4-4-2018

Crawling Tor's Hidden Services and Depicting their Interconnectivity

John-Luke N. Navarro
Andrews University, johnluke@andrews.edu

J. N. Andrews Honors Program
Andrews University

HONS 497
Honors Thesis

Crawling Tor's Hidden Services and Depicting their Interconnectivity

John-Luke Navarro

4/4/2018

Advisor: Dr. Rodney L. Summerscales

Primary Advisor Signature: 

Department: Engineering and Computer Science

Crawling Tor's Hidden Services and Depicting their Interconnectivity

John-Luke N. Navarro

***Abstract*— The Tor network is a popular online privacy platform that enables anonymous browsing, but is also notorious for the vast number of illicit marketplaces, goods and services available to users. Tor's protocols also secure hosted websites, known as hidden services, against unwanted tracking and location. Tor has attracted the attention of law enforcement agencies, whom are interested in hidden service data analysis. However, little large-scale analysis is currently performed. To help address this issue, I constructed two tools. A specialized web crawler downloads bulk page-content from Tor's hidden services. This program crawls the Tor Network broadly, securely, and with more ease of use than other similar solutions. A graphing program depicts connections between crawled hidden services as directed graphs. Both of these tools allow investigative agencies and researchers to more effectively gather and analyze Tor network content.**

I. INTRODUCTION

On October 2013, the notorious digital black-marketplace Silk Road was shut down by law enforcement. For two years, Silk Road allowed users to buy and sell illicit products on the internet with less concern for getting caught, as access was exclusive to users connecting through the Tor Network. Tor is free software that guarantees privacy for it's users by obfuscating network routing, encrypting their data, and other anonymization techniques. Thousands of websites known as hidden services are hosted similarly to Silk Road on Tor. Tor's protocols also protect hidden service hosts and servers from being located [1].

The vast number of illicit hidden services similar to Silk Road have caught the attention of law enforcement agencies. However, due to the obfuscated and discontinuous nature of Tor, discovering and investigating vast numbers manually can be extremely inefficient. In this paper, I describe the development of two pieces of software[1] that automatically perform hidden service content-collection and depict site connections visually. The first program is a web crawler developed to crawl the Tor network securely and automatically. This crawler downloads web-pages from hidden services and notes links between them. The second program processes these links into connected graphs that can be rendered by external applications.

The web crawler is relatively easy to configure and run; it requires only the necessary software libraries installed, a starting list of hidden service URLs to begin crawling from, and a configured local SOCKS-compatible proxy server. From the graphing program's connected graphs, information such as popular hidden services or browsing patterns of Tor users can be extrapolated.

[1]https://github.com/Argonne-National-Laboratory/torantula

II. RELATED WORK

A. Cryptopolitik and the Darknet

Moore and Rid discuss political and technological ramifications of increased cryptography, then explore in detail Tor network history and functionality. The authors also developed a specialized web crawler to explore Tor Network content. Features they developed such as depth and page-count limits were inspirational towards similar implementations in my software. The authors state their starting list of hidden services was crawled in two months, but don't state if they reiterated over that list. This might indicate potential for improving speed.

B. The Tor Dark Net

Owen and Savage performed two projects; the first involved operating a series of Tor relays and mining data about hidden service descriptors and the number of requests each received. The authors noticed that a very small number of hidden services persisted over the course of their eighteen month project. The second project involved operating a web crawler to download hidden service content and extract "key data points". The authors make the claim that broad, long-term crawls risk over-representing short-lived hidden services. [2]

C. Towards a Comprehensive Insight into the Thematic Organization of the Tor Hidden Services

Spitters, et. al describe briefly Tor network operations and types of illegal content that are common on the platform, before affirming that access to Tor content analysis by organizations could be useful towards assisting investigations or preparing for possible attack vectors. The authors also operated their own Tor network crawler and note several important design features, such as breadth-first crawling and per-site download limitations. They operated their crawler for about a year, and note that within the span of five months, their crawler "discovered almost two thousand new hidden service addresses" [3].

III. METHODOLOGY

Both the crawler and the graphing program were developed in Python 3.6 on Ubuntu 16.04. The crawler was developed with Scrapy[4], a framework for developing web data extraction tools, including web crawlers. The graphing program was developed with NetworkX[5], used to create, manipulate, and analyze network graphs. Privoxy[6] was the local proxy server software used to connect the crawler to Tor, although other proxy servers could be substituted.

The crawler begins performing hidden service HTTP requests from a user-provided list of hidden service URLs, and iterates through each over the course of a "session." Responses from hidden services consisting of HTML are downloaded directly into a session-specific directory created by the crawler's data pipeline at runtime. A local MySQL database that stores metadata for top-level domains, URLs, and the date and time can be enabled or disabled.

