BŞEÜ Fen Bilimleri Dergisi
6 (2), 147-161, 2019

BSEU Journal of Science
DOI: 10.35193/bseufbd.608555

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Araştırma Makalesi - Research Article

TOR Gizli Servis Tarayıcılarının Performans Karşılaştırması

Merve VAROL ARISOY [1*], Ecir UĞUR KÜÇÜKSİLLE [2]

Geliş / Received: 21/08/2019 ***Revize / Revised: 21/11/2019*** ***Kabul / Accepted: 06/12/2019***

ÖZ

TOR (The Onion Routing), kullanıcısına anonimliği sağlaması sebebiyle son zamanlarda popülerliği artan ve onion uzantılı gizli servisler tarafından sıklıkla tercih edilen bir ağ yapısıdır. Gizliliğin esas olması nedeniyle dikkatleri üzerine çeken bu ağda, her geçen gün depolanan veri miktarı artmakta bu da verilerin taranma ve analiz edilme durumlarını zorlaştırmaktadır. Bu ağda yer alan servislerin (onion uzantılı web sayfaları) taranması için çeşitli crawler yazılımları geliştirilmiştir. Yalnız, burada yapılan tarama yüzey ağında yapılan taramadan farklıdır. Çünkü TOR ağı, yüzey ağının alt katmanlarında yer almakta ve buradaki sayfalara yalnızca TOR tarayıcısı aracılığıyla ulaşılmaktadır. Geliştirilen crawler yazılımlarında bu durum dikkate alınmış ve gizliliği korumak adına, adreslere yapılan her istekte farklı düğümler üzerinden yol seçimi yapılarak veri edinimine dikkat edilmiştir. TOR ağında kullanıcıların gönderdiği her istekte farklı düğümler üzerinden geçilerek hedef adrese ulaşılması bu ağı yavaşlatmaktadır. Ayrıca, TOR üzerinden bilgi getirmeye çalışan bir crawler yazılımının performansının düşük olması da uzun süreler beklemeyi beraberinde getirir. Bu yüzden tarama ve bilgi elde etme hızı yüksek crawler yazılımları ile çalışmak, araştırmacıların analiz süreçlerini de iyileştirecektir. Bu alanda araştırma yapacak olan kişileri yönlendirmesi ve crawler yazılımlarının birbirlerine karşı olan üstün ve zayıf yönlerinin değerlendirilmesi açısından 4 farklı crawler yazılımı çeşitli kriterlere göre değerlendirilmiştir. Gerçekleştirilen çalışma, TOR web servislerinin analizini yapmak isteyen araştırmacıların ilk çıkış noktaları anlamında doğru bir crawler yazılımını seçmeleri hususunda önemli bir bakış açısı sunmaktadır.

Anahtar Kelimeler- TOR, Crawler Yazılımı, Performans Karşılaştırması

[1]*Sorumlu yazar iletişim: mvarisoy@mehmetakif.edu.tr (http://orcid.org/0000-0003-2085-1964)
Enformatik Bölümü, Burdur Mehmet Akif Ersoy Üniversitesi, Burdur Mehmet Akif Ersoy Üniversitesi İstiklal Yerleşkesi 15030/BURDUR
[2]İletişim: ecirkucuksille@sdu.edu.tr (http://orcid.org/0000-0002-3293-9878)
Bilgisayar Mühendisliği Bölümü, Süleyman Demirel Üniversitesi, Süleyman Demirel Üniversitesi 32260 Çünür/ISPARTA

BŞEÜ Fen Bilimleri Dergisi
6 (2), 147-161, 2019

BSEU Journal of Science
DOI: 10.35193/bseufbd.608555

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Performance Comparison of TOR Hidden Service Crawlers

ABSTRACT

TOR (The Onion Routing) is a network structure that has become popular in recent years due to providing anonymity to its users and is often preferred by hidden services. Because the privacy is essential, this network draws attention, so the amount of data stored increases day by day, making it difficult to scan and analyze. Various crawler software has been developed in order to scan the services (onion web pages) in this network. However, crawling here is different from the surface network. Because the TOR network is located on the lower layers of the surface network and the pages in TOR are accessed only through the TOR browser. In the requests made to the addresses, to protect the confidentiality, the data was obtained by selecting paths through different relays. In TOR network, reaching the target address by passing over different relays in each request, slows down it. Also, the low performance of a crawler that tries to retrieve information through TOR, brings long periods of waiting. Therefore, working with a software with high crawling and information acquisition speed, will improve the analysis process of the researchers. 4 different crawler software was evaluated according to various criteria in terms of guiding the people who will conduct research in this field and evaluating the superior and weaknesses of the crawlers against each other. The study provides an important point of view for choosing the right crawler in terms of initial starting points for the researchers want to analyze of Tor web services.

Keywords- ***TOR, Crawler Software, Performance Comparison***

BŞEÜ Fen Bilimleri Dergisi
6 (2), 147-161, 2019

BSEU Journal of Science
DOI: 10.35193/bseufbd.608555

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

I. INTRODUCTION

The internet environment (www, world wide web), which contains an large amounts of data, apart from the surface network part of which general computer users are interested (such as searching, sending e-mail), consists of deep web and dark web (darknet) sections, which are not known much and which constitutes a large section of the internet. The section of the internet network that can be indexed by search engines represents the surface network. The indexing process performed here is provided by the search engines using small pieces of software called the web crawler that perform the crawling. [1].

