There are no broad discussions or procedures can be found on the internet on scraping the Dark Web. Our research purpose was to explore the E-commerce world of Dark Web and try to scrape or crawl the websites so that we could analyze the structures of e-commerce websites on dark web. By structure we mean that the html structure of the website, the content type and format that dark web developers prefer to include on the websites etc.

But due to the unavailability of a proper dark web scraper, we couldn’t be able to crack how to do it successfully. The first question is, can dark websites be scraped or crawled? The answer is, yes. Dark Web can be scraped and crawled. The problem is the availability of the scraper.

**Our Attempts**

We found some dark web scraper on Github. For example there is a developer who developed OSINT tool for scraping dark websites. The code is written in Python. We tried that code but doesn’t work. Maybe because the developer tried it in Kali Linux and also the instructions are not so clear, or maybe the lack of our skill which results in failure. However we found 2-3 other scraper in Github which also doesn’t work. There are websites like X-Byte enterprise crawling and iWeb scraping who offers good scraping tools but they all are paid tools.

We faced more challenges like security issues as we were trying to test the available tools from our personal pc, scraping is something that can create a high chance of exposing our IP address to the hackers around the dark web. So we couldn’t just attempt some riskier task which may provide us the success but also expose our IP address on the dark web. There was also no one who will guide us to the proper path we should in the way of crawling dark web.

Some Information and Descriptions about tools that we failed to use:

TorBot is an open source intelligent tool that can be helpful for us for scraping dark web. It can be found on GitHub where 24 contributors are working on this. It includes .onion crawler as well. But the basic setup was complex, we failed to do the basic setup before installing it. But the features and description of it are written below just to inform about it.

If you’re looking for an advanced tool for dark web research, TorBot probably is and will continue to be overkill. As of this writing, the last update to TorBot was in February. It uses Python 3.x and requires a Tor dependency. TorBot has a list of features that makes it useful for multiple applications. Features include:

1. Onion Crawler (.onion).
2. Returns Page title and address with a short description about the site.
3. Save links to database.
4. Get emails from site.
5. Save crawl info to JSON file.
6. Crawl custom domains.
7. Check if the link is live.
8. Built-in Updater.

OS Dependencies

* Tor
* Python 3.x

Python Dependencies

* beautifulsoup4
* pyinstaller
* PySocks
* Termcolor
* Requests
* requests\_mock
* yattag

**Fresh Onions**

**Fresh Onions** is a tool that hasn’t been updated in a while. As a disclaimer, you may have issues running the script as 2017 was the last GitHub push. However, even as an academic piece of what is possible on the dark web using Python, it’s worth taking a look at what features this tool offers or once offered. Here’s a list of the features:

* Crawls the darknet looking for new hidden service
* Find hidden services from a number of clearnet sources
* Optional fulltext elasticsearch support
* Marks clone sites of the /r/darknet superlist
* Finds SSH fingerprints across hidden services
* Finds email addresses across hidden services
* Finds bitcoin addresses across hidden services
* Shows incoming / outgoing links to onion domains
* Up-to-date alive / dead hidden service status
* Portscanner
* Search for “interesting” URL paths, useful 404 detection
* Automatic language detection
* Fuzzy clone detection (requires elasticsearch, more advanced than superlist clone detection)

Infrastructure

Fresh Onions runs on two servers, a frontend host running the database and hidden service web site, and a backend host running the crawler. Probably most interesting to the reader is the setup for the backend. TOR as a client is COMPLETELY SINGLETHREADED. I know! It's 2017, and along with a complete lack of flying cars, TOR runs in a single thread. What this means is that if you try to run a crawler on a single TOR instance you will quickly find you are maxing out your CPU at 100%.

The solution to this problem is running multiple TOR instances and connecting to them through some kind of frontend that will round-robin your requests. The Fresh Onions crawler runs eight Tor instances.

Debian (and ubuntu) comes with a useful program "tor-instance-create" for quickly creating multiple instances of TOR. I used Squid as my frontend proxy, but unfortunately it can't connect to SOCKS directly, so I used "privoxy" as an intermediate proxy. You will need one privoxy instance for every TOR instance. There is a script in "scripts/create\_privoxy.sh" to help with creating privoxy instances on debian systems. It also helps to replace /etc/privoxy/default.filter with an empty file, to reduce CPU load by removing unnecessary regexes.

**TorCrawl**

Another crawling tool developed in Python. TorCrawl not only crawls hidden services on tor, it extracts the code on the services’ webpage. So, what is this useful for? In a world with infinite time, you could setup and run TorBot, figure out how to get everything running, and have a reliable tool that will consistently get new DLCs. In a semi perfect world you’d have the time to database services with subscriptions, manual tools, and Fresh Onions, then inspect each onion webpage for possible malicious content, then manually inspect each page for your investigation. But it’s not a perfect world and in most cases, the Pareto Principle applies and you have to get the most amount of work done in the least amount of time. So instead of worrying about crawling, inspection, then investigation, do it all in one with TorBot. You get the webpage markup so you can view the content without having to physically access the page. You can also view the static webpage by saving it as an .html file.