A log file stored locally contains real-time data on web-pages requested, domains ignored, exceptions raised, crawl statistics, and other information during the course of a crawling session.

A. Connecting the crawler to Tor

By default, web crawlers developed with Scrapy are unable to interface with Tor, as Scrapy doesn't support the SOCKS protocol which Tor depends on. However, Scrapy is compatible with HTTP proxies, which can interface with Tor through SOCKS. A local Privoxy server was configured to connect to the Tor service through the SOCKS protocol. The crawler then interfaces with this server as a HTTP proxy, which then relays requests to Tor through SOCKS.

B. Crawler exploration and downloading strategies

Specifying a positive depth priority and indicating to the request scheduler to operate in FIFO order allows the crawler to operate in a breadth-first-search manner. One benefit is a representative sample of the domain names expected in the final dataset manifests earlier in the dataset directory during a crawling session. This occurs because the crawler attempts to download the homepage of each queued hidden service URL (which may succeed or fail) before downloading additional pages at greater depths.

Limiting the depth of pages crawled on hidden services has several benefits. For instance, this protects the crawler from recursively exploring links that loop within domains (as each link is treated as one depth deeper). This also help prevent over-representing data from a hidden service's deeper pages, which may be too specific to describe the site's general purpose. Most hidden services will display the content that best reflects the nature of their site within the first few pages.

Limiting the number of web-pages downloaded from each hidden service keeps hidden service-specific datasets from becoming too large. Once the crawler notes that it has downloaded the maximum number of pages from a domain, that domain is excluded from further requests and downloads during the crawling session. This also prevents the crawler from following links back to hidden services that have already been crawled.

These two strategies of limiting page-depths and page-numbers guarantee that only a limited number of relatively "shallow" web pages represent each hidden service.

C. Crawler data pipeline strategies

The crawler pipelines are responsible for sorting, storing, and logging downloaded page content and metadata. A session-specific directory, labeled with the current time and date, is created by the crawler at the beginning of every crawling session. For each hidden service crawled during a session, a domain-specific directory is inserted into this session directory. It is into these domain-specific directories that the contents of each crawled hidden service are inserted. Each web-page's HTML is saved as a text file, named for the URL that it was downloaded from.

All the links found on a hidden service are appended into a `found_links.txt` file generated for each crawled domain. This is placed into each domain-specific directory.

D. Crawler user protections

By connecting to the Tor network, the crawler automatically has the same anonymity protections as other normal Tor users, including encryption of data and IP addresses. As a preventative measure against crawler banning, for each outbound request the crawler randomly selects a Tor-browser user-agent from a predefined list. These user-agents reflect a variety of different operating systems connecting to Tor through the Tor browser. Although the crawler may still appear automated in terms of speed and algorithmic exploration, user-agent spoofing may still protect against some site policies that disallow non-human interaction.

The crawler checks the text of downloaded pages for blacklisted user-defined keywords, and if any are found, the matching content is discarded and all further requests or responses from the source-domain are ignored for the remainder of the crawling session. This prevents the crawler from downloading illicit or unwanted information, as long as the blacklisted keywords are defined by the user.

E. Graphing Program functionality

The graphing program accepts as input one of the crawl-session directories generated by the crawler. A directed graph is initialized using the NetworkX library. Each domain subdirectory within the crawl-session directory indicates a hidden service, so the title (which reflects the domain name) of each subdirectory is appended to the directed graph as a node. The program then iterates through each of these domain subdirectories, searching for `found_links.txt` files that indicate the crawler encountered links to other hidden services. Each one of these files is parsed for links, which are inserted as edges into the directed graph with the current subdirectory node as the source and the discovered link as the destination. These graphs are then dumped as `.json` and `.gexf` formats, which can be rendered or analyzed by a variety of applications.

IV. RESULTS

A. Crawler Results

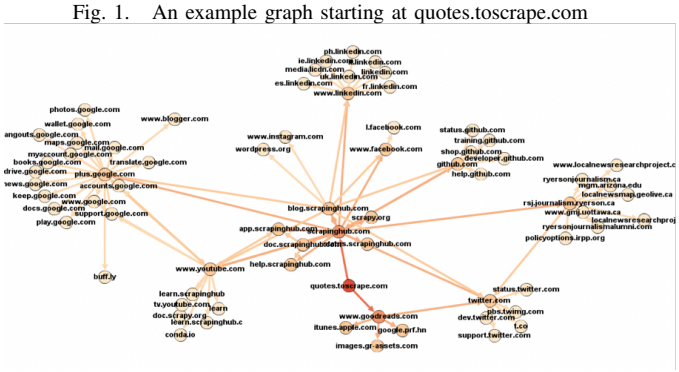
The crawler's initial list of hidden services was populated with ten-thousand known hidden service URLs collected by Ahmia's Tor Search Engine [7]. This list includes The Hidden Wiki, one of the most popular start-pages for Tor users [2]. This list was chosen with the expectation that starting with an extensive, representative list is more likely to yield new hidden service discoveries.