When a crawler visit a web page, it detects all links from that page to other pages and then begins to visit these new pages. In the meantime, it sends the data obtained from the pages to the search engine and ensures that these data are indexed and stored in the search engine databases as keywords and page locations. So when an internet user writes and submits a query to the search engine, the search engine returns the pages in the database that match the word in the query [1].

The Deep Web section of the Internet, which contains more than the amount of data available on the surface network, contains web pages that are not indexed by search engines and do not link to the pages on the surface network. [1].

In the lower layers of Deep Web, there is a network structure called Dark Web (Dark Net) whose data size is not known exactly. In this network, which can be accessed by using special software such as TOR, I2P, FreeNet, data transmission is provided by providing an encrypted structure. As with the Deep Web, web pages on the Dark Web are not indexed. Anonymity is provided to users in this network by hiding their location and IP address, and the data transmitted across the network is encrypted and sent from end to end. [1]. Figure 1 gives a representation of how much area is covered by each of the Internet layers.

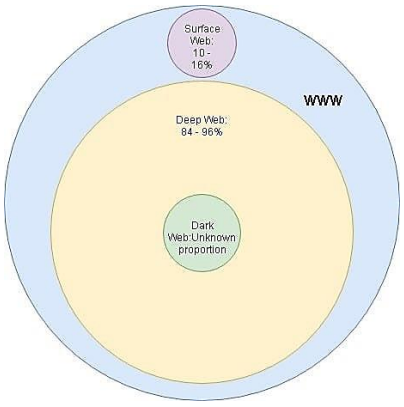

Figure 1. Layers of the internet [2]

A. What is TOR?

TOR (The Onion Router) is a network structure that hosts more than 3000 servers at different points in the world and transmits data from one end to the other by using these servers as input, intermediate and output nodes in established virtual network circuits. TOR software, which is operated by a non-profit organization, is used by people who request their transactions and identity to be kept confidential while on the internet. Therefore, this structure hosts some web pages that may constitute a crime (Such as arms trade, sensitive information trade, malware and spyware trading) as well as personal transaction movements in which the traffic on the network is requested not to be monitored. [1].

The TOR network has attracted the attention of many segments from different areas due to its privacy policy. Therefore, the amount of data it contains is increasing day by day. The need to access and process large

amounts of data accumulated here has also emerged. Analysis of the Tor network will allow situations such as to find connections between addresses, determining whether a page has been cloned or not. In addition, it is necessary to monitor Tor traffic in order to uncover illegal activities and to create related measures. For this purpose, numerous crawler software are being developed [3, 4, 5]. This softwares send requests to the seed onion address which is given them as a parameter and download the content of that address if the connection is provided. The data obtained may be in HTML or JSON format, depending on the coding format of the crawler software. In some crawler software, only the index page is obtained, while in others, the sub-pages of the onion link or other pages associated with it can be downloaded. It has been observed that among the crawler software examined, there is a software which only extracts the links on the given onion page and gives it in JSON format.

B. The Working Principle of TOR

TOR is a network structure that adds 3 types of intermediate nodes (input, intermediate, exit nodes) between the source user and the destination address during the access of a web page.

The basis of the onion routing used in the Tor network is based on the multilayered encryption of data transmitted from the source address to the destination address as it passes through the nodes within the virtual network circuit set up when the source user requests access to a web page. In order to protect confidentiality, such an encryption method is used. [6].

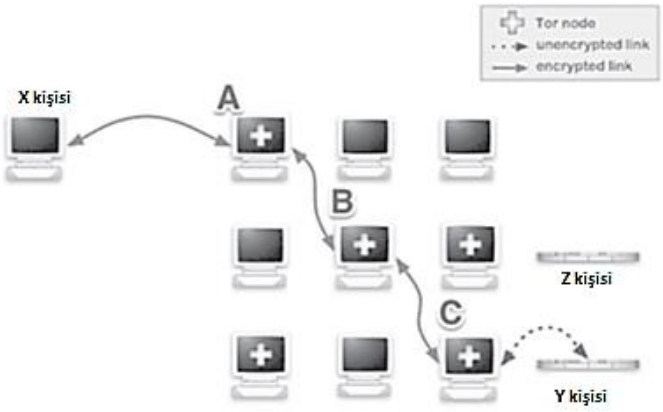

Figure 2. The working principle of TOR [6]

When the Figure 2 is examined, it can be seen that the person X communicates with the person Y in an anonymous form (by hiding the IP address and location information). Here, a virtual network circuit is created during the web page access request. In this virtual circuit, point A is the input node, point B is the intermediate node, and point C is the exit node. The nodes A, B, C selected here are randomly determined, and these nodes may change in another communication that X will make later with Y. In order to ensure confidentiality during this communication, the access request of the X person is encrypted in multi-layer and sent to node A which is the input node. At each node, one encryption layer is removed from the encrypted data and sent to the next node. Therefore, a node only recognizes and communicates with the next node. Only the last node (C, exit node) recognizes the destination address, but does not know from which source the request is made to this address. The number of nodes to be used in the transmission of data may be more than one. [1].

