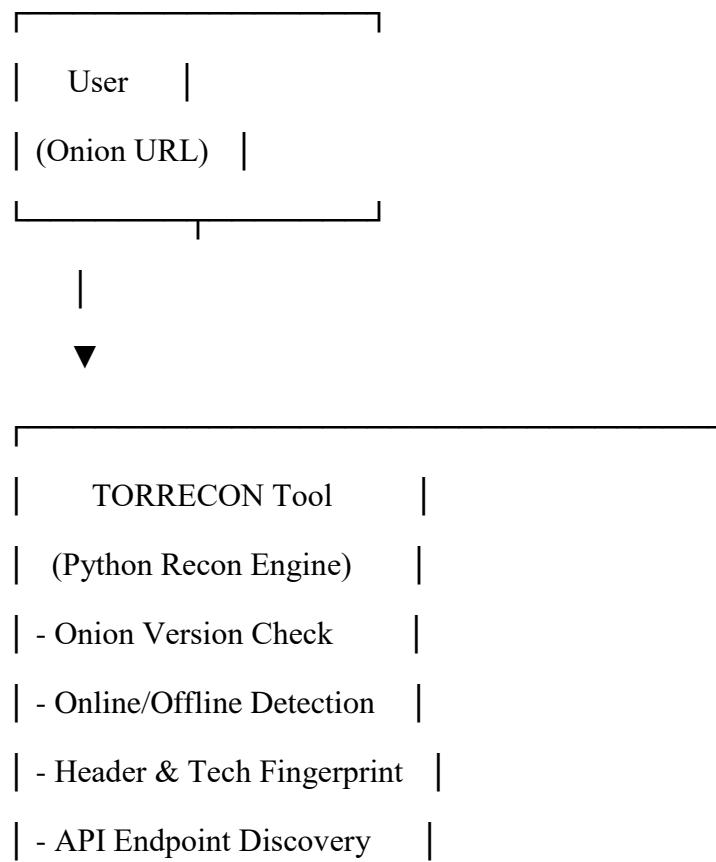
❖ Output example:

```
nginx
Python 3.13.0
```

[Copy code](#)

Image 2: Architecture Diagram

- Visual flow of TORRECON → Tor → Onion Service



| - robots.txt / sitemap.xml |



| HTTP Requests

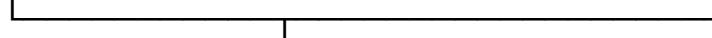
| via Tor Proxy



| Tor SOCKS Proxy (Local) |

| 127.0.0.1 : 9050 |

| socks5h routing enabled |



| Tor Network |

| (Encrypted Onion Routing) |



| Onion Service Target |

| http://xxxxxx.onion |



|

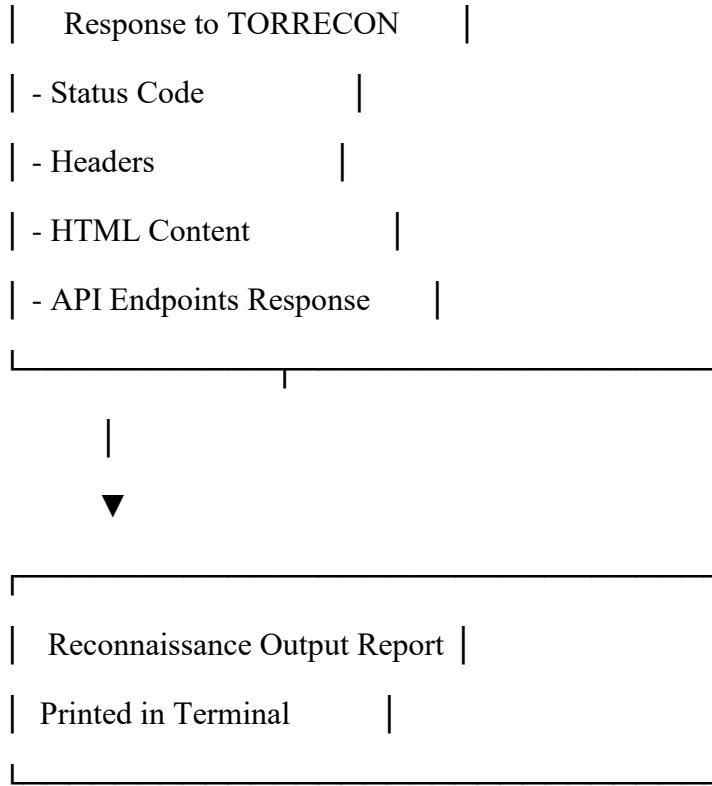


Image 3: Tor Service Running
- Terminal or Tor Browser showing Tor active

```

└─(kali㉿kali)-[~/Torrecon]
└─$ sudo service tor start

```

Image 4: Tool Execution Command
- python torrecon.py <onion_url>

```

└─(kali㉿kali)-[~/Torrecon]
└─$ python Torrecon.py http://xmh57jrknzkhv6y3ls3ubitzfqnkrxhopf5aygt

```

Image 5: Onion Version Detection Output
- v2 / v3 identification

```

[*] Onion Version:
  ✓ v3 onion service

```

Image 6: Online / Offline Status Output

- HTTP status result

```
[*] Service Status:  
    ✓ Online (HTTP 200)
```

Image 7: HTTP Headers Output

- Server and response headers

```
[*] HTTP Headers:  
    Date: Fri, 16 Jan 2026 16:06:06 GMT  
    Server: Apache/2.4.38 (Debian)  
    Last-Modified: Sat, 05 Sep 2020 19:58:49 GMT  
    ETag: "44-5ae9669f65500"  
    Accept-Ranges: bytes  
    Content-Length: 68  
    Content-Type: text/html
```

Image 8: Framework Detection Output

- Flask / Django / Express etc.

```
[*] Frameworks:  
    Unknown
```

Image 9: Backend Language Detection

- Python / PHP / Node.js

```
[*] Backend Language:  
    Unknown
```

Image 10: API Endpoint Discovery

- /api, /api/v1, /graphql results

```
[*] API Endpoints:  
    None found
```

Image 11: Tor-Aware Protection Detection

- CAPTCHA / JS challenge

```
[*] Tor-aware Protections:  
    None detected
```

Image 12: robots.txt Output

- robots.txt content

```
[*] robots.txt:  
    Not found
```

Image 13: sitemap.xml Output

- sitemap.xml content

```
[*] sitemap.xml:  
    Not found
```

7. RESULTS

All reconnaissance modules executed successfully via Tor without traffic leakage.

```
[v] TORRECON scan completed
```

8. CONCLUSION

The PoC confirms TORRECON as a safe, ethical, and effective Tor reconnaissance tool.