Over a four-hour crawling session, the web crawler iterated over all the URLs on the list, crawling each that was operational, while following links to other hidden services. The actual number of hidden services finally crawled (roughly one-thousand two-hundred) is much less than the initial list of provided URLs, but this was to be expected due to the short-lived nature of roughly 85% of hidden services [2]. There were no indications that the crawler was impeded or blocked by any hidden service. The 1.5 GB of HTML downloaded from these hidden services does not contain duplicate or other undesirable content. The downloaded content, sorted by domains in nested directories, was parseable by the graphing program and other analysis tools.

B. Graphing Program Results

The graphing program successfully parses through the data downloaded by the web crawler and generates connected graphs depicting links between the crawled hidden services. These graphs are then dumped in `.json` and `.gexf` formats. When rendered by external applications, these graphs initially appear chaotic, but can be made more legible if a force-directed graph-drawing algorithm is applied.

Fig. 1. An example graph starting at quotes.toscrape.com



V. DISCUSSION

Due to some security concerns that the crawler downloaded personally-identifiable or classified textual data, extensive quantitative analysis of the dataset was not possible. As the scope of this project was to develop software that downloaded HTML from hidden services, it was not difficult to verify that the software performed this function, and that downloaded HTML matched the corresponding crawled hidden service. Any hidden services that stored the bulk of their page content several pages deep might not have enough representative data within the dataset, due to the depth limit imposed on the crawler.

The crawler is inherently limited in its crawling scope by the initial hidden service list. The crawler may discover new hidden services, but they will have been linked to on known hidden services. If a hidden service exists and is not on the initial list, isn't linked to on any of the listed hidden services, or its URL is only shared privately, the crawler will not discover it. Generating hidden service URLs and attempting to crawl them is an intentionally difficult task computationally. In any case, hidden services that attempt to attract customers will not hide their URLs. The most trafficked hidden services are almost guaranteed to be on the initial list.

At any time, only one copy of the crawler was running, iterating through the entire starting list of hidden services. Multiple copies of the crawler could be run simultaneously, each processing a portion of the starting hidden service list. This would result in faster data collection and redundancy in the event of a failure.

Although the MySQL database was only used to store crawl-session metadata, it could also be used to store full page-texts. This would allow easier interfacing with the entire dataset by other applications not specifically designed to work with the crawler.

This version of the web crawler and graphing program was designed mainly with functionality in mind and was tested only on a single platform (Ubuntu). For wider release and use by organizations, it would need to be tested on a variety of platforms, and made easier to install and use.

Future versions of this project could store more information between sessions, so the crawler could avoid redundant work. For example, the crawler might note which of the URLs in the initial hidden service list have not been operational, and either only periodically attempt requests, or discard them completely. Furthermore, the crawler could automatically update the initial list with any new hidden service discoveries, to be directly accessed on future crawls.

Eventually, the content collected by the crawler and the graphs generated by the graphing program should be synthesized to perform analysis on Tor network content, and enable the generation of real-time content reports on Tor Network content. This capability will be valuable for both law-enforcement professionals and researchers who seek to understand the Tor network.

ACKNOWLEDGMENTS

I would like to thank Joshua Lyle at Argonne National Laboratory for mentoring and guiding this project, and encouraging this learning opportunity. I would also like to thank Rodney L. Summerscales, Ph.D., at the Andrews University department of Engineering and Computer Science for instruction and guidance on preparing this paper and other deliverables for this project.

REFERENCES

[1] Daniel Moore, Thomas Rid. *Cryptopolitik and the Darknet*. Survival, vol. 58, no. 1, Feb. 2016, pp. 7-38., doi:10.1080/00396338.2016.1142085

[2] Gareth Owen, Nick Savage. *The Tor Dark Net* Global Commission on Internet Governance, ser. 20, 30 Sept. 2015. CIGI publications, `www.cigionline.org/publications/tor-dark-net`

[3] Martijin Spitters, Stefan Verbruggen, Mark van Staalduinen *Towards a Comprehensive Insight into the Thematic Organization of the Tor Hidden Services* 2014 IEEE Joint Intelligence and Security Informatics Conference, 2014, doi:10.1109/jisic.2014.40.

[4] *Scrapy.* Scrapy, 2017. https://scrapy.org/

[5] *NetworkX.* NetworkX, 2017. https://networkx.github.io/

[6] *Privoxy.* Privoxy, 2017. https://www.privoxy.org/

[7] *Ahmia.* Ahmia, 2017. https://ahmia.fi/