The aim of this study is to give superior, weaknesses, and performance comparisons of the selected crawler software. Subsequently, to provide an important point of view to the researchers who want to analyze TOR hidden services, to choose the right crawler software which is best suited to their requirements.

For this purpose, 4 different crawler software were selected. As feed address, same links were given to each agent software and the crawl conditions were examined. One of the links given here is

"torlinkbgs6aabns.onion" and the rest consists of the addresses on this page. In the following chapters, respectively, firstly literature studies are given, and then features of each crawler software and how to operate these crawlers are mentioned. Then, the findings section provides detailed information about the crawling results and performance comparisons obtained by crawlers as a result of the crawl performed.

In the fifth chapter, a general evaluation of the crawler softwares is presented and the researchers who will work in this field are guided.

II. LITERATURE

In the work of [3], a crawler software for the general evaluation of the content and the use of the English TOR ecosystem is developed. In their software, through subject and network analysis, they have characterized web pages and other linked pages which are hosted on the TOR network, in the context of their content.

In the study of [7], they proposed a new URL ranking algorithm and developed a web crawling software by applying various web mining techniques on these suggestions. Firstly, they gave feed address to their systems and then they obtained a score for the link crawled by making semantic analysis and site popularity according to the algorithm they proposed. According to the magnitude of the score they obtained, they arranged the priority of downloading the related link. They then downloaded web documents with threads running in parallel. Rather than developing web crawling software in their work, they focused on ranking the importance of the links given to this software as a feed address. Furthermore, their work has been developed for the surface network only, not including the TOR network.

In the work of [5], they presented a TOR hidden service crawling method that can monitor the status of the onion pages in the TOR network for content analysis. They developed TOR crawling software with Docker support. They also ran the TOR browser in parallel to improve crawling performance. They applied a clustering technique on the links they obtained as a result of the crawling and thus reduced the targets for crawling. Thus, they do not crawl the pages repeatedly which are with different addresses but similar content, thus improved the crawling time. Their work is mostly on proposing a method for improving crawling performance. Likewise, another study on performance belongs to [8]. In this study, they propose a method to crawl web databases faster than traditional crawlers. In their method, hundreds of threads are run over a single web crawling software on a single computer, and these threads are distributed to hundreds of proxy servers. They stated that the web crawling strategy they use greatly increases the speed of crawling and analysis and is safer than using a single web crawler thread.

In the work of [4], used a crawling software running on the TOR network by running a large number of TOR servers, to collect lists of hidden services, to classify their contents, and to calculate the number of requests. In the study of [9], they conducted a review of the features and performance comparison of various open-source crawling software such as Scrapy, Apache Nutch, Heritrix, WebSphinix. They evaluated the comparison according to criteria such as the features of the crawler software, the language in which they were written, the operating system on which they were working, their licenses, their parallel operation status, their speed and the type of link visited. Their work covers only the surface web. In this study, crawler software running on the TOR network was analyzed, but in Yadav and Goyal's study, this analysis was limited to the surface network.

In the work of [10], they conducted a study on identifying crawling software behaviour using web server access logs from five different academic sites in three countries. Based on these logs, they analyzed the effectiveness of different crawlers belongs to five search engines, including Google, AltaVista, Inktomi, FastSearch, and CiteSeer. In the analysis of browser behaviour, they considered general www (world wide web) traffic and access logs characteristics. They also evaluated the requests of the crawlers to web pages.

In the study of [11], a data collection and analysis with a crawler software based on social network analysis and content selection from web sites hosted in TOR hidden services and which support terrorism, extremists is conducted.

In the work of [12], they examined an e-commerce site that trades illegal products. Their crawler software simulates the authentication step that allows people to log into this illegal e-commerce site, and also collects data using LAMP Stack which is a web development platform.

In the study of [13], they developed a crawler software that identifying web pages which provides information about the way handmade explosives are made and that trade it. By giving crawler software some feed addresses as the initial address set, they aim to reach pages that overlap with the content in those addresses. A similar crawler software on this subject is described by [14]. Also, in the work of [15], they proposed new crawling methods that can be used on TOR; in the work of [16], they developed advanced crawling and indexing systems like LIGHTS, for use on the TOR network.

In the study of [17], they developed a crawler software that serves a search engine which aims to detect harmful content comes from suspicious websites. Their crawler software downloads the contents of web pages that are given as feed addresses, in HTML format and also by detecting the links from the web pages of these feed addresses to other pages, stores the contents of these new link pages. It then performs a scan of the presence of suspicious information on all the data obtained.

A crawler software can crawl web pages hosted on the TOR network, and also crawl on hidden databases too. In the study of [18], a semi-automatic crawler is developed which is able to crawl hidden databases and obtain data from these databases.

III. PROPERTIES OF THE CRAWLERS

This section provides detailed information about the characteristics of the crawling softwares and how they are operated. Respectively, the reviewed crawling softwares are OnionCrawler, TorBot, TorScrapper and OnionScan.

A. OnionCrawler

It is a scrapy spider to recursively crawl for TOR hidden services. In Table 1. , some prerequisites (libraries) to be installed before running OnionCrawler software and default usage of OnionCrawler is given.

Table 1. Prerequisites and default usage of OnionCrawler

Prerequisites	Default Usage
Torsocks	torsocks scrapy crawl OnionCrawler (By Default, No Search Terms are Set So All Crawled Websites are Scraped.)
Python 2.7	
Scrapy	
For Postgresql Support	
Python-Sqlalchemy	
Python-Psycopg2	

To customize the crawler's operating behaviour the argument/value pairs described in Table 2. can be used by adding each with a prefix '-a'. The use of the arguments given in Table 2 is accomplished by appending them to the end of the expression "torsocks scrapy crawl OnionCrawler". Some example operation modes of this crawler are given in Table 3. All operating modes of this crawler are realized by first opening the terminal screen in the ubuntu system and then writing the necessary operating modes here.

Table 2. Some argument/value pairs to customize the crawler's behaviour

Argument/Value	Usage Type	Description
inputURL=<SingleURL>	torsocks scrapy crawl OnionCrawler –a inputURL=https://onion.torproject.org	Single URL to Start Crawling
inputOnionList=<pathToOnionList>	torsocks scrapy crawl OnionCrawler –a inputOnionList=list.txt	Path to .onion URL List
inputHSProbeLog=<HSProbeLogFile>	torsocks scrapy crawl OnionCrawler –a inputHSProbeLog=hsprobe.log	Read onion Address from HS Probe Logfile
searchTerms=<SearchTerms>	torsocks scrapy crawl OnionCrawler –a inputOnionList=list.txt –a searchTerms='TOR OR onion'	Search Terms can be Logically Linked Using AND and OR Operators
caseSensitive=<trueOrFalse>	torsocks scrapy crawl OnionCrawler –a inputOnionList=list.txt –a searchTerms='TOR OR onion' –a caseSensitive=true	Activate Case Sensitivity When Searching for Terms
pipelineFile=<trueOrFalse>	torsocks scrapy crawl OnionCrawler –a inputOnionList=list.txt –a searchTerms='TOR OR onion' –a caseSensitive=true –a pipelineFile=true	Deactivate Pipeline Scraping to Filesystem
pipelinePostgres=<trueOrFalse>	torsocks scrapy crawl OnionCrawler –a inputOnionList=list.txt –a searchTerms='TOR OR onion' –a caseSensitive=true –a pipelinePostgres=true	Activate Pipeline Scraping to Postgres Database

Table 3. Other usage versions of OnionCrawler

Other Usage Versions of Crawler	
torsocks scrapy crawl OnionCrawler -a inputOnionList=list.txt -a searchTerms='TOR AND onion' -a caseSensitive=true -a pipelineFile=false -a pipelinePostgres=true	This Crawling Type Reads .onion Names from list.txt, Scrape All Crawled Websites Which Contain Both Search Terms "TOR" and "onion" (Case Sensitive) and Store in PostgreSQL Database.
torsocks scrapy crawl OnionCrawler -a inputURL=https://onion.torproject.org/ -a searchTerms='"Apache Server" OR "The Tor Project"' -a caseSensitive=true	This Crawling Type Reads Hidden Services from the TOR Project's List of onion Services, Search Crawled Websites for One of the Search Terms "Apache Server" or "The Tor Project" (case sensitive), and Store in Filesystem.

If the crawled links are to be stored only as a file (not in the database), the corresponding pipeline must be activated or if it is desired to be recorded in the database, this time the pipeline permission for the database should be given. An example of the "pipeline Postgres = true" specified in the first use in Table 3. , shows that the permission to save downloaded data to a database is granted. In the second use in Table 3. , "pipeline Postgres" value is assumed to be false by default because "pipeline Postgres" expression is not present and the downloaded data is saved to a file, not a database. In this case, the downloaded files are also in HTML format. In order to store scraped websites in a PostgreSQL database, the following steps has to be done primarily:

- A PostgreSQL database should be created.
- Database credentials (user name, password, database name) should be changed in setting.py file.
- The parameter "AllowOutboundLocalhost 1" in the torsocks configuration file (/etc/tor/torsocks.conf) should be uncommented. "torsocks.conf" and "setting.py" files comes with OnionCrawler software.

Its default start URL for crawling is "https://facebookcorewwwi.onion". As a default no search terms are set. If the search term is entered without paying attention to case sensitivity, the crawler scans both the uppercase and lowercase versions of the respective term. For example: tor, TOR, onion, ONION etc. This crawler's general usage is shown below.

torsocks scrapy crawl OnionCrawler [-a argument1=value1] [-a argument2=value2] [...]

B. TorBot

It is a Dark Web OSINT (Open Source Intelligence Tool for the Dark Web) Tool. It crawls addresses with onion extension. Some properties of this crawler is given below:

- Gets emails from site.
- Saves crawl info to JSON file.
- Crawls custom domains.
- Checks if the link is live.
- Built-in Updater.
- Returns Page title and address with a short description about the site.

In Table 4. , some prerequisites (libraries) to be installed before running TorBot software and default usage of TorBot is given. It is also possible to operate the Torbot crawler software in different ways with the arguments given in Table 5. In order to use the Torbot crawling software, the ubuntu terminal screen should be opened first and then the usage vesion of the crawler should be typed. TorBot can be operated by adding different arguments in each operation (such as torBot.py –h or torBot.py –s) or by combining several arguments one after another. In the usage in Table 4. , an execution mode is shown in which more than one argument is added.

Table 4. Prerequisites and default usage of TorBot

Prerequisites	**Default Usage**
Beautifulsoup4==4.6.0	python3 torBot.py
PySocks==1.6.7	**Other Usage Version**
PyQt5==5.11.3	torBot.py [-h] [-v] [--update] [-q] [-u URL] [-s] [-m] [-e EXTENSION] [-l] [-i] (By Adding Arguments to the Default Usage, Different Usage Forms can be Created.)
Requests==2.20.0	
Validators==0.12.2	
Yattag==1.10.0	
Pyinstaller==3.4.0	
Ete3==3.1.1	
Requests_mock==1.4.0	
Termcolor==1.1.0	
Tor	
Python 3.x	

Table 5. Some argument/value pairs to customize the TorBot crawler's behaviour

Argument/Value	**Usage Type**	**Description**
-h, --help	torBot.py -h	Shows This Help Message and Exit
-v, --version	torBot.py -v	Shows Current Version of TorBot.
--update	torBot.py --update	Updates TorBot to the Latest Stable Version
-q, --quiet	torBot.py -q	Prevents Header from Displaying
-u URL, --url URL	torBot.py –u torlinkbgs6aabns.onion	Specifies a Website Link to Crawl, Currently Returns Links on That Page
-s, --save	torBot.py -s	Saves Results to a File in JSON Format
-m, --mail	torBot.py -m	Gets E-mail Addresses from the Crawled Sites
-e EXTENSION	torBot.py –e com	Specifies Additional Website Extensions to the List (.com or .org etc)
-l, --live	torBot.py -l	Check if Websites are Live or Not
-i, --info	torBot.py -i	Info Displays Basic Info of the Scanned Site

C. OnionScan

Since OnionScan is a Go language based application, version 1.5 of the Go language should be installed on the ubuntu system before running the crawler. Then the TOR service should started by typing "service tor restart" phrase on the terminal screen. This crawler is used with 4 different functions. At the first phase of the OnionScan software, the onion links to be crawled are added to a text file as seed address and the crawling is performed by reading the links in this file line by line. This text file includes 8658 onion links and these links can be obtained by typing a query like ".onion link list" in the tor browser or from "torlinkbgs6aabns.onion" address. The data obtained by crawling these links are saved as JSON files that have the name of each onion link in an automatically created folder at run time. These files contains the following informations:

- url of the site
- date of crawling
- Whether the web page is active
- Other onion extension links
- Whether it contains some crawl words
- SSH Key information

Furthermore, if new onion links are found during crawling, except for the onion links originally provided, these links are added to the crawl list. The first step of the OnionScan crawler is executed by writing "python onionrunner.py" phrase to the terminal screen. At every stage of the OnionScan software there is a script that serves a different purpose. Therefore, the operation of each of these scripts is separate. The first script only aims to scan onion addresses and find new onion addresses. In the second stage of the OnionScan crawler, it is checked whether there are onion extension services with the same SSH [1]key and the connection between the secret services is determined. The second step of the OnionScan crawler is executed by typing "python sshkeys.py" phrase on the terminal screen. In the third phase of the crawler, the connection between the onion services which is linked to each other, is visualized by the Gephi program. This step is executed by typing "python hidden_services_graph.py" on the terminal screen. The "gexf" file obtained as a result of the execution of this script code, is then opened with Gephi program and the connections are visualized. In the last step, it is determined whether there are clones of an onion address given as a parameter, by using Scikit-Learn library. While performing this step, it is tried to find tfidf similarity between the onion addresses previously crawled and the address given as parameter. The last step of the OnionScan software is executed by running the "clone_finder.py" script.

D. TorScrapper

It is a scraper made in python with BeautifulSoup and Tor support. It scrapes onion and normal links, also save the output (web page content) in HTML format in a Output folder. Table 6. provides a list of prerequisites that should installed before the execution of this crawler and how the crawler is operated.

Table 6. Prerequisites and default usage of TorScrapper

Prerequisites	Default Usage
Python3	Crawler is Executed by Entering the Project Folder on the Terminal and Typing “python3 TorScrapper.py” Command.
Tor	
Beautifulsoup4==4.6.0	
PySocks==1.6.7	
Stem==1.5.4	
Tldextract	

[1] The SSH fingerprint which can be used to uniquely identify servers and devices, is a short sequence of characters that represents the larger public key of the server that is connected [19].

Before starting to install TorScrapper, it is recommended to create a virtual environment in the system and install the crawler here. It has been reported that installing TorScrapper in a virtual environment will help prevent conflicts with previously installed python packages throughout the system. The TOR service should be run before crawling process. Also, the onion links to be crawled shuld be added to a txt file under the project folder.

The data in HTML format, which is obtained as a result of crawler's execution, is saved in a folder under the project folder that is created automatically at the runtime. In addition, other links within the links given to crawler as parameters are included in the queue list, and the links that are finished scraping are removed from this list and added to the crawled list. Even if a single link is given to TorScrapper as a feed link, it is seen that it can reach many links with both onion and other extensions (such as .com, .net). In this context, it is a useful crawling bot for finding a large number of new URLs.

IV. RESULTS

In this study, 4 crawlers were examined. These crawlers are OnionCrawler, TorBot, TorScrapper and OnionScan. The results obtained by running each crawler are given below. According to this, Table 7. gives information about how long time each crawler completed the crawling of the target link that requested and how much data it obtains. According to this, it took more than 24 hours for the OnionCrawler software to crawl "uj3wazyk5u4hnvtk.onion", "facebookcorewwwi.onion", "torlinkbgs6aabns.onion" links in Table 7. The reason for this is that, the pages of these 3 addresses have a large number of interconnected sub-web pages and that each of these pages is accessed by downloading one by one. The rationale behind this view is that during the crawling of these 3 web pages, the crawler software is in a continuous data download state and the amount of data downloaded increases gradually. Therefore, the requests sent by OnionCrawler software to these connections did not cause any freeze and there was a continuous data download.

When Table 7. is examined for TorBot crawler software, it is seen that the data at the given address could be downloaded in the longest 17 minutes. This crawler software is not downloading the whole page content as in the OnionCrawler software, but only extracting the links at the given address and writing them to a file as JSON format. For this reason, TorBot has run faster than OnionCrawler software at all given addresses. However, it was able to store less data than OnionCrawler software.

When the data of the TorScrapper crawler in Table 7. is examined, it is seen that a working period of one minute is reserved for each address. This is due to the fact that each link terminates at different times, mostly for days and that the crawl takes longer than the other two crawler (OnionCrawler, TorBot) so that each link given to this crawler as a parameter is executed for one minute. When analyzed in terms of downloaded data, it is seen that the amount of data that OnionCrawler downloaded after a long operation time, was generally available in a shorter time by the TorScrapper software. Therefore, TorScrapper outperformed OnionCrawler. When these 3 crawling software in Table 7. are evaluated, it is understood that the software that can obtain the most amount of data is TorScrapper. Because this amount of data was obtained only during the one-minute execution period. If the execution time is extended, the amount of data downloaded will also increase. Among these 3 crawler software, TorBot is the lowest performing crawler software in terms of the amount of data it can download.

Table 7. Crawling Time and Downloaded Data Amount

	Links	Crawling Time (Responding and Downloading Related Content)	Downloaded Data Amount (Web Page Contents, Links as Html and Json Format)
OnionCrawler	msydqstlz2kzerdg.onion	9 sec	292.1 KB
	secmailw453j7piv.onion	6 sec	23.1 KB
	cardshopffielsxi.onion	13 min	1.1 MB
	4jcfkzkwbl6t3qq4.onion	14 sec	207.3 KB
	uj3wazyk5u4hnvtk.onion	Long-Term Crawling	11.4 GB
	matrixtxri745dfw.onion	23 sec	6.9 KB
	facebookcorewwwi.onion	Long-Term Crawling	182.8 MB
	torlinkbgs6aabns.onion	Long-Term Crawling	43.4 KB
TorBot	msydqstlz2kzerdg.onion	6 sec	200 byte
	secmailw453j7piv.onion	6 min	17 byte
	cardshopffielsxi.onion	10 min	57 byte
	4jcfkzkwbl6t3qq4.onion	5 min	57 byte
	uj3wazyk5u4hnvtk.onion	2 min	232 byte
	matrixtxri745dfw.onion	16 sec	56 byte
	facebookcorewwwi.onion	17 min	1.1 KB
	torlinkbgs6aabns.onion	2 min	10.2 KB
TorScrapper (Since Each Link Terminates at Different Times and so Crawls Take Longer, Each Link to This Crawler is Run for 1 Minute)	msydqstlz2kzerdg.onion	1 min	136.1 KB
	secmailw453j7piv.onion		4.4 KB
	cardshopffielsxi.onion		1.9 MB
	4jcfkzkwbl6t3qq4.onion		4.4 MB
	uj3wazyk5u4hnvtk.onion		1.2 MB
	matrixtxri745dfw.onion		256.1 KB
	facebookcorewwwi.onion		864.6 KB
	torlinkbgs6aabns.onion		97.7 MB

Table 8. shows how many links Torbot crawler has detected from the addresses given to it. Accordingly, "torlinkbgs6aabns.onion" is the address containing the maximum number of sublinks. This explains why the OnionCrawler and TorScrapper crawling software allocates a fairly long crawling time for "torlinkbgs6aabns.onion" link.

Table 8. The number of links detected by the Torbot crawler

	Crawled Addresses	Number of Detected URLs
TorBot	msydqstlz2kzerdg.onion	4
	secmailw453j7piv.onion	0
	cardshopffielsxi.onion	1
	4jcfkzkwbl6t3qq4.onion	1
	uj3wazyk5u4hnvtk.onion	7
	matrixtxri745dfw.onion	1
	facebookcorewwwi.onion	18
	torlinkbgs6aabns.onion	232

In Table 9. how many addresses are crawled in one minute by TorScrapper crawler and how many more addresses are waiting at queues for crawling in the same time period is presented. Accordingly, during the one minute execution time, the maximum number of link was crawled in the case where the address "torlinkbgs6aabns.onion" was given as a parameter. The maximum number of link waiting to be crawled in the queue occurred when the address "uj3wazyk5u4hnvtk.onion" was given as a parameter.

Table 9. Number of URLs crawled and waiting to be crawled by TorScrapper

		Number of URLs Crawled in 1 Min.	**Pending URL in Queue in 1 Min.**
TorScrapper	msydqstlz2kzerdg.onion	2	11
	secmailw453j7piv.onion	5	0
	cardshopffielsxi.onion	216	486
	4jcfkzkwbl6t3qq4.onion	1584	19147
	uj3wazyk5u4hnvtk.onion	220	19212
	matrixtxri745dfw.onion	3	32
	facebookcorewwwi.onion	111	5763
	torlinkbgs6aabns.onion	3698	0

A. Findings After Operating OnionCrawler

- It crawls single URL link and stores it to the file system. When this type of crawling has been done, it stores index page and other pages of the crawled website. It automatically opens a folder whose name is the onion link name that is being crawled, in the project folder.

- Also it crawls default website (https://facebookcorewwwi.onion) in a seamlessly manner. It stores facebook pages to the file system according to every country domain. (.tr, .fr etc.)

- When “http://torlinkbgs6aabns.onion” link is crawled, a continuous waiting situation is encountered. The request to that address is not answered for a long time.

- Among the given URL addresses, there were problems in 3 addresses (uj3wazyk5u4hnvtk.onion, facebookcorewwwi.onion, torlinkbgs6aabns.onion). In the crawlings where these addresses were given as parameters, very long waiting periods were encountered. Also, although it is seen that the tables related to the records are in the database, the contents of the tables could not be reached even if pgadmin4 was used.

- The default URL address (https://facebookcorewwwi.onion) and each of the sublinks in this address have been crawled quickly in itself, but since the default address contains a large number of sub-addresses, reaching all the addresses one by one and crawling each one and downloading the contents of these pages has been a long time-consuming process.

B. Findings After Operating TorBot Crawler

- python3 torBot.py –u onionlink → when a crawl is done in this way, all links in the given URL are listed.

- python3 torBot.py –u onionlink –s → in this way, all links in the given URL are saved in the JSON format under the project page of the crawler.

- The control of whether a given URL is active, does not work fully.
- python3 torBot.py –u onionlink –info→ This type of crawling operation was run smoothly.

- python3 torBot.py –u onionlink –m → This type of crawling operation was run smoothly.

C. Findings After Operating TorScrapper

- It can crawl faster and more detailed.
- In the determined period of operation, it crawled a large number of URL addresses in a short time and wrote these addresses to the related file.
- It was also able to write a large number of new URL addresses, which were waiting to be crawled, to the corresponding text file during its execution.

D. Findings After Operating OnionScan

When the OnionScan crawler is evaluated in general, it takes a long time to crawl all of the given onion lists and save them as a JSON file. After 24 hours of scanning, only 336 addresses were examined. Therefore, it is anticipated that it will take quite a long time to scan the entire list (8658 onion links). Although the OnionScan crawler scans the address it receives as a parameter in detail, the fact that scanning takes too long is considered a factor that diminishes the preferability of this crawler.

V. CONCLUSION

Within the scope of the study, 4 crawler software were examined. These softwares have been evaluated according to various criteria such as crawl time, amount of data downloaded, the number of detected links, count of crawled URLs and amount of the URLs waiting to be crawled in the queue. In this respect, TorScrapper, which performs the necessary download process by detecting a large number of new sublinks related to the link given as a parameter and started crawling a new link with a fast circulation, has been found to be the most performing crawling software among the examined softwares. Except of this, TorScrapper was the most trouble-free crawling software that is able to run properly, among the examined crawling software. In addition, the amount of data stored by this crawling software in one-minute execution time is higher than TorBot and OnionCrawler software. The only disadvantage of the TorScrapper software is that it requires a long time to crawl each link address due to it performs a extensive crawl. However, it is appropriate to use this scanning software when time is not a problem and web page content is wanted to be downloaded in HTML format.

TorBot software, on the other hand, is inadequate compared to other crawling software since it only determines the links in the links given as parameters and does not write the content information of these pages in HTML, JSON formats. However, since it is capable of bringing results in a short time, it can be preferred only in the situations that the researchers aiming to determine the links on the pages.

The OnionCrawler crawling software is similar to the TorScrapper software in terms of the type of data it stores (in HTML format). To be able to select the links that contain some special expressions (such as Tor, onion, bitcoin) and be able to crawl these filtered addresses, is a positive feature of OnionCrawler software. However, considering the 3 connections (“uj3wazyk5u4hnvtk.onion”, “facebookcorewwwi.onion”, “torlinkbgs6aabns.onion”) given as a parameter, this software has a slow crawling performance so it took longer to crawl also in general, it was able to store less data than TorScrapper software. In order to be able to perform a continuous active crawl and to avoid having to re-establish a connection due to a disconnection with the requested address it would be more reasonable to prefer TorScrapper rather than OnionCrawler.

The OnionScan crawler provides more detailed analysis than the other 3 crawlers in terms of the information it provides (crawling the given links more detailed, checking if the given link has the same SSH key with the link given previous, visualizing the connection between the addresses, determining whether a given onion address has been cloned). However, because it takes days for the links to be crawled, it remains behind TorScrapper in terms of performance, even though it is a more thorough crawler than the TorScrapper. In cases such as downloading the content of the requested address in JSON format, checking whether the page content of one address exists at another address and filtering the pages that contain some special expressions, choosing the OnionScan crawling software, will be the proper choice that leads to the target. Because TorScrapper does not have the capability to provide these features.

The data obtained from this study provided an insight into the characteristics and scanning performances of the crawling softwares examined. In this respect, the study provides guidance for researchers who will be interested in the crawling and analysis process within the TOR network.

REFERENCES

[1] AlKhatib, B., Basheer, R. (2019). Crawling the Dark Web: A Conceptual Perspective, Challenges and Implementation. *Journal of Digital Information Management*, 17(2), 51-60.

[2] Hoelscher, P. (2018). *What is the Difference Between the Surface Web, the Deep Web, and the Dark Web?* Infosec Resources, https://resources.infosecinstitute.com/what-is-the-difference-between-the-surface-web-the-deep-web-and-the-dark-web/#gref, (01.12.2019).

[3] Zabihimayvan, M., Sadeghi, R., Doran, D., Allahyari, M. (2019). A Broad Evaluation of the Tor English Content Ecosystem. In Proceedings on WebSci 2019, June 30–July 3, Boston, Massachusetts, 333-342.

[4] Owen, G., Savage, N. (2016). Empirical analysis of Tor Hidden Services. *IET Information Security*, 10(3), 113-118.

[5] Park, J., Mun, H., Lee, Y. (2018). Improving Tor Hidden Service Crawler Performance. In 2018 IEEE Conference on Dependable and Secure Computing (DSC), 10-13 December, Kaohsiung, Taiwan, 1-8.

[6] Casenove, M., Miraglia, A. (2014). Botnet over Tor: The illusion of hiding. In 2014 6th International Conference On Cyber Conflict, 3-6 June, Tallinn, Estonia, 273-282.

[7] Pundhir, S., Rafiq, M. Q. (2011). Performance Evaluation of Web Crawler. In IJCA Proceedings on International Conference on Emerging Technology Trends (ICETT), 43-46.

[8] Achsan, H. T. Y., Wibowo, W. C. (2014). A Fast Distributed Focused-Web Crawling. *Procedia Engineering*, 69, 492-499.

[9] Yadav, M., Goyal, N. (2015). Comparison of Open Source Crawlers-A Review. *International Journal of Scientific & Engineering Research*, 6, 1544-1551.

[10] Dikaiakos, M., Stassopoulou, A., Papageorgiou, L. (2003). Characterizing Crawler Behavior from Web Server Access Logs. In E-Commerce and Web Technologies, 2-5 September, Prague, 369-378.

[11] Zulkarnine, A. T., Frank, R., Monk, B., Mitchell, J., Davies, G. (2016). Surfacing collaborated networks in dark web to find illicit and criminal content. In 2016 IEEE Conference on Intelligence and Security Informatics (ISI), 28-30 September, Tucson, AZ, 109-114.

[12] Baravalle, A., Lopez, M. S., Lee, S. W. (2016). Mining the Dark Web: Drugs and Fake Ids. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 12-15 December, Barcelona, 350-356.

[13] Kalpakis, G., Tsikrika, T., Iliou, C., Mironidis, T., Vrochidis, S., Middleton, J., Williamson, U., Kompatsiaris, I. (2016). Interactive Discovery and Retrieval of Web Resources Containing Home Made Explosive Recipes. In International Conference on Human Aspects of Information Security, Privacy, and Trust, 17 - 22 July, Toronto, 221-233.

[14] Iliou, C., Kalpakis, G., Tsikrika, T., Vrochidis, S., Kompatsiaris, I. (2016). Hybrid Focused Crawling for Homemade Explosives Discovery on Surface and Dark Web. In 2016 11th International Conference on Availability, Reliability and Security (ARES), 31 August-2 September, Salzburg, 229–234.

[15] Zhang, Y., Zeng, S., Huang, C., Fan, L., Yu, X., Dang, Y., A Larson, C., Denning, Roberts, N., Chen, H. (2010). Developing a Dark Web collection and infrastructure for computational and social sciences. In

2010 IEEE International Conference on Intelligence and Security Informatics, 23-26 May, Vancouver, BC, 59–64.

[16] Ghosh, S., Das, A., Porras, P., Yegneswaran, V., Gehani, A. (2017). Automated Categorization of Onion Sites for Analyzing the Darkweb Ecosystem. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2017, Halifax, Nova Scotia, 1793–1802.

[17] Pannu, M., Kay, I., Harris, D. (2018). Using Dark Web Crawler to Uncover Suspicious and Malicious Websites. In International Conference on Applied Human Factors and Ergonomics, 21-25 July, Orlando, Florida, 108-115.

[18] Raghavan, S., Garcia-Molina, H. (2001). Crawling the Hidden Web. In Proceeding VLDB '01 Proceedings of the 27th International Conference on Very Large Data Bases, 11 – 14 September, San Francisco, CA, 129-138.

[19] Seitz, J. (2016). *Dark Web OSINT with Python Part Two: SSH Keys and Shodan on Automating OSINT*. Automating OSINT, http://www.automatingosint.com/blog/2016/08/dark-web-osint-with-python-part-two-ssh-keys-and-shodan/, (28.07.